Representation and Generalization in Autonomous Reinforcement Learning

vorgelegt von

Diplom Informatiker

Wendelin Böhmer

geb. in Waddeweitz

von der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften

– Dr. rer. nat. –

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr. Henning Sprekeler
Gutachter:	Prof. Dr. Klaus Obermayer
Gutachter:	Prof. Dr. Marc Toussaint
Gutachter:	Prof. Dr. Manfred Opper

Tag der wissenschaftlichen Aussprache: 16. Dezember 2016

Berlin 2017

Abstract

This PhD thesis investigates the role of representations that generalize values in autonomous reinforcement learning (RL). I employ the mathematical framework of function analysis to examine inductive and deductive RL approaches in continuous (or hybrid) state and action spaces. My analysis reveals the importance of the representation's metric for inductive generalization and the need for structural assumptions to generalize values deductively to entirely new situations. The thesis contributes to two related fields of research: *representation learning* and *deductive value estimation* in large state-action spaces. I emphasize here the agent's autonomy by demanding little to no knowledge about (or restrictions to) possible tasks and environments. In the following I summarize my contributions in more detail.

I argue that all isomorphic state spaces (and fully observable observation spaces) differ only in their metric (Böhmer et al., 2015), and show that values can be generalized optimally by a diffusion metric. I prove that when the value is estimated by a linear algorithm like least squares policy iteration (LSPI), slow feature analysis (SFA) approximates an optimal representation for all tasks in the same environment (Böhmer et al., 2013). I demonstrate this claim by deriving a novel regularized sparse kernel SFA algorithm (RSK-SFA, Böhmer et al., 2012) and compare the learned representations with others, for example, in a real LSPI robot-navigation task and in extensive simulations thereof. I also extend the definition of SFA to γ-SFA, which represents only a specified subset of "anticipated" tasks.

Autonomous inductive learning suffers the *curse of insufficient samples* in many realistic tasks, as environments consisting of many variables can have exponentially many states. This precludes inductive representation learning and both inductive and deductive value estimation for these environments, all of which need training samples sufficiently "close" to every state. I propose structural assumptions on the state dynamics to break the curse. I investigate state spaces with sparse conditional independent transitions, called Bayesian dynamic networks (DBN). In difference to value functions, sparse DBN transition models can be learned inductively without suffering the above curse. To this end, I define the new class of linear factored functions (LFF, Böhmer and Obermayer, 2015), which can compute the operations in a DBN, marginalization and point-wise multiplication, for an entire function analytically. I derive compression algorithms to keep LFF compact and three the inductive LFF algorithms density estimation, regression and value estimation.

As inductive value estimation suffers the curse of insufficient samples, I derive a deductive variant of LSPI (FAPI, Böhmer and Obermayer, 2013). Like LSPI, FAPI requires predefined basis functions and can thus not estimate values autonomously in large state-action spaces. I develop therefore a second algorithm to estimate values deductively for a DBN (represented by LFF, e.g., learned by LFF regression) directly in the function space of LFF. As most environments can not be perfectly modeled by DBN, I discuss an importance sampling technique to combine inductive and deductive value estimation. Deduction can be furthermore improved by mixture-of-expert DBN, where a set of conditions determines for each state which expert describes the dynamics best. The conditions can be constructed as LFF from grounded relational rules. This allows to generate a transition model for each configuration of the environment. Ultimately, the framework derived in this thesis could generalize inductively learned models to other environments with similar objects.

Zusammenfassung

Diese Doktorarbeit untersucht die Rolle der Zustandsrepräsentation beim generalisieren der *Valuefunktion* im autonomen *reinforcement learning* (RL). Ich nutze mathematische Techniken der Funktionsanalysis um induktive und deduktive Ansätze für RL in kontinuierlichen Zustands- und Aktionsräumen zu analysieren. Meine Analyse offenbart den zentralen Einfluss der Repräsentationsmetrik auf induktive Generalisierung und die Notwendigkeit von strukturellen Annahmen um deduktiv *values* zu komplett neuen Situationen generalisieren zu können. Die Arbeit leistet Beiträge zu zwei verwandten Feldern: *induktives Repräsentationslernen* und *deduktive Schätzung* der Valuefunktion in großen Zustands- und Aktionsräumen. Ich betone dabei die Autonomie des Agenten, indem ich die zu erlernenden Aufgaben nicht einschränke und kaum Vorwissen voraussetze. Im Detail sind meine Beiträge:

Isometrische Zustands- und Beobachtungsräume unterscheiden sich nur in ihrer Metrik (Böhmer et al., 2015), und ich zeige dass *values* optimal durch eine Diffusionsmetrik generalisiert werden. Ich beweise dass, wenn *values* von einem linearen Algorithmus wie *least squares policy iteration* (LSPI) geschätzt werden, *slow feature analysis* (SFA) eine optimale Repräsentation für alle Aufgaben in der gleichen Umgebung approximiert (Böhmer et al., 2013). Ich demonstriere dies indem ich einen neuen *regularized sparse kernel SFA* Algorithmus herleite (RSK-SFA Böhmer et al., 2012) und diesen mit anderen erlernten Repräsentationen vergleiche, zum Beispiel anhand einer LSPI Navigationsaufgabe mit einem realen Roboter und in umfangreichen Simulationen. Zusätzlich erweitere ich SFA zu γ-SFA, welches eine optimale Repräsentation für eine vorher definierbare Teilmenge von "zu erwartenden" Aufgaben erzeugt.

Autonomes induktives Lernen leidet unter dem *curse of insufficient samples*, da Umgebungen, welche durch viele Variablen beschrieben werden, oft exponentiell viele Zustände haben. Induktive Generalisierung ist aber darauf angewiesen dass jeder Zustand "in der Nähe" von Trainingsbeispielen liegt. Die vielen Zustände machen daher induktives Repräsentationslernen und sowohl induktive als auch deduktive Schätzung der Valuefunktion praktisch unmöglich. Der Ansatz dieser Arbeit ist diesem Problem durch strukturelle Annahmen an die Zustandsdynamik zu begegnen. Ich untersuche Zustandsräume mit bedingt unabhängigen Zustandsänderungen, welche *Bayesian dynamic networks* (DBN) genannt werden. Modelle dieser Dynamik können, im Unterschied zu Valuefunktion, induktiv gelernt werden ohne unter dem oben genannten Fluch zu leiden. Hierfür definiere ich die Funktionsklasse der *linear factored functions* (LFF, Böhmer and Obermayer, 2015), welche die notwendigen Operationen zur Berechnung eines DBN, Marginalisierung und punktweise Multiplikation, für die gesamte Funktion analytisch durchführen können. Ich leite Kompressionsalgorithmen, welche LFF kompakt halten, und drei induktive Lernalgorithmen her: Dichteschätzung, Regression und Valueschätzung.

Da induktive Schätzung der Valuefunktion unter dem *curse of insufficient samples* leidet, leite ich eine deduktive Variante von LSPI her (FAPI, Böhmer and Obermayer, 2013). FAPI benötigt allerdings, genau wie LSPI, im voraus definierte Basisfunktionen, und kann daher die Valuefunktion in großen Zustands- und Aktionsräumen nicht ohne Vorwissen schätzen. Ich leite deshalb einen zweiten Algorithmus her, welcher die Valuefunktion direkt im Raum der LFF mittels eines DBN deduktiv schätzt. Dieses DBN kann z.B. durch LFF-Regression gelernt werden. Da

sich jedoch viele Dynamiken nicht perfekt durch DBN modellieren lassen, führe ich ebenfalls eine *importance sampling* Technik ein, mit der man induktive und deduktive Schätzungsmethoden vereinen kann. Die deduktive Schätzung kann zusätzlich verbessert werden, indem man ein *mixture-of-expert* DBN nutzt. Hierbei bestimmt eine Reihe von Konditionen für jeden Zustand *welcher* Experte die Dynamik am besten beschreibt. Die Konditionen sind ihrerseits LFF, welche man aus *grounded relational rules* berechnen kann. Dies erlaubt für jede Konfiguration der Umgebung ein eigenes Modell der Dynamik zu erzeugen. Die in dieser Doktorarbeit vorgestellte Methodik sollte es daher erlauben induktiv erlernte Modelle der Dynamik auf unbekannte Umgebungen mit ähnlichen Objekten zu übertragen, um hiermit deduktiv die Valuefunktion zu schätzen.

Acknowledgements

This thesis is the crowning achievement of my long and windy road to science. After being a mediocre pupil at school, I studied computer science without much passion. I would never have ended up in science if not for Klaus Obermayers courses "Neuronale Informationsverarbeitung I+II". Klaus introduced me to the foundations of machine learning (ML) and neural networks, and for some reason, something in the back of my head just made *click*. I was always fascinated by the philosophical questions about the human mind, but were never satisfied with the way philosophers were trying to answer them. ML offered a *constructive* computational interpretation of "thinking", and the mathematical tools to analyze it in an unambiguous way. As a classically trained computer scientist, the computational part was trivial, but it took me another three years of studying to catch up with the math. In all that time, Klaus supported me by giving me the job of a student assistance in the NeuRoBot project under the supervision of Steffen Grünewälder. After my diploma in 2008, he also helped me to secure a scholarship from the integrated graduate program "human-centric communication" at the TU-Berlin. During the last 7 years, I was essentially free to do research on practically anything I wanted. Klaus made sure I did not waste my time on impossible or trivial problems, but always gave me free rein to pursue ideas of my own. I used this freedom to think far, maybe sometimes a little too far, outside the box and condensed my ideas in two successful project proposals for the DFG priority program PPR-1527 "autonomous learning". This thesis is both a summary of my work and a guide into the way that I, in my own flawed and unique ways, have started to interpret the fundamental question of what "thinking" is, and how any process, animal or human may be able to do it.

I first and foremost have to thank therefore Klaus Obermayer, for introducing me to science and for his continued support during my PhD studies. I also must give credit to Steffen Grünewälder, who awoke my interest in reinforcement learning. Furthermore, I need to thank Yun Shen, with whom I discussed many of my theories, and who was always able to find the flaws in my math. Much of my theoretical work would have been impossible without him. I also need to acknowledge the people who gave the inspiration for the work presented in this thesis. My work on SFA representation started with a paper by Mathias Franzius, Henning Sprekeler and Laurenz Wiskott, and I thank Mathias and Henning for giving me a personal introduction into the topic. I also thank Jost Tobias Springenberg and Joschka Boedecker for their excellent summary of end-to-end RL and auto-encoder representations in my 2015 review paper. The idea for linear factored functions, the corner stone of the entire Chapter 5, came from Marc Toussaint. After I tried to pitch him an idea for a project (that would never have worked), he casually wrote down a factored value function to make his point. The image stuck with me and became the core of the first successful DFG project proposal I was involved in. I also thank my office-mate Rong Guo for enduring my long periods of bad mood, and all the other members of the neural information processing group at the TU-Berlin.

In my personal life I want to thank my mother, who raised me all by herself, and my father, who always supported my during my studies, no matter how long they lasted. I also thank my friends Chang, Töffy, Till, Arne, Hans-Phillip, Thom, Inga and Steffi, my sister Eva, my brother Fabian, as well as my roommates Ralph, Malte, Tobi, Maria and Serena, for enduring me and my many problems during all that time.

Contents

List of Tables

List of Theorems

List of Figures

List of Algorithms

Chapter 1

Introduction

The 21st century is expected to be the breakthrough for many sciences, for example, the century of biology (Venter and Cohen, 2004), the century of the brain (Bear et al., 2001) and the century of renewable energy (Bostan et al., 2012). Only time can tell whether these predictions will come true, but one can make an educated guess based on the practical *need* for a particular technology. For example, should this century fail to gain major insights in inner workings of our brain, humanity may simply declare the 22nd century the century of the brain. If we fail to shift our energy production to renewable, on the other hand, there probably will be no 22nd century that can support such a scientific endeavor. However, such a shift in technology will yield more and more decentralized infrastructure, like "smart-grids" or "industry 4.0", that has to be monitored and controlled. Although less prominent in the mind of the public, I believe that the 21st century will also be a century of *autonomous control*. However, autonomous control is also at the core of a much larger question: how animal- or human-like minds are constructed. Very much in line with the century of the brain, the field investigates mathematically and computationally how an embodied agent can *learn* to interact with its environment in a meaningful way. These are the questions that sparked my interest in science and philosophy for as long as I can remember. Answering them should provide insights for psychology and neuroscience, but also lead to practical applications in robotics and autonomous vehicles. The techniques I developed are universal and should be applicable to decentralized systems as well, but my research focuses on embodied agents and the thesis is written in this spirit.

Full autonomy requires a different approach to control than traditional *artificial intelligence* or *robotics*, though. These disciplines are based on exact understanding and formalization of the controlled system (Russell and Norvig, 2009), which allows very little room for adaptivity and learning. The relatively young field of *machine learning* developed an alternative approach to autonomous control. Sutton and Barto (1998) gave birth to the class of *temporal difference learning* algorithms, to model conditioning in psychology, and named the young discipline in this terminology *reinforcement learning* (RL). RL is mathematically well-grounded in *Markov decision processes* (MDP, Bellman, 1957), which describe control in stochastic environments. The goal is to find a control policy that maximizes the *rewards* the agent will receive in the future. To this end, the sum of expected future rewards (called the *value*) must be estimated for all possible situations. RL algorithms learn a value function from interaction with the environment and are thus framed as *induc-*

tive machine learning problems. In contrast, classical artificial intelligence methods use *deductive* planning to derive an optimal control from a model of the environment. A closer look on these two approaches reveals complementary strengths a weaknesses: (i) inductive learning can continuously adjust the control by interacting with the environment and allows therefore to correct initial estimation errors. The control policy can only be learned for situations "similar" to those experienced during training, though. (ii) Deductive planning allows to generalize to completely new situations, but must predict these with models. As there is no interaction with the environment, errors in these models can not be rectified later. While deductive planning can work under highly controlled factory-conditions, both methods regularly fail in realistic, uncontrolled environments.

The reason for this failure is twofold: realistic dynamics can rarely be described exact enough for planning, while learning approaches suffer what I call the *curse of insufficient sampling*. Here a high dimensional description of the *state* of the system yields too many situations to sample in realistic time. At the current state-of-the-art, engineers have to choose a proper *representation* of the state such that: (i) the representation restricts variability in all states can be experienced during inductive training, *or* (ii) the state dynamics exhibit almost deterministic behavior during deductive planning.

This thesis aims to increase the agents autonomy by *learning* a near-optimal representations and to overcome the respective weaknesses of inductive and deductive value estimation by *merging* both approaches. In other words,

> I INVESTIGATE THE ROLE OF REPRESENTATIONS THAT GENERALIZE CONTINUOUS VALUES IN AUTONOMOUS REINFORCEMENT LEARNING.

The thesis is structured the following way: I introduce the different approaches to autonomous control in Chapter 2 and analyze how inductive learning and deductive planning generalize values from experiences in continuous state spaces to unseen situations in Chapter 3. My analysis reveals that the inductive approach depends crucially on the *metric* between states, which in turn depends on the state representation. In Chapter 4 I summarize the literature of representation learning. I also prove analytically that under some assumptions the unsupervised learning technique *slow feature analysis* approximates optimal representations for inductive value estimation. The deductive approach, on the other hand, generalizes to unseen situations using predictions of a *transition model*. A truly autonomous agent must learn this model inductively, and in Chapter 5 I show which *structural assumptions* on the representation are necessary to break the curse of insufficient samples. These assumptions allow me to include a *relational description* into the framework, which can generate an adjusted transition model for each configuration of the environment. Chapter 6 finishes the thesis with a summary of the presented material, my conclusions from it, and open questions for future research.

During my PhD studies I published three journals articles about *representation learning*, which can be found in Appendix A, and two papers about factored representations for deductive planning, which can be found in Appendix B. However, this thesis goes far beyond the scope of these publications. Besides an unpublished modification to SFA and countless new propositions and lemmas, the thesis contains also three unpublished algorithms (Algorithms 6, 7 and 9), novel modifications to two existing algorithms (Algorithms 1 and 2) and discusses various extensions of these algorithms for specific purposes.

Chapter 2

Autonomous Control

Intuitively, we call the behavior of an animal, a robot or a power plant "controlled", if it acts *towards* something. While our language allows us to pursue "happiness", "fulfillment" or "justice", an artificially constructed agent may only strive for measurable REWARDS or GOALS. Although these two concepts are often treated separately, goals are in fact special cases of (single) rewards. These are usually set by the system designer with a specific TASK in mind. Defining rewards that elicit the desired behavior requires expert domain knowledge and usually quite extensive tinkering. Methods like IMITATION LEARNING aim therefore to remove the designer by inferring the rewards from demonstrations performed by experts (see Section 2.3.3 for more details). Rewards can thus be seen as equivalent to tasks, as rewards can specify the task and be specified by performing it.

REWARDS, GOALS AND TASKS

Rewards must also be tied to specific advantageous or disadvantageous STATES of the ENVIRONMENT. The agent does not necessarily need to be able to observe these states[1], but it needs to observe the reward. A bee that drinks nectar from a flower (a famous example from Dayan and Abbott, 2005) must not be able to observe *what* it is drinking, only that it contains rewarding sugar. This example also demonstrates the importance of the agent's BODY, which effectively generates the primal reward from sensory observations. The EMBODIED COGNITION hypothesis argues that the body is essential for the computation and development of true autonomous control (see e.g. Anderson, 2003). However, although I often use the example of embodied agents in this thesis, I will not discuss the influence of the body and regard it instead as part of the environment. This yields directly observable rewards, may they be generated by the agent's body, some sensor measurements or an external trainer.

STATES OF THE ENVIRONMENT

A good control will choose ACTIONS that steer the system towards a rewarded goal. However, specifying every interaction with the environment in advance is impossible for all but the most primitive tasks. Any sufficiently complex control therefore includes decisions made AUTONOMOUSLY by the agent. Multiple definitions exist (e.g., for small robots by Floreano and Wood, 2015), but I define *autonomy* here simply as "acting without specific orders". However, the chosen actions are not arbitrary, but based on two different types of logical reasoning:

AUTONOMOUS PLANNING AND LEARNING

[1] Although the *full* state of the environment is certainly *not* observable, in practice the concept of states is used ambiguous and flexible: a state may be the configuration of a machine, or the location and waveform of all particles in the universe. The decision is an integral part of the system's design.

- PLANNING *deduces* favorable decisions from given KNOWLEDGE using RULES.
- LEARNING *induces* favorable decisions from previous experiences.

These approaches differ fundamentally in their autonomy and flexibility. Planning allows predictions for any situation, but can not *correct* for flawed rules or faulty knowledge. Learning can correct initial misconceptions by interaction with the environment, but can only deal with situations that have been *experienced* before. Ideal control would learn rules and knowledge inductively, to flexibly plan actions and reactions deductively. Of course, this is only possible in highly structured environments and requires the agent to recognize the patterns[2] that underly this structure. These are enormous challenges and rarely discussed in literature. In Chapter 5, I mention some noteworthy approaches and introduce my own work on deductive control with learned rules.

I start this chapter in Section 2.1 by reviewing the deductive planning approach of classical ARTIFICIAL INTELLIGENCE (AI, Russell and Norvig, 2009), with a focus on how the state has to be represented. To highlight the alternative inductive approach, I continue in Section 2.2 with REINFORCEMENT LEARNING (RL, Sutton and Barto, 1998). This child of *psychology* and *machine learning* learns control from interaction with non-deterministic environments. Section 2.3 finishes with a discussion of some of the most critical challenges to autonomous control.

2.1 Planning in artificial intelligence

Artificial intelligence was originally conceived as a replica of human intelligence. Early pioneers of the field followed ancient philosophical traditions of language analysis and logical deduction. In retrospect, this approach was too narrowly focused on what we can reflect upon ourselves. Observations from actual human behavior, which had been extensively studied in the field of psychology, were mostly ignored. Nonetheless, classical AI led to some truly marvelous machines like the *chess world-champion* of 1996 DEEP BLUE (Hsu, 2002), or the 2011 *Jeopardy!* winning WATSON (Ferrucci, 2012). These examples are highly specialized EXPERT SYSTEMS, though, which work exclusively in one domain, that must be specified by a system of symbols and deduction rules. The formal approach did not lead to a more versatile "human-like" intelligence, at least not so far. Some schools of thought, like the aforementioned embodied cognition hypothesis, claim that such an intelligence can not emerge without direct interaction with the real world. As a matter of fact, seemingly "simple" robotic tasks, like *walking* and *grasping*, have proven surprisingly hard to solve. Despite decades of research, we still lack sufficiently precise and versatile control methods for numerous potential applications. In this thesis, I do not want to review the formal approach to AI (see e.g. the STRIPS planner by Fikes and Nilsson, 1971). Instead I focus on continuous or discrete states x, which could (at least in principle) describe the real-world state of a machine, an animal or any controllable system that can be formalized as a *Markov decision process*.

EXPERT SYSTEMS

[2] It is worth noticing that the third type of logical reasoning, ABDUCTION, promises to be of great use in learning these patterns (e.g. by abducting observed transitions to model sparse system dynamics, Pasula et al., 2007). See the discussion in Section 5.5.3 for more details.

2.1.1 Markov decision processes

The discussed states have to form a COMPACT SET[3] $x \in \mathcal{X}$, which is called the STATE SPACE, irrespective of whether it is a bounded vector space (e.g. $\mathcal{X} := [0,1]^d \subset \mathbb{R}^d$), a manifold embedded therein (e.g. $\mathcal{X} \subset \mathbb{R}^d$), a loose collection of discrete states (e.g. $\mathcal{X} := \{x_1, \dots, x_n\}$), or a combination of the above. Similarly, I define the ACTION SPACE $\mathcal{A}$ under the assumption that all actions can be executed in all states. In this chapter I will usually assume that the state and action spaces are FINITE, which I extend to (potentially) continuous sets in Chapter 3. Executing these actions $a \in \mathcal{A}$ at time t causes STATE TRANSITIONS $x(t) \overset{a(t)}{\mapsto} x(t+dt)$. For deterministic transitions and continuous time (i.e., in the limit of $dt \to 0$), the problem of optimal control has been examined for the last 300 years and given rise to the field of CONTROL THEORY (Sussmann and Willems, 1997). In the 1940's, Richard Bellman extended this theory to non-deterministic transitions with discrete time steps (i.e., $dt \gg 0$), called MARKOV DECISION PROCESSES (MDP, Bellman, 1957). In the following I will ignore direct solutions to continuous time problems (e.g. differential equations) and approximate these cases as MDP with very small discrete time steps dt.

STATE AND ACTION SPACE

Formally, a MDP is a tuple $(\mathcal{X}, \mathcal{A}, P, R)$ containing the states $\mathcal{X}$ and actions $\mathcal{A}$. The state transitions are governed by a probabilistic TRANSITION MODEL, which is defined[4] as $P : \mathcal{X} \times \mathcal{A} \times \mathcal{B}(\mathcal{X}) \to [0,1], \int P(dx'|x,a) = 1, \forall x \in \mathcal{X}, \forall a \in \mathcal{A}$. Rewards may depend on either part of an observed transition: on the start-state $x \in \mathcal{X}$, the action $a \in \mathcal{A}$ or the end-state $x' \in \mathcal{X}$. To model stochastic rewards as well, I define the REWARD DISTRIBUTION $R : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \times \mathcal{B}(\mathbb{R}) \to [0,1]$, that is, $R(B|x,a,x')$ denotes the probability that the observed reward is within Borel set $B \in \mathcal{B}(\mathbb{R})$. Classical planning algorithms usually assume deterministic transition and reward models, but all real-world applications contain *some* amount of noise. However, this section will touch the subject only superficially. Non-deterministic transitions are discussed in depth within the context of reinforcement learning (Section 2.2).

TRANSITION AND REWARD MODEL

2.1.2 Trajectory planning

Given a (potentially unknown) start-state $x_0 \in \mathcal{X}$, all future developments form a DECISION TREE. Each node of this tree represents the current state, where each available action $a \in \mathcal{A}$ is executed. The successive states resulting from these transitions are added as children of the node. The new child will be marked a leaf and not further evaluated if either a certain SEARCH DEPTH or a goal state is reached. This way, each path from the tree's root to its leafs is a sequence of actions, which I call a PLAN. The simplest way to populate this tree is to execute each plan, starting at x_0. This usually requires a simulator. If the environment is non-deterministic, each plan can be executed multiple times to estimate a statistical average, which is called MONTE CARLO SAMPLING (MC). MC does not require knowledge of the

DECISION TREES

[3] Compactness implies that an infinite ergodic Markov chain comes arbitrary close to every element.

[4] The notation $[a,b]$ refers here to the set $\{y \in \mathbb{R} \,|\, a \leq y \leq b\}$, and $\mathcal{B}(\mathcal{X})$ refers to the collection of all *Borel sets* of the METRIC space $\mathcal{X}$. To cast the setting into the mathematical framework of *function analysis*, I need to assume the existence of a *metric* on $\mathcal{X}$ and $\mathcal{A}$. For example, vector spaces $\mathbb{R}^d$ may use the *Euclidean metric*, manifolds therein may use either the metric of their embedding space or a *diffusion metric* (Coifman et al., 2005; Böhmer et al., 2013), and discrete $\mathcal{X}$ can use the *discrete metric* with distance function $d(x,x') = 0$, if $x = x'$, or otherwise $d(x,x') = 1$.

state, if one can either reset the simulator or find some other way to return to x_0. This is an advantage if the state of the world $x \in \mathcal{X}$ is high dimensional or hidden.

OPTIMALITY CRITERIA

An OPTIMAL CONTROL selects the *best* plan. There are many ways to define "best". Examples are the shortest *path to a goal*, the highest *sum of rewards*, the lowest *experienced punishment*, a *regular intake* of life-sustaining fuel or any combination thereof. However, one measure will play a special role in this thesis: I define the VALUE $V(\cdot)$ as the *expected discounted sum of future rewards* r_t, that is,

$$V(x_0, a_0, a_1, \ldots, a_{T-1}) \quad := \quad \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t \, r_t \;\middle|\; \begin{array}{c} x_{t+1} \sim P(\cdot|x_t, a_t) \\ r_t \sim R(\cdot|x_t, a_t, x_{t+1}) \end{array}\right], \tag{2.1}$$

DISCOUNT

where $T \in \mathbb{N}$ is the search depth, also called PLANNING HORIZON, and $\gamma \in [0, 1]$ is the DISCOUNT FACTOR, representing how shortsighted[5] the agent is. Most common optimality criteria can be modeled this way, using different discount and reward schemes[6]. Most importantly, for $\gamma < 1$ the value of any infinite plan is *bounded*[7]. This is not directly relevant for decision trees, but will become essential for reinforcement learning in Section 2.2.

BRANCHING FACTOR

The number of plans (and nodes) in a decision tree grows exponential with the planning horizon, that is, a decision tree with planning horizon T and $|\mathcal{A}|$ possible actions at each node (called BRANCHING FACTOR) contains $|\mathcal{A}|^T$ plans. This becomes even more troublesome in probabilistic environments, where each possible outcome of a transition has to be simulated as an independent branch to calculate the average value for a decision. Simulating and evaluating each plan can thus become infeasible for many problems. To achieve grandmaster performance, advanced programs like *deep blue* rely on two shortcuts to avoid calculating every plan:

- *Heuristic value:* define an upper bound on the value of any future plan.
- *Evaluation order:* the most promising plans are simulated first.

MCTS

For example, the board game *Go* has an enormous branching factor, which prevents deep trees. MONTE CARLO TREE SEARCH (MCTS, Coulom, 2006) explores all possible plans randomly, but prefers to sample branches that already led to favorable outcomes in the past. The algorithm can be stopped any time to return the best plan sampled so far. Today MCTS plays Go at the *dan-level* (Lee et al., 2009) and is also implemented in the real-time computer strategy games like TOTAL WAR: ROME II (Champandard, 2014).

A*-SEARCH

One can combine both tricks mentioned above to derive the A*-ALGORITHM for *informed search*. This algorithm maintains a *queue* of unresolved nodes, ordered according to their *total estimated value*, which is composed of the value of the plan so far plus a heuristic that estimates an upper bound on the future value. In each iteration the leading node is removed and all its children sorted into the queue. If a solution with a value larger than the leading node is found, no better plans

---

[5] Small γ will select shortsighted plans, which prefer immediate rewards to those collected in the far future. For γ close to one, on the other hand, the control will choose farsighted plans with the largest *mean reward*. The exponential discount γ^t has multiple interpretations in various fields of application. For example, discount in finance models foregone interest by not investing in secure stocks at an interest rate of $1 - \gamma$ and deterministic planning in stochastic domains models the error of each prediction as proportional to $1 - \gamma$.

[6] For example, setting $\gamma = 1$ yields the *sum of reward*, which becomes the *path to goal* criteria with a constant negative reward for all actions until the goal is reached. Non-linear criteria, on the other hand, need to be modeled differently (e.g. lowest punishment as *risk*, Shen et al., 2013).

[7] As long as the reward is bounded itself, i.e. $|r_t| < \infty, \forall t \in \mathbb{N} \;\Rightarrow\; \lim_{T\to\infty} |V(x_0, a_0, \ldots)| < \infty$.

can exist and the algorithm finishes. A* is provably the optimal informed search algorithm (Russell and Norvig, 2009), but requires access to the simulator's state and scales badly with the branching factor. For example, probabilistic environments usually have a huge branching factor, as all possible outcomes need to be considered. The branching factor of *chess*, on the other hand, can be heuristically reduced[8] to a handful moves. Developing an agent that plays chess essentially[9] boils down to defining a good heuristic value function and using a fast computer. Other impressive applications of the A*-algorithm are an autonomous agent that plays the classical video game SUPER MARIO (Karakovskiy and Togelius, 2012) and two-dimensional path-finding in virtually any computer game (Millington and Funge, 2009).

2.1.3 Reactive planning

Applying both MCTS and A* to real-world control problems is problematic, as simulations of reality are always flawed and occurring prediction errors accumulate over time. The easiest solution would be to stop the execution at each unexpected outcome to compute a new plan from that start-position. However, planning is a time-consuming process and in real-time environments it is paramount to compute as much in advance as possible. One would thus like to learn a control, that can *react* to unexpected events by *adjusting* the plan. This is of particular importance in probabilistic environments, where each execution of a plan can result in a different trajectory. In principle, the full decision tree would yield such a control, but is usually too large to be stored in memory. However, much of this information is redundant, as the same situation may be reached with multiple plans or again and again at various times. Instead of keeping track of the whole tree with all its nodes, one can store only the *best* action for each reachable *state*, called a POLICY. Note that this *unfolds* the control from all plans in *time* to all plans in *space*. Formally, I define a stationary[10], stochastic policy π as a conditional distribution over actions given a state, that is, $\pi : \mathcal{X} \times \mathcal{B}(\mathcal{A}) \to [0, 1]$, $\int \pi(da|x) = 1, \forall x \in \mathcal{X}$.

POLICY

The simplest version of a policy is called a FEEDBACK CONTROLLER. Here one first plans a deterministic state-trajectory, often with a grossly simplified transition model of the agent. During execution, the controller simply chooses an action *towards* the next state in the trajectory. This decision requires only a *local* transition model of the agent and its immediate neighborhood. For example, a robot's route is often deterministically pre-planned, and the controller only considers the robot's joint-dynamics to move in the right direction. The involved actions can be either modeled or learned before planning begins (see e.g. Cully and Mouret, 2015, for a learned controller). However, the behavior of a feedback controller relies heavily on the state-trajectory that it follows. As these are often planned without considera-

FEEDBACK CONTROLLER

[8] Often one can *ignore* entire branches that can not influence the decision, called ALPHA-BETA PRUNING. Moving forward and backward, for example, does not influence to the game (except waisting moves) and the entire branch can thus be pruned.

[9] Chess is played against an adversary and A* must accommodate this. A MINIMAX TREE chooses the action with the maximum value for the agent, and with the minimum value for the adversary. For more information on multi-agent systems see Section 2.3.2.

[10] Finite horizon tasks (like selling stocks within a certain time limit) may require a non-stationary policy, as the best action in a state can depend on the time left to act. In this case one can simply *augment* the state space by the time to make the optimal policy stationary. Infinite horizon tasks, on the other hand, have always stationary optimal policies (see Section 2.2 for details).

tion of the executing controller, which does not plan ahead by itself, this can lead to erratic behavior. To avoid collisions, for example, feedback controllers for robot navigation normally refuse to take actions that would approach any obstacle too closely. A carelessly planned trajectory can easily lead to a situation, in which this safeguard catches the robot in an endless loop, alternating between two states (for more details on oscillating behavior see Section 3.4.1). Furthermore, robots can be led completely astray by moving obstacles, as at each step the direction of evasion is computed without consideration of the obstacle's movements.

The problem can be partially solved if the controller is allowed to plan more than one step ahead. For example, Kim et al. (2014) demonstrated a robot arm that caught flying objects like tennis rackets out of mid-air. Their approach plans near-optimal movements for each (task-relevant) state ahead of time, that is, stores the best actions for a wide range of possible object/hand positions and dynamics. However, to plan ahead for all possible (or at least probable) states in realistic environments is hardly feasible. The state of the physical world is ambiguous, not fully observable and, however the interpretation, mind-blowingly huge! For the purpose of control we only need to consider states that are relevant to the task, but the definition of the "true" state of the world is essentially a philosophical question.

2.1.4 Stimuli, memories and state heuristics

From a biological point of view the true state is irrelevant, as no animal can ever hope to *observe* it completely. Neurobiological control must thus be defined in terms of sensor OBSERVATIONS of the environment. The influential *behaviorism* movement in psychology introduced for this purpose the concept of STIMULI: reproducible sensor patterns that invoke reproducible reactions in an animal. What *can* be called a stimuli depends on the animal, that is, on its preexisting ability to sense and recognize the physical presence of the stimulus. In practice, most stimuli are either as basic as possible, like a flashing light or a pronounced sound, or are based on neurological insights, for example moving dots or oriented edges based on specialized cells found in the visual cortex of mammals. Similar approaches for artificial agents are possible, but require considerable structural insight into the often high-dimensional sensor input (e.g. visual SIFT-features, Lowe, 1999). It is also not clear how to select the task-relevant stimuli, which is rarely discussed in psychology. However, my strongest argument *against* behaviorism for autonomous control is the "out of sight, out of mind" problem. Consider the simple task of running away from a tiger. As soon as the agent turns its back on the tiger (the stimulus), a control policy based exclusively on stimuli will no longer know how to react. The best control would always run backwards to keep an eye on the tiger. As this leaves the agent vulnerable to collisions, pure behaviorism is clearly a bad evolutionary choice.

OBSERVATIONS

The agent may survive by also reacting to "internal states", that is, by having a SHORT-TERM MEMORY. In the above example it is sufficient to remember that there *is* a tiger behind the agent, as *any* direction without a tiger is a good direction. In general, however, control policies in PARTIALLY OBSERVABLE MARKOV DECISION PROCESSES (POMDP, Kaelbling et al., 1998) depend on the entire *history* of observations. POMDP assume knowledge of the transition model in the unobservable state space $\mathcal{X}$ and the probability for observations given such an underlying state $x \in \mathcal{X}$. The agent's *belief* of the current state is expressed as a probability distribution over $\mathcal{X}$ and can be updated after each observation using Bayes' theorem. This

POMDP

distribution is called the BELIEF STATE. POMDP can be solved like a normal MDP (see Section 2.2) with all distributions over $\mathcal{X}$ as the state space. Fleeing a tiger, for example, requires only one binary variable to indicate the dangerous presence. If the tiger is observed, the agent will strongly belief in its presence, turn around and run away. After some time is passed, this belief has been reduced enough for the agent to turn around and make sure the tiger is really gone. Due to their Bayesian nature, POMDP policies always balance costs and benefits of belief verification.

Although POMDP can be efficiently solved for small state spaces $\mathcal{X}$, the presented approach requires knowledge of all probabilities involved. Estimating these probabilities has proven to be notoriously hard in practice (Shatkay and Kaelbling, 2002): fleeing too long wastes precious energy, but assuring oneself too early can have dire consequences. Moreover, autonomously constructing the underlying state space $\mathcal{X}$ from experience alone seems almost impossible due to the huge number of possible interpretations for the same history of observations. Instead of defining an underlying state, PREDICTIVE STATE REPRESENTATIONS (PSR, Littman et al., 2001; Wingate, 2012) track the statistics of *future histories* of observations conditioned on *past histories*. In their framework the sufficient statistics of this distribution is finite dimensional and can be estimated from interaction with the environment. One can see PSR as an example for *learning* internal states (i.e. short-term memories) by *compressing* histories. The compressed statistics of each history provide a "belief" over the unknown non-observable state, which allows to solve the POMDP. PSR

As we will see for the simpler observable case in Chapter 4, estimating a latent state space can become infeasible when that space is too large. From an engineering point of view, there is actually no reason to consider the complete state space $\mathcal{X}$. A TASK-RELEVANT SUBSPACE of $\mathcal{X}$ would allow to solve the given task much more effective. For example, the state in robotics is often described as the *degrees of freedom* of a robot, called the CONFIGURATION SPACE (Russell and Norvig, 2009). External obstacles enter this state description by restricting the accessible "free" region of this space. Normally an environment composed of many objects would require exponentially many states due to the combinatorics of all possible arrangements (see FACTORED STATE SPACES in Section 2.3.1). This can become infeasible even for a moderate amount of objects. For many tasks, however, the robot can assume most of these to be *irrelevant*. If the robot avoids touching them, the task-relevant subspace is only composed of the free region and the state of the relevant objects. Adding stationary obstacles effectively *reduces* the size of the subspace. TASK-RELEVANT SUBSPACE

Once a task-relevant state space is defined by the engineer, sensors are carefully constructed to observe this heuristic state as flawless as possible from sensor readings. Temporarily hidden features (stimuli) can be tracked using *Kalman filters* (Kalman, 1960). Measurements and estimates depend strongly on the correct calibration of sensors and identification of obstacles' geometry. In practice, highly specialized algorithms perform often reoccurring tasks like SIMULTANEOUS LOCALIZATION AND MAPPING (SLAM, Smith et al., 1990; Davison, 2003) of the environment. Individual robot dynamics, however, have to be as simple as possible. Robots with many degrees of freedom are often discarded in favor of more manageable models. A noticeable exception are humanoid robots, which have considerable advantages in environments shaped for humans. To use classical planning, traditional robotics also restricts itself to actuators and materials that exhibit predictable deterministic transitions. This excludes flexible materials and muscle-like actuators, which are preferred by biology. Soft materials have only recently found their way into robotics and opened a treasure SLAM

trove of new body shapes (Rus and Tolley, 2015). Controlling these under-actuated robots with large degrees of freedom is still a generally unsolved problem, though.

2.2 Reinforcement learning

While early artificial intelligence was aiming for the apparent *deductive* capabilities of the human mind, neurobiology provided the first insights in its *inductive* capabilities to learn. The early models from experimental observations by McCulloch and Pitts (1943) already introduced the basic architecture of ARTIFICIAL NEURAL NETWORKS (ANN, see e.g. Haykin, 1998). Soon thereafter the first algorithms to train single artificial neurons with UNSUPERVISED LEARNING (*Hebb's rule*, Hebb, 1949) and SUPERVISED LEARNING (the *perceptron*, Rosenblatt, 1962) emerged. However, due to a tremendous amount of shortsightedness in academic AI circles and probably the absence of fast computers to exhaustively test the approach, research on neural networks was shortly thereafter abandoned for two decades.

MACHINE LEARNING

The field that is known today as MACHINE LEARNING (ML, Bishop, 2006, see also the more detailed description in Chapter 3) has been rekindled by the *connectionst* movement through the reinvention of the *back-propagation* algorithm (Rumelhart et al., 1986). ML aims to estimate the probability distribution of data from samples. If the training samples are labeled, the unknown label of a test sample can be predicted by conditioning the distribution on the observable quantities. Distributions can be estimated parameter free, e.g. with *Gaussian processes* (GP, Rasmussen and Williams, 2006), but it is often advantageous to fit the parameters of a flexible model class like ANN to the training data. Here the optimization is more complicated, but can borrow techniques from *function analysis* and often obtain more compact models of the data. An example for this trade-off can be found in my own work (Böhmer and Obermayer, 2015, Appendix B.2).

REINFORCEMENT LEARNING

During the resurgence of machine learning in the 1980's, statistical estimation techniques were applied to the control problem defined by a MDP. Due to the connectionists roots in psychology, the approach was named REINFORCEMENT LEARNING (RL, Sutton and Barto, 1998). RL combines psychological terminology, mathematical control theory and machine learning algorithms to inductively *learn* a control policy through interaction, with almost no prior knowledge of the environment.

2.2.1 Psychological roots

Cognitive psychology has been developed out of, and in opposition to, behaviorism. This field has been significantly shaped by the two pioneers Ivan Pavlov (1849–1936) and B. F. Skinner (1904–1990). Their two approaches to CONDITIONING are so general, that one can recognize them to this day in RL theory.

CONDITIONING

Pavlov noticed that the dogs he was working with started to salivate, not only when they were fed, but also when his colleague who fed them entered the room. He devised the CLASSICAL CONDITIONING experiment: a neutral stimulus (called *conditioned* or CS) is paired with a rewarding stimulus (called *unconditioned* or US). Before conditioning, only the US produced a reliable response, like the dogs salivating in the presence of food. After some time the dogs seemed to associate the previously neutral stimulus CS (e.g. a sound) with the US and started to salivate even if the CS was not followed by food. Although classical conditioning establishes a basic framework of learning, it is restricted to reflexive responses. To investigate

how this relates to changes in the voluntary *behavior* of the animal, Skinner popularized OPERANT CONDITIONING. Rather than pairing the reward with a stimulus, he rewarded certain *actions* that the animal performed voluntarily. After a few rewards the animal's behavior changed measurably.

Both approaches share some properties, like *blocking*, *inhibition* and *extinction*. These effects are not intuitive, but can be reproduced by RESCORLA-WAGNER MODELS (Rescorla and Wagner, 1972). To extend the predictions of these models to sequential decision making, the first RL algorithm, TEMPORAL DIFFERENCE LEARNING (TD, Barto and Sutton, 1982; Sutton, 1988), was born. It can be shown that TD models have the ability to replicate observed behavior in both classical (Sutton and Barto, 1990) and operant conditioning tasks (Barto et al., 1990). Moreover, the TD-ERROR, used to update the value estimation as in Equation 2.9 on Page 13, has been found to predict the response of *dopamine* neurons in mammal brains (Schultz et al., 1997). This finding has been replicated using other techniques like EEG and fMRI, and created a broad scientific consensus that mammal brains implement some kind of TD model to learn behavior on a subconscious level.

TD-MODELS

During my time as a PhD student I have also used my insights into reinforcement learning to help develop models for behavior analysis. I co-authored two journal articles that construct modified TD models to explain seemingly irrational human behavior in learning tasks (Houillon et al., 2013; Tobia et al., 2014). However, these works are not the primary focus of my studies and thus not included in this thesis.

2.2.2 Value iteration

I have already introduced the underlying concepts required for a formal description of reinforcement learning: the environment (including the agents body) is described by a MDP $(\mathcal{X}, \mathcal{A}, P, R)$ (Section 2.1.1 on Page 5) and an agent's control is defined by a policy π (Section 2.1.3 on Page 7).

To measure the success of any given policy, one can calculate the average VALUE (Equation 2.1) when following this policy from some start state $x_0 \in \mathcal{X}$. This value has an *infinite planning horizon*, and can be expressed recursively in a closed analytical form based on the probabilities defined by the MDP and policy:

VALUE

$$\begin{aligned} V^\pi(x_0) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \;\middle|\; \begin{array}{rcl} a_t & \sim & \pi(\cdot|x_t) \\ x_{t+1} & \sim & P(\cdot|x_t, a_t) \\ r_t & \sim & R(\cdot|x_t, a_t, x_{t+1}) \end{array} \right], \qquad \forall x_0 \in \mathcal{X} \qquad (2.2) \\ &= \int_{\mathcal{A}} \pi(da|x_0) \int_{\mathcal{X}} P(dx'|x_0, a) \Big(\int_{\mathbb{R}} R(dr|x_0, a, x')\, r + \gamma V^\pi(x') \Big) . \end{aligned}$$

One can express this recursive equation in terms of FUNCTION ANALYSIS, that is, as a function from a LEBESGUE SPACE $L^p(\mathcal{X}, \xi)$. These *Banach spaces* of real-valued functions $f : \mathcal{X} \to \mathbb{R}$ are defined by their *Lebesgue measure* $\xi : \mathcal{B}(\mathcal{X}) \to [0, \infty)$, e.g., a *probability measure* with $\xi : \mathcal{B}(\mathcal{X}) \to [0, 1]$ and $\int \xi(dx) = 1$, and their L_p-norm:

L^p SPACES

$$\|f\|_p = \Big(\int_{\mathcal{X}} \xi(dx)\, |f(x)|^p \Big)^{\frac{1}{p}}, \qquad p \geq 1 . \qquad (2.3)$$

In this chapter I will only consider[11] the $L^\infty(\mathcal{X})$ space with SUPREMUM NORM[12]:

$$\|f\|_\infty \quad = \quad \lim_{p\to\infty} \|f\|_p \quad = \quad \sup_{x\in\mathcal{X}} |f(x)| \,. \tag{2.4}$$

$L^\infty(\mathcal{X})$ contains all functions that are uniformly *bounded*. This excludes some mathematical tricks like setting the reward of undesirable or unobtainable states to $-\infty$, but includes all functions that can be expressed by a computer. For example, as the value is defined as the *average* sum of discounted rewards, the influence of reward in Equation 2.2 can be summarized as MEAN REWARD FUNCTION:

$$r(x,a) \quad = \quad \int_{\mathcal{X}} P(dx'|x,a) \int_{\mathbb{R}} R(dr|x,a,x')\, r \quad \in \quad L^\infty(\mathcal{X}\times\mathcal{A}) \,. \tag{2.5}$$

Under the reasonable assumption that all rewards are *bounded*, this function is always in $L^\infty(\mathcal{X}\times\mathcal{A})$. For transition model P and policy π, on the other hand, one must introduce LINEAR OPERATORS[13], which are mappings from functions to functions:

$$\hat{\mathrm{P}}[f](x,a) \quad = \quad \int_{\mathcal{X}} P(dx'|x,a)\, f(x') \quad \in \quad L^\infty(\mathcal{X}\times\mathcal{A}) \,, \quad \forall f \in L^\infty(\mathcal{X}) \,, \tag{2.6}$$

$$\hat{\Gamma}_\pi[g](x) \quad = \quad \int_{\mathcal{A}} \pi(da|x)\, g(x,a) \quad \in \quad L^\infty(\mathcal{X}) \,, \quad \forall g \in L^\infty(\mathcal{X}\times\mathcal{A}) \,. \tag{2.7}$$

With these concepts, we can reformulate the value function in Equation 2.2 as the fix-point v^π of the so called BELLMAN OPERATOR for VALUE ITERATION:

VALUE ITERATION

$$v^\pi \quad = \quad \hat{\Gamma}_\pi\big[r + \gamma\, \hat{\mathrm{P}}[v^\pi]\big] \quad =: \quad \hat{\Gamma}_\pi\big[\hat{\mathrm{B}}[v^\pi]\big] \quad \in \quad L^\infty(\mathcal{X}) \,. \tag{2.8}$$

The operator is a contraction mapping in the supremum norm, that is, any two functions become point-wise more *similar* after the application of the Bellman operator.

Proposition 2.1 (see e.g. Bertsekas, 2007) *The value iteration operator* $\hat{\Gamma}_\pi\big[\hat{\mathrm{B}}[v]\big] = \hat{\Gamma}_\pi[r] + \gamma\, \hat{\Gamma}_\pi\big[\hat{\mathrm{P}}^\pi[v]\big]$, $\forall v \in L^\infty(\mathcal{X})$, *is for* $\gamma < 1$ *a contraction mapping under the supremum norm.*
Proof: can be found in Appendix C.2 on Page 234. □

This implies that repeated application the update rule $v \leftarrow \hat{\Gamma}_\pi\big[\hat{\mathrm{B}}[v]\big]$ will in the limit converge to the unique fix-point v^π, irrespective of the initial function $v \in L^\infty(\mathcal{X})$. At least theoretically it is also possible to determine v^π with an INVERSE OPERATOR:

Proposition 2.2 (see e.g. Bertsekas, 2007) *The fix point of value iteration operator* $\hat{\Gamma}_\pi[\hat{\mathrm{B}}]$ *in* $L^\infty(\mathcal{X})$ *is*

$$v^\pi \quad = \quad \hat{\Gamma}_\pi\big[\hat{\mathrm{B}}[v^\pi]\big] \quad = \quad \Big(\hat{\mathrm{I}} - \gamma\hat{\Gamma}_\pi[\hat{\mathrm{P}}]\Big)^{-1}[\hat{\Gamma}_\pi[r]] \quad \in \quad L^\infty(\mathcal{X}) \,,$$

where $(\hat{\mathrm{A}})^{-1}$ *denotes the inverse operator of* $\hat{\mathrm{A}} : L^\infty(\mathcal{X}) \to L^\infty(\mathcal{X})$ *with* $\hat{\mathrm{A}}[(\hat{\mathrm{A}})^{-1}] = (\hat{\mathrm{A}})^{-1}[\hat{\mathrm{A}}] = \hat{\mathrm{I}}$, *and* $\hat{\mathrm{I}}$ *denotes the identity operator* $\hat{\mathrm{I}}[f] = f, \forall f \in L^\infty(\mathcal{X})$.
Proof: can be found in Appendix C.2 on Page 234. □

However, it is in general not possible to exactly compute the inverse operator of $(\hat{\mathrm{I}} - \gamma\hat{\Gamma}_\pi[\hat{\mathrm{P}}])$ in $L^\infty(\mathcal{X})$ for continuous $\mathcal{X}$. For finite state spaces, on the other hand, the fix-point can be computed by a matrix inverse, as discussed in Section 3.3.1.

[11] In Chapter 3 I will also introduce the $L^2(\mathcal{X},\xi)$ spaces, which contain all functions from $L^\infty(\mathcal{X})$, but are *Hilbert spaces* and in some ways better suited for function approximation.

[12] The second equality holds only if ξ is absolute continuous w.r.t. a uniform measure ϑ, i.e. $\xi \ll \vartheta$.

[13] Any conditional measure $A : \mathcal{X}\times\mathcal{B}(\mathcal{Y}) \to [0,\infty)$ induces a linear operator $\hat{\mathrm{A}} : L^\infty(\mathcal{Y}) \to L^\infty(\mathcal{X})$, $\hat{\mathrm{A}}[f](x) = \int A(dy|x) f(y), \forall x \in \mathcal{X}, \forall f \in L^\infty(\mathcal{Y})$, which here bears the same name with a hat.

2.2.3 Online and batch estimation

Every linear operator based on probability measures can be estimated by computing the average over training samples drawn from these distributions. A trajectory controlled by policy π, called a MARKOV CHAIN, provides the samples to do just that by sampling all actions $a_t \sim \pi(\cdot|x_t)$, successive states $x_{t+1} \sim P(\cdot|x_t, a_t)$ and rewards $r_t \sim R(\cdot|x_t, a_t, x_{t+1})$. One can estimate the effects of the value-iteration operator by computing an average over this set with the update rule

MARKOV CHAINS

$$v(x_t) \quad \leftarrow \quad v(x_t) \; + \; \alpha_t \big(\underbrace{r_t + \gamma v(x_{t+1}) - v(x_t)}_{\text{TD-error}} \big) \,. \tag{2.9}$$

Here $\alpha_t \in (0, 1]$ is the LEARNING RATE, discussed in detail below, followed by the TEMPORAL DIFFERENCE ERROR (eponymous to temporal difference learning, Sutton and Barto, 1998, see Section 2.2.1).

Note that Proposition 2.1 states that the Bellman operator is a *point-wise* contraction mapping. Value iteration therefore also converges if only one state is updated at a time. Applying Equation 2.9 repeatedly to the entire Markov chain will therefore estimate the Bellman operator and *simultaneously* perform value iteration. In the limit of an infinite Markov chain that visits each state infinitely often[14], any initial value function $v \in L^\infty(\mathcal{X})$ will converge to (or fluctuate around) the fix point v^π. The value of a given policy can therefore be *learned* inductively by executing it.

The update rule in Equation 2.9 is an example of ONLINE LEARNING. To allow convergence, the learning rate α_t needs to shrink over time (Sutton and Barto, 1998). However, it is often preferred to forgo convergence in favor of adaptivity to potentially non-stationary environments. To give a simplified example, let f_n be the online-estimated average of "target" training samples $\{y_t\}_{t=1}^n$, drawn i.i.d. with

ONLINE LEARNING

[14] This is called an ERGODIC Markov chain. Each state can eventually be reached from each other state with a non-zero probability. I will define the concept more thoroughly in Section 3.1.1, as ergodicity is fundamental for function approximation in $L^2(\mathcal{X}, \xi)$ spaces.

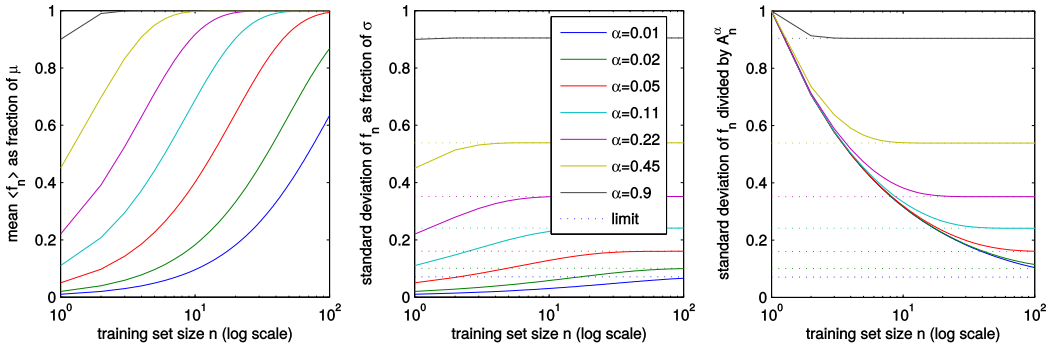


Figure 2.1: Analytical convergence of online estimates with constant learning rate α. In the limit of infinite samples, the estimate will fluctuate around the true mean μ, with standard deviation σ depending on α. The left plot shows the mean online estimate (as fraction of μ) against the number of training samples, which is equivalent to A_n^α (see text). The middle plot shows the corresponding standard deviation. However, if the estimate is divided by A_n^α, the mean is for all n bias-free, while the standard deviation still converges to the same limits (right plot).

mean μ and variance σ^2. Online-learning is equivalent to a *temporal convolution kernel*,

$$f_n \quad = \quad f_{n-1} \; + \; \alpha\,(y_n - f_{n-1}) \quad = \quad \sum_{t=0}^{n-1} \alpha(1-\alpha)^t\, y_{n-t}\,. \tag{2.10}$$

Although the f_n does not converge, it is easy to show that the average over infinite training sets $\mathbb{E}[f_\infty] = \mu$ is *bias-free*, and the variance $\mathbb{E}[f_\infty^2] - \mathbb{E}[f_\infty]^2 = \frac{\alpha\,\sigma^2}{2-\alpha}$ can be controlled by α. This is similar to a *sliding window* average, with smaller α corresponding to larger windows. However, for small training sets a constant learning rate α systematically under-estimates the mean, as for small n the *area* under the convolution kernel $A_n^\alpha := \sum_{t=0}^{n-1} \alpha(1-\alpha)^t = 1-(1-\alpha)^n$ is small, too. Figure 2.1 shows f_n's analytical mean and standard deviation over training sets for multiple α, as a function of training set size n. It is evident that small α strongly reduce the standard deviation, but need significantly more training samples to estimate the mean. To obtain a bias-free estimate for finite training sets, the online learning rate can be adapted over time by $\alpha_t = \frac{\alpha}{A_t^\alpha}$, which reduces variance as well (right plot in Figure 2.1). However, as in the constant case, the online estimate will not converge.

Convergence can only be achieved by considering all training samples equally, that is, in the limit case $\alpha \to 0$. Intuitively, the learning rate α determines the length of the convolution kernel and the lower limit will maximize this length by considering all training samples equally.

Lemma 2.3 *For the bias-free online learning rate* $\alpha_t = \frac{\alpha}{1-(1-\alpha)^t}$ *holds* $\lim_{\alpha\to 0} \alpha_t = \frac{1}{t}$.

Proof: can be found in Appendix C.2 on Page 235. □

Proposition 2.4 *Online learning, with initial value* f_0 *and adaptable learning rate* $\alpha_t = \frac{1}{t+\lambda}$, $\lambda \in [0,\infty)$, *is equivalent to calculating the regularized empirical mean over the training set* $\{y_t\}_{t=1}^n$, *that is,*

$$f_n \quad = \quad f_{n-1} \; + \; \alpha_n(y_n - f_{n-1}) \quad = \quad \tfrac{1}{n+\lambda} \sum_{t=1}^{n} y_t \; + \; \tfrac{\lambda}{n+\lambda} f_0\,.$$

Proof: can be found in Appendix C.2 on Page 235. □

Note that initialization regularizes the estimate for $\lambda > 0$, but the influence will shrink with $\frac{\lambda}{n+\lambda}$ and thus vanish for large n. For $\lambda = 0$, online learning calculates the empirical mean $\frac{1}{n}\sum_{t=1}^n y_t$, which converges in probability to the true mean of the targets' stationary distribution. However, the target $y_t = r_t + \gamma v(x_{t+1})$ in Equation 2.9 is *not* stationary! Although $x_{t+1} \sim \int \pi(da_t|x_t) P(\cdot|x_t, a_t)$ may have a stationary distribution, the value $v(x_{t+1})$ will change during value iteration. BATCH LEARNING avoids this non-stationarity by estimating an new value function $v^{i+1} = \hat{\Gamma}_\pi[\hat{\mathrm{P}}[v^i]] \in L^\infty(\mathcal{X})$ based on an existing (e.g. previous) estimate $v^i \in L^\infty(\mathcal{X})$:

BATCH LEARNING

$$v^{i+1}(x_t) \;\leftarrow\; v^{i+1}(x_t) + \tfrac{1}{\eta_t(x_t)}\Big(r_t + \gamma\, v^i(x_{t+1}) - v^{i+1}(x_t)\Big) \;\approx\; \hat{\Gamma}_\pi\big[\hat{\mathrm{B}}[v^i]\big](x_t)\,, \tag{2.11}$$

where $\eta_t(x)$ denotes how often state $x \in \mathcal{X}$ was present in the hitherto seen training set $\{x_k, r_k\}_{k=1}^t$. Regularization (see Section 3.2.4) can be easily implemented by initializing this function with $\eta_0(x) = \lambda, \forall x \in \mathcal{X}$, and update it with $\eta_t(x_t) = \eta_{t-1}(x_t) + 1$. Note that batch learning only estimates *one* application of the value iteration operator $\hat{\Gamma}_\pi[\hat{\mathrm{B}}]$, while keeping track of three functions (v^i, v^{i+1} and η_t) from $L^\infty(\mathcal{X})$. The fix point v^π can be approached by repeatedly estimating the value function v^{i+1} based on the previous estimate v^i, which requires to see all training

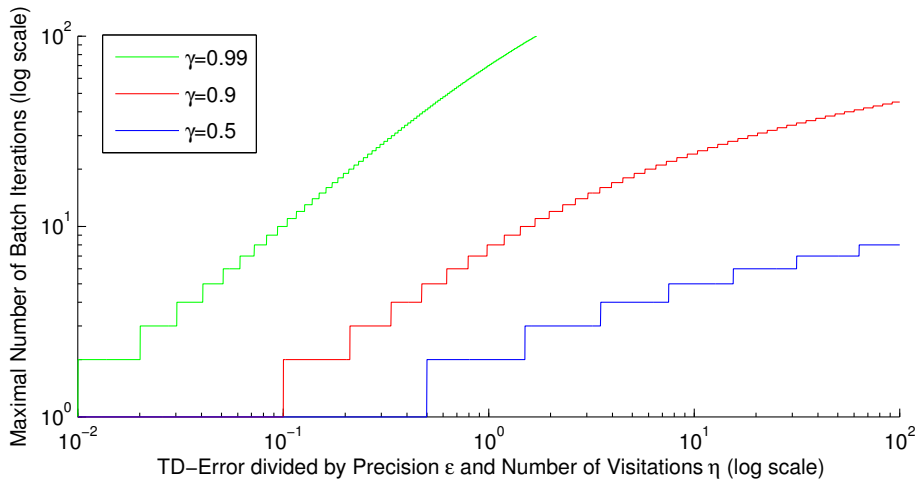


Figure 2.2: Bound on number of iterations until convergence from Prop. 2.5.

samples. The iteration finishes when some convergence criterion, like the function norm of differences being smaller than some small ϵ, $\|v^{i+1} - v^i\|_\infty < \epsilon$, is fulfilled.

Each time a new transition is observed, value iteration has to continue until convergence to the new fix point. Proposition 2.5 bounds the number of iterations necessary, which is visualized in Figure 2.2. Note that the TD-error is divided by the number of visitations $\eta_n(x_n)$ in the current state x_n. Batch value iteration in well known states converges therefore much faster than in newly explored ones.

Proposition 2.5 *Given a converged value function $v^i \in L^\infty(\mathcal{X})$, $\|v^i - v^{i-1}\|_\infty \leq \epsilon$, batch learning (Equation 2.11) converges in at most k value iterations after observing a new transition with TD-error $TD_n := r_n + \gamma v^i(x_{n+1}) - v^i(x_n)$:*

$$k \quad \leq \quad 1 \; - \; \log_\gamma \left(\gamma + \tfrac{1}{\epsilon} \tfrac{|TD_n|}{\eta_n(x_n)} \right) .$$

Proof: can be found in Appendix C.2 on Page 235. □

In comparison, both value estimation methods have their strengths and weaknesses. On the one hand, batch learning requires the storage of all training samples and more computations to approach the fix point v^π. On the other hand, it is convergent and the reuse of training samples requires much less interaction with the environment for a reliable estimate. Online learning is therefore *faster* and requires *less memory*, but batch learning is more *sample efficient*.

2.2.4 Policy improvement

Evaluating each potential policy's value by sampling a Markov chain is not feasible in practice. Fortunately, there exist a greedy method to *improve* the policy, that is, to maximize the value of each state. The value function of a policy provides the average future reward of a state, but does not directly reveal which actions would improve it. One alternative is to estimate the average sum of discounted future reward for each *action* in each state, called the Q-VALUE function: Q-VALUE

$$q^\pi \quad = \quad r + \gamma \hat{\mathrm{P}} [\hat{\Gamma}_\pi [q^\pi]] \quad = \quad \hat{\mathrm{B}} [\hat{\Gamma}_\pi [q^\pi]] \quad = \quad \hat{\mathrm{B}} [v^\pi] \quad \in \quad L^\infty(\mathcal{X} \times \mathcal{A}) . \qquad (2.12)$$

Note that the Q-value is closely related to the value $v^\pi = \hat{\Gamma}_\pi[q^\pi]$. Moreover, $\hat{\mathrm{P}}[\hat{\Gamma}_\pi[\cdot]]$ is a transition operator in $L^\infty(\mathcal{X}\times\mathcal{A})$, as seen by $\iint P(dy|x,a)\,\pi(db|y) = 1, \forall x \in \mathcal{X}$, $\forall a \in \mathcal{A}$. In other words, Q-values are values of this transition operator in "state space" $\mathcal{X}\times\mathcal{A}$, and all statements about values apply to Q-values as well. The Q-value function q^π of a policy π can therefore be estimated from a Markov chain executing that policy. Sampling from the *evaluated* policy π is called ON-POLICY learning. It is necessary for value iteration, as here the successive state x_{t+1} is determined by $\int \pi(da_t|x_t)\,P(\cdot|x_t,a_t)$. For Q-value iteration, on the other hand, this is not the case, allowing OFF-POLICY learning. Here the Markov chain is controlled by a stochastic sampling policy π, but the Q-value evaluates another policy π' in the operator $\hat{\Gamma}_{\pi'}[\cdot]$.

Q-values allow a simple greedy policy improvement: choose at each state the action that maximizes a given Q-value function $q \in L^\infty(\mathcal{X}\times\mathcal{A})$. This is called a GREEDY POLICY. For example, POLICY ITERATION (PI) improves the policy by estimating the next Q-value function q_{i+1} of the greedy policy π'_i, that maximizes the previous Q-value function q_i. Provided with one Markov chain[15], *batch PI* can optimize the policy iteratively by using off-policy Q-value iteration (Lagoudakis and Parr, 2003). In an online-setting, on the other hand, one can choose the maximum Q-value directly, yielding the non-linear GREEDY POLICY OPERATOR

POLICY ITERATION

$$\hat{\Gamma}_*[q](x) \quad = \quad \sup_{a\in\mathcal{A}} q(x,a) \quad \in \quad L^\infty(\mathcal{X})\,, \quad \forall q \in L^\infty(\mathcal{X}\times\mathcal{A})\,. \tag{2.13}$$

Q-LEARNING estimates the fix-point q^* of operator $\hat{\mathrm{B}}[\hat{\Gamma}_*[\cdot]]$ with online-learning (Watkins and Dayan, 1992), that is, applies at time t the update rule

Q-LEARNING

$$q(x_t,a_t) \quad \leftarrow \quad q(x_t,a_t) \;+\; \alpha\Big(r_t + \gamma \sup_{a'\in\mathcal{A}} q(x_{t+1},a') - q(x_t,a_t)\Big)\,. \tag{2.14}$$

Note that although the operator $\hat{\mathrm{B}}[\hat{\Gamma}_*[\cdot]]$ is not linear, it is still a contraction mapping (shown in Proposition 2.6) and Q-learning will thus always converge to (or oscillate around) a unique $q^* \in L^\infty(\mathcal{X}\times\mathcal{A})$.

Proposition 2.6 (see e.g. Watkins and Dayan, 1992) *The Q-learning operator* $\hat{\mathrm{B}}[\hat{\Gamma}_*[q]] = r + \gamma\hat{\mathrm{P}}[\hat{\Gamma}_*[q]]$, $\forall q \in L^\infty(\mathcal{X}\times\mathcal{A})$, *is for* $\gamma < 1$ *a contraction mapping under the supremum norm.*

Proof: can be found in Appendix C.2 on Page 236. □

The greedy policy π^* corresponding to the fix point $q^* \in L^\infty(\mathcal{X})$, chooses the action with the highest value in *each state*. π^* is thus the OPTIMAL POLICY, that is,

OPTIMAL POLICY

$$q^*(x,a) \;\geq\; q^\pi(x,a)\,, \qquad \forall x \in \mathcal{X}\,, \quad \forall a \in \mathcal{A}\,, \qquad \forall \pi : \mathcal{X}\times\mathcal{B}(\mathcal{A}) \to [0,1]\,. \tag{2.15}$$

For more information about the convergence of value and Q-value iteration with greedy policies, the interested reader is referred to Williams and Baird (1993).

2.2.5 Exploration exploitation

Off-policy methods like Q-learning estimate the Q-value function $q^* \in L^\infty(\mathcal{X})$ of the optimal policy π^*, while following another sampling policy π. For confident

[15] The ergodic Markov chain must visit each state-action pair sufficiently often, that is, the sampling policy π must be stochastic and the Markov chain long enough.

estimates, however, the sampling policy must visit all state-action pairs sufficiently often. This unveils a fundamental antagonism of autonomous control called the EXPLORATION EXPLOITATION DILEMMA. On the one hand, an autonomous agent is supposed to fulfill a task and can not spend its entire time exploring the environment. On the other hand, fulfilling the task depends on accurate estimates, which can only be obtained by exploration. At each decision, an agent must therefore choose to either maximize the *known* future reward by exploiting the learned Q-values, or to maximize the *potential* future rewards by exploring new states and actions.

In practice this decision is often made by heuristics. The easiest and most frequently used exploration-exploitation policy is called ϵ-GREEDY (Sutton and Barto, 1998). Here the stochastic sampling policy π decides randomly to either exploit the Q-value by choosing the highest valued action with probability ϵ, or to explore random actions otherwise. Starting with a small ϵ, the agent will explore the environment, but increasing ϵ slowly over time allows the agent to exploit its growing knowledge. This approach is easy to implement and can work quite well in practice. However, ϵ-greedy policies have two shortcomings: (i) an appropriate growing rate of ϵ depends on the the agent's experiences and can thus not be determined before training, and (ii) the exploration chooses actions randomly without any regard of known traps or shortcuts. ϵ-GREEDY

SOFTMAX policies (also called Gibbs or Boltzmann distributions, Sutton and Barto, 1998) also choose actions randomly, but address the second shortcoming by adjusting the probabilities to the Q-values in this state: SOFTMAX

$$\pi(B|x) \quad := \quad \int_B da \, \frac{\exp\left(\beta \, q(x,a)\right)}{\int da' \, \exp\left(\beta \, q(x,a')\right)} \,, \qquad \forall B \in \mathcal{B}(\mathcal{A}) \,, \quad \forall x \in \mathcal{X} \,. \tag{2.16}$$

$\beta \in [0, \infty)$ is called the INVERSE TEMPERATURE and corresponds to the exploration parameter ϵ. The agent starts with $\beta = 0$ and increases β during training. In difference to ϵ-greedy policies, softmax policies also exploit the learned Q-values during exploration. Actions that are known to have disastrous results will have a comparatively low Q-value and will thus be avoided.

Choosing the action w.r.t. the difference of Q-values has some unwanted side-effects, though. In sparsely rewarded tasks with few "goal states", for example, the Q-values of *all* actions drop exponentially with the distance to the next goal. Softmax policies far away from a goal will therefore be much more stochastic, and it is almost impossible to "exploit" Q-values far away from goal states. To achieve uniform exploration everywhere, I often used a heuristic that normalizes the Q-values $q \in L^\infty(\mathcal{X})$ in Equation 2.16 by enforcing *zero mean* and *unit variance*:

$$\bar{q}(x,a) \quad := \quad \frac{q(x,a) - \int db \, q(x,b)}{\sqrt{\int da' \, (q(x,a'))^2 - (\int da' \, q(x,a'))^2}} \,, \qquad \forall x \in \mathcal{X} \,, \quad \forall a \in \mathcal{A} \,. \tag{2.17}$$

To address the other shortcoming of ϵ-greedy, one can initialize the Q-value function with an OPTIMISTIC INITIAL VALUE (Sutton and Barto, 1998): OPTIMISTIC INITIALIZATION

$$q_0(x,a) \quad := \quad \frac{r_{max}}{1-\gamma} \quad \geq \quad q^\pi(x,a) \,, \quad \forall \pi \,, \quad \forall x \in \mathcal{X} \,, \quad \forall a \in \mathcal{A} \,, \tag{2.18}$$

where $r_{max} \geq r(x,a), \forall x \in \mathcal{X}, \forall a \in \mathcal{A}$, is an upper bound on the mean reward function and $\frac{r_{max}}{1-\gamma}$ thus an upper bound for the value of any state. Exploration of a state-action pair will thus *reduce* the corresponding Q-value and thus make unexplored

actions in this state more attractive. Value iteration can also carry this attraction to neighboring states, which yields a straight forward exploration-exploitation trade-off controlled by the magnitude of initial value $\frac{r_{max}}{1-\gamma}$. This technique is often used for online Q-learning. Once initialized, however, an online-learning agent can not *exploit* its knowledge until all state-action pairs have been visited sufficiently often. Batch policy iteration is more flexible, as described below.

E^3

All above approaches are heuristics, that is, do not come with any guarantees of convergence. The EXPLICIT EXPLORATION AND EXPLOITATION (E^3, Kearns and Singh, 2002) algorithm provides such theoretical support for finite state spaces. E^3 explores the state space by "balanced exploration" while building up a set of "known states", which have been visited sufficiently often to estimate a reliable *model* of transitions and rewards. The agent calculates two value functions based on this model: one where transitions that exit the known state set yield the value $v_0 = 0$ and one where they yield $v_0 = \frac{r_{max}}{1-\gamma}$. In known states where the former is larger, the agent will exploit its knowledge of reward in the known set. When the latter is larger, the agent will explore by heading for the nearest transition that leaves the set. The E^3 algorithm has become very popular among theorists, as it guarantees the optimal policy will be found within bounds that are *polynomial* in all relevant parameters. However, estimating the model restricts the algorithm to finite state spaces and is in practice also slow and memory intensive when compared with heuristics.

In comparison, E^3 and optimistic initialization work almost identical: the difference lies in the fast *propagation* of information by the estimated model. Changes in the value induced by exploration are immediately applied to all states when the fix-point of the changed model is determined. As we have seen, this *sample efficiency* is also a hallmark of batch learning. An equivalent IMPLICIT EXPLORATION EXPLOITATION algorithm should in principle be possible, using policy iteration with batch value estimation. Here the function $\eta_n \in L^\infty(\mathcal{X})$ already keeps track how often a state has been visited. The only changes are: if $\eta_n(x)$ is below some threshold for "known states", then the (otherwise greedy) exploration strategy π performs balanced exploration; and if $\eta_n(x_{t+1})$ is below that threshold, then one replaces $v^i(x_{t+1}) = \frac{r_{max}}{1-\gamma}$ in Equation 2.11 during batch value estimation. The resulting Q-value function should exploit if a large enough reward is close enough and otherwise explore by leaving the known set to perform balanced exploration. I never had the time to sufficiently follow this line of thought, though. Theoretical guarantees and empirical evaluation are thus left to future works.

It is one of the great strengths of MDP to allow any TOPOLOGY OF STATES, and general exploration approaches like E^3 support this by visiting every state-action pair equally. However, in practice this is also highly inefficient in METRIC STATE SPACES. For example, an autonomous robot that explores a new room with E^3 must execute *every* available action in *every* spot multiple times to include the positions in the set of "known states". The robot could explore the room much faster by exploiting the room's metric structure and the similar metric effect of actions, that is, by utilizing PRIOR KNOWLEDGE. This situation is similar to the informed exploration of *decision trees* in Section 2.1.2. The A* algorithm used a heuristic that bounded the value from above, to both explore the most promising states first and to stop the exploration once no better path is possible. Optimistic initial value and E^3 already use such a heuristic (for very similar reasons): the exploration value $\frac{r_{max}}{1-\gamma}$ is an upper bound to the potential Q-value of any state-action pair. Although to the best of my knowledge nobody has attempted this yet, it would be straight forward to use other heuristics

PRIOR KNOWLEDGE

in E^3. For example, if there is one known "goal-state" $x_g \in \mathcal{X}$, E^3 could set the value of a transition leaving the known set in state x to $v_0(x) = \gamma^{\|x-x_g\|}\frac{r_{max}}{1-\gamma}$, using some metric $\|\cdot\|$ in $\mathcal{X}$. As in A* with a similar heuristic, the algorithm would explore the states closer to the goal first and stop the exploration as soon as all unknown states are provably farther away from the goal than the path followed by the current policy. However, this is only one possible heuristic to aid exploration. Another example is to choose or learn an appropriate STATE REPRESENTATION, which is one major focus of this thesis and exhaustively discussed in Chapter 4.

2.3 Challenges to autonomous control

The field of autonomous control has made tremendous advances in the last two decades, but autonomous agents still have a long way to go. During my PhD studies I identified a wide range of obstacles to autonomy. In this section I summarize some of the challenges to autonomous control that are beyond the scope of this thesis, but offered me enough food for thought to be worth mentioning.

2.3.1 Large state spaces

State spaces of real tasks are often too large for exploration. Some, like open ended, multi-objective tasks, as often envisioned in robotics, are even too large for deductive planning. To allow farther planning horizons, HIERARCHICAL RL (HRL, Sutton et al., 1999) introduces OPTIONS, which represent partial policies (or simply sequences of actions) that end up with a high probability in some set of target states. These options are treated as additional actions that can be chosen in all states within the option's scope. The agent must therefore only learn one decision to securely traverse long distances in state space. For example, walking from my office to a specific room in the same building can take a thousand or more steps. Rolling out all possible trajectories of this length is obviously not feasible in practice. Given a set of options in each room, which each navigate to one neighboring room or hallway, the same task can be solved within a dozen decisions. Whether or not HRL speeds up planning/learning depends primarily on the usefulness of the provided options (Solway et al., 2014). A host of literature has been dedicated to the identification of general-purpose options (see e.g. Simsek and Barto, 2008; Konidaris et al., 2010; Solway et al., 2014) and has revealed some interesting connections to neural representations (Ribas-Fernandes et al., 2011; Schapiro et al., 2013).

HIERARCHICAL RL

HRL is following an implicit assumption, though. Take the example of navigation in a building. The states (possible positions) in each room are limited and one can plan/learn efficient options to reach the target states, in this case doors leaving the room. Tasks involving multiple objects, which can be picked up and moved between the rooms, on the other hand, would increase the state space tremendously. Moreover, there is no structure of "shortcuts" HRL could exploit with options. Every original option must be planned/learned w.r.t. all possible positions all objects in the building can adopt. These COMBINATORIAL STATE SPACES grow exponentially in the number of variables like movable objects. Planning/learning are often not feasible here. However, HRL's intuition of temporal abstractions may still be valid, if one interprets an option as an optimization problem, rather than a fixed policy.

COMBINATORIAL STATE SPACES

From this point of view, a higher hierarchical layer represents the state and possible transitions more coarsely, for example, in a lower resolution or as symbols and

their relationships. On this level, options are the action primitives that transfer the abstract state into an abstract target state. Planning is therefore less detailed, but can predict outcomes over long time horizons. Executing a chosen option is itself a planning problem on the next lower layer, in which only the abstracted target states are rewarded. In difference to the classical interpretation, however, this planing problem does not need to consider *every* possible combination of variables. In the above example, only the objects currently present in the room can possibly affect a plan to reach the door. This constitutes only a TASK-RELEVANT SUBSPACE of the original combinatorial state space (see Section 2.1.4), and optimization may become feasible. However, there are many side-effects that can invalidate plans: maybe the key to open a door is not in the same room, so that the lower level must refer a new sub-goal to the higher planning process. Nonetheless, if these side-effects are expressed sufficiently in the higher (e.g. the symbolic) layer, the curse of dimensionality in combinatorial state spaces can be broken. I shortly discuss hierarchies in the context of factored transition models in Chapter 5, but can not claim to have solved any major problem. Nonetheless, hierarchical RL in combinatorial state spaces, in combination with factored representations, seems very promising and calls for further research.

2.3.2 Multi-agent systems

GAME THEORY

The interaction between multiple agents in the same environment is the domain of classical GAME THEORY (Neumann and Morgenstern, 1944). Here the action space is composed of all possible *combinations* of actions, that individual agents may choose. This leads usually to *huge* action spaces. Notice that each agent's optimal policy can depend on all other decisions. For example, when multiple robots carry large objects together, each robot's decision influences the others optimal reactions. Each robot must therefore check whether his response is in line with the group's and change it accordingly. This change may also evoke additional changes in other robots' decisions or entire policies, until a so called NASH EQUILIBRIUM is reached (if one exists, Nash, 1951). Yasuda and Ohkura (2008) demonstrate such a coordinated control using pressure sensors to "communicate" actions. However, many tasks require more foresight and agents' Q-values must incorporate the combined behavior over longer time horizons. Agents must therefore be able to estimate each others policies. As this usually takes time, is error-prone and vulnerable to deception, there exist (to the best of my knowledge) no general solvers to achieve the Nash equilibrium.

ZERO-SUM GAMES

The best known example for solvable multi-agent interaction are ZERO-SUM GAMES for agents with access to the same information (Neumann and Morgenstern, 1944). Here agents belong to two COMPETITIVE factions with exactly opposing reward functions and the MINIMAX THEOREM guarantees that all agents arrive at the same (or inverted) value function (Osborne and Rubinstein, 1994). Agents of one faction will therefore always choose actions that maximize the value, whereas the competing faction will always minimize it. In principle, this is a simple extension of the policy improvement step, similar to RISK-SENSITIVE REINFORCEMENT LEARNING (Shen et al., 2013). In practice, however, the exponential growth of the action space overburden classical MDP based planners and learners. Most existing approaches focus therefore on special state-action representations to speed up computation (for example Guestrin et al., 2001b; Petrik and Zilberstein, 2009).

Many autonomous agents are supposed to live in a world populated with humans and other agents. Very few tasks in these environments are zero-sum games, though. A personal-assistant robot, for example, may meet other people and robots on its way to shop in the local grocery store. None of these pedestrians have a purely competitive or collaborative role. The development of reinforcement learning algorithms that can identify and deal with other agents' intentions is therefore an important challenge for future research. In particular predicting human (inter-)actions will become a major focus, as robots leave their cages in INDUSTRY 4.0 FACTORIES.

2.3.3 Imitation learning

Humans are "cultural" animals. We learn much of our behavior by imitating other people. This allows us to transfer skills, but also to know how to act in situations we have never encountered before. The analogous IMITATION LEARNING (see e.g. the overview of Argall et al., 2009) aims at teaching autonomous agents the desired behavior by demonstration, rather than specifying rewards heuristically by hand. Due to applications in industrial robotics (e.g. the Baxter robot by rethink robotics, 2008), the field has become very popular recently. Imitation learning has two essential advantages: (i) the agent can learn expert skills that are hard to formalize and (ii) the agent can focus on the demonstrated states, which can speed up learning/planning in large state spaces immensely (Kober et al., 2013).

Formally, the structure of MDP does not assign SEMANTIC MEANING to states: the same continuous state space $\mathcal{X} := [0,1]^2$ could represent a position in any building, town or planet. Any semantic is imposed by the transition model P. With a fixed P, an agents optimal behavior π^* depends exclusively on the mean reward function r, i.e. $\pi^* = \hat{\Theta}[r] := \arg\sup_\pi\{\|v^\pi\| \mid v^\pi = \hat{\Gamma}_\pi[r] + \gamma\hat{\Gamma}_\pi[\hat{P}[v^\pi]]\}$. This optimal policy π^* can be found with policy iteration (Section 2.2.4). APPRENTICESHIP LEARNING turns the relationship between policy and reward on its head, by finding the optimal reward function r^* that fits some observed behavior π (Abbeel and Ng, 2004; Klein et al., 2011), i.e. $r^* = \arg\inf_r\{d(\pi, \hat{\Theta}[r])\}$. Here $d(\cdot,\cdot)$ denotes a distance measure between policies (e.g. the *Kullback-Leibler divergence*), which can be estimated on a set of *training trajectories* executing π. For example, Abbeel et al. (2007) successfully learn spectacular aerobatic flight maneuvers with a remote-controlled helicopter from demonstrations of human experts. The recorded flight trajectories serve here as training sets for both the transition model P and the optimal reward r^*. Note, however, that apprenticeship learning requires the teacher to demonstrate the imitated behavior *as* the learning agent. An autonomous car can be the apprentice of a human driver, but an autonomous robot can never learn how to walk by apprenticeship learning, unless a human learns to remote-control it first (or moves the robot by hand as in Englert et al., 2013).

APPRENTICESHIP LEARNING

Before an agent can learn how to walk by observing a human, it must first learn to map the human motion into its own frame of reference, which is called the CORRESPONDENCE PROBLEM (Nehaniv and Dautenhahn, 2002). This usually involves three steps: (i) one needs to *detect* the human's states and actions. For example, Lukic et al. (2014) record the demonstrations from the perspective similar to the robot's. The end-effector positions (states) and movements (actions) are extracted using computer-vision algorithms and the imitated behavior is learned with apprenticeship learning. (ii) most agents will not have access to observations from the trainers perspective and thus have to *map* the trainer's state into the agent's.

CORRESPONDENCE

This is not a trivial problem as Alissandrakis et al. (2007) show for bodies with the same degree of freedom. (iii) the imitating agent needs to learn two transition models P, one to estimate the trainer's reward r^* with apprenticeship learning and one to plan its own policy π^*. Using all three steps, autonomous robots have successfully imitated behavior as complex as catching objects in flight (Kim et al., 2014).

I am convinced that imitation is the future of task specification. Performing this *autonomously* may even remove the notion of singular "tasks" in favor of continuously meaningful interaction with the environment. A household robot, for example, may be expected to clean, watch the baby and communicate at the same time, without any of these tasks having a defined beginning or end. However, in order to learn by observation, the agents needs to autonomously decide *what* should be imitated. As mentioned in Section 2.3.2, this may require to consider human intentions. In face of an incomprehensible amount of possibilities, the question of *what* to imitate (and how to *correct* misjudgments) is of enormous importance for true autonomy and calls for future research.

Chapter 3

Machine Learning

Reinforcement learning is a natural choice for autonomous control in probabilistic environments. I have shown in Section 2.2 how well this inductive technique can capture arbitrary control problems in finite state spaces. When confronted with continuous states, it seems to be natural to turn to classical MACHINE LEARNING (ML) techniques. Many who used APPROXIMATE REINFORCEMENT LEARNING algorithms were surprised by their often unexpected behavior, though. Dealing with the inaccurate representation of functions in continuous input spaces, ML faces a host of problems that are foreign to classical RL analysis in $L^\infty(\mathcal{X})$. Reversely, the dynamic properties of RL introduce sources of error never seen before in the ML framework.

In this chapter, I introduce the basic ML formalism, starting with Hilbert spaces in Section 3.1. Techniques and pitfalls of ML are demonstrated at the example of regression in Section 3.2 and the application to RL is reviewed in Section 3.3. I finish with my personal assessment of challenges to machine learning for autonomous control, and how these are addressed in this thesis, in Section 3.4.

3.1 Function spaces

I demonstrated in Section 2.2 how well the reinforcement learning operations *value iteration* and *policy improvement* can be performed in $L^\infty(\mathcal{X})$. In particular the convergence guarantee for Q-learning in Proposition 2.6 on Page 16 is closely tied to the supremum norm of $L^\infty(\mathcal{X})$ and cannot be replicated in other L^p spaces. However, estimating the values *point-wise*, as in Equations 2.9, 2.11 and 2.14, is only possible in finite state spaces. As every state must be visited (multiple times), no finite Markov chain can estimate the value in an infinite state space (for example real coordinates $X \subset \mathbb{R}^d$) using the update rules presented in Chapter 2.

3.1.1 Hilbert spaces

Many prominent machine learning techniques solve this problem by reformulating the optimization in terms of INNER PRODUCTS in the space of SQUARE-INTEGRABLE FUNCTIONS $L^2(\mathcal{X}, \xi)$. The probability measure $\xi : \mathcal{B}(\mathcal{X}) \to [0, 1], \int \xi(dx) = 1$, refers here to the distribution of *i.i.d.* training data. $L^2(\mathcal{X}, \xi)$ contains[1] all functions from $L^\infty(\mathcal{X})$, and therefore all possible value functions. $L^2(\mathcal{X}, \xi)$ are also the *only*

L² SPACES

[1] $L^2(\mathcal{X}, \xi)$ contains all functions f that are bounded *almost everywhere* w.r.t. ξ, that is $\|f\|_\xi < \infty$. This includes all point-wise bounded functions, in other words all functions in $L^\infty(\mathcal{X})$.

HILBERT SPACES amongst the L^p spaces, that is, have an inner product $\langle \cdot, \cdot \rangle_\xi$. For sufficiently large training sets $\{x_t\}_{t=1}^n$, drawn i.i.d. from ξ, holds $\forall f, g \in L^2(\mathcal{X}, \xi)$:

$$\|f\|_\xi^2 \;=\; \langle f, f \rangle_\xi \,, \qquad \langle f, g \rangle_\xi \;=\; \int \xi(dx)\, f(x)\, g(x) \;\approx\; \tfrac{1}{n} \sum_{t=1}^{n} f(x_t)\, g(x_t) \,. \tag{3.1}$$

Approximating norms and inner products between functions is therefore equivalent to the respective averages over the training set. Note that for non-stationary distributions the average can be approximated by online learning. When all computations can be expressed in terms of inner products between functions, the result can therefore be approximated *without* visiting each state $x \in \mathcal{X}$!

Markov chains are in general not *i.i.d.* distributed, though. Furthermore, some TRANSIENT Markov chains may never reach or return to a state (or region of $\mathcal{X}$), after some crucial transition has occurred. Here no STEADY STATE DISTRIBUTION ξ exist and empirical averages do not converge to the inner product. Non-transient
ERGODICITY Markov chains, on the other hand, are called ERGODIC[2], and the transition models P and policies π used to generate them yield the ERGODIC TRANSITION OPERATORS $\hat{\Gamma}_\pi[\hat{\mathrm{P}}] : L^2(\mathcal{X}, \xi) \to L^2(\mathcal{X}, \xi)$ and $\hat{\mathrm{P}}[\hat{\Gamma}_\pi] : L^2(\mathcal{X} \times \mathcal{A}, \xi\pi) \to L^2(\mathcal{X} \times \mathcal{A}, \xi\pi)$. Ergodicity guarantees the existence of steady state distribution ξ with

$$\iint \xi(dx)\, \pi(da|x)\, P(B|x, a) \;=\; \xi(B) \,, \quad \forall x \in \mathcal{X},\ \forall a \in \mathcal{A},\ \forall B \in \mathcal{B}(\mathcal{X}) \,. \tag{3.2}$$

Inner products and norms in $L^2(\mathcal{X}, \xi)$ can therefore be approximated by estimating the empirical mean over ergodic Markov chains. Transient Markov chains are in practice often reset after a predetermined number of transitions to enforce ergodicity.

Using the fact that training samples are (after a random permutation) drawn i.i.d. from ξ, it is easy to show that the estimation operator $\hat{\mathrm{E}} : L^2(\mathcal{X}, \xi) \to \mathbb{R}$ of the point-wise product[3] $[f \cdot g]$, $\hat{\mathrm{E}}[f \cdot g] := \frac{1}{n} \sum_{t=1}^n f(x_t)\, g(x_t)$, is a *bias-free* estimate of $\langle f, g \rangle_\xi$ and its *variance* decreases with n:

$$\mathbb{E}\Big[\hat{\mathrm{E}}[f \cdot g]\Big] = \langle f, g \rangle_\xi \,, \quad \text{and} \quad \mathbb{E}\Big[\big(\hat{\mathrm{E}}[f \cdot g] - \langle f, g \rangle_\xi\big)^2\Big] = \frac{1}{n}\Big(\underbrace{\langle f^2, g^2 \rangle_\xi - \langle f, g \rangle_\xi^2}_{\text{variance of } [f \cdot g]}\Big) \,. \tag{3.3}$$

Note that the variance of $\hat{\mathrm{E}}$ (and thus the confidence in the estimate) also depends on the variance of the point-wise product $[f \cdot g]$ over the domain $\mathcal{X}$. Products over functions that do not change much over $\mathcal{X}$ can therefore be reliably estimated with few samples. If one of the functions is afflicted by normal distributed i.i.d. noise, i.e. $\tilde{g}(x_t) := g(x_t) + \epsilon_t, \epsilon_t \sim \mathcal{N}(0, \sigma^2)$, only the estimate's variance changes:

$$\mathbb{E}\Big[\big(\hat{\mathrm{E}}[f \cdot \tilde{g}] - \langle f, g \rangle_\xi\big)^2\Big] \;=\; \frac{1}{n}\Big(\langle f^2, g^2 \rangle_\xi - \langle f, g \rangle_\xi^2\Big) \;+\; \frac{\sigma^2}{n} \Big\|f\Big\|_\xi^2 \,. \tag{3.4}$$

MEAN-VARIANCE TRADE-OFF Estimations based on noisy observations of g often attempt a MEAN-VARIANCE TRADE-OFF, by enforcing a low function norm $\|f\|_\xi^2$. Although this introduces a

[2] Formally, a Markov chain is *ergodic* if it is *aperiodic* and *positive recurrent*. This implies there exists a non-zero probability to reach each (infinite) Borel set $B \in \mathcal{B}(\mathcal{X})$ from each state $x \in \mathcal{X}$ eventually. Two Markov chains sampled with the same ergodic transition models will therefore converge to the same distribution ξ, although the required amount of samples can be enormous.

[3] Throughout this thesis, I use a central dot as *infix* operator $\cdot : L^2(\mathcal{X}, \xi) \times L^2(\mathcal{X}, \xi) \to L^2(\mathcal{X}, \xi)$ to denote the point-wise product of two functions $[f \cdot g](x) = f(x)\, g(x)\,, \forall x \in \mathcal{X}, \forall f, g \in L^2(\mathcal{X}, \xi)$. Note however, that the same dot also denotes an unnamed parameter if standing alone, e.g. $f(\cdot)$.

bias to the estimate, the certainty/reliability can be greatly improved. This is one example of *regularization terms* or *priors*, which are discussed in Section 3.2.

3.1.2 Reproducing kernel Hilbert spaces

There is one particular class of of Hilbert spaces that is of importance to this thesis, partially because its use in *kernel SFA* (Böhmer et al., 2012), partially because it provides unique insight into regularization (Section 3.2.2). Here $\kappa \in L^\infty(\mathcal{X} \times \mathcal{X})$ is called a POSITIVE SEMI-DEFINITE KERNEL[4] on $\mathcal{X}$ if $\iint \xi(dx)\,\xi(dy)\,f(x)\,\kappa(x,y)\,f(y) \geq 0$, $\forall f \in L^2(\mathcal{X},\xi)$. A positive semi-definite kernel induces a linear operator $\hat{\mathrm{K}} : L^2(\mathcal{X},\xi) \to L^2(\mathcal{X},\xi)$, i.e. $\hat{\mathrm{K}}[f](x) = \int \xi(dy)\,\kappa(x,y)\,f(y)$, with infinitely[5] many EIGENVALUES $\lambda_i \in [0,\infty)$ and EIGENFUNCTIONS $\varphi_i \in L^2(\mathcal{X},\xi)$: KERNEL

$$\exists \varphi_i \in L^2(\mathcal{X},\xi): \quad \hat{\mathrm{K}}[\varphi_i] \;=\; \lambda_i\,\varphi_i\,, \quad \text{with} \quad \begin{smallmatrix}\lambda_i \in [0,\infty)\\ \langle\varphi_i,\varphi_j\rangle_\xi = \delta_{ij}\end{smallmatrix}\,, \quad \forall i,j \in \mathbb{N}\,. \tag{3.5}$$

The eigenvalues are non-negative due to positive-semi-definiteness and the eigenfunctions form due to the *Hilbert-Schmidt theorem* an ORTHONORMAL BASIS of $L^2(\mathcal{X},\xi)$, that is, $\langle\varphi_i,\varphi_j\rangle_\xi = \delta_{ij}$, $\forall i,j \in \mathbb{N}$. Every function $f \in L^2(\mathcal{X},\xi)$ can therefore be expressed as linear combination $f(x) = \sum_{i=0}^\infty \langle f,\varphi_i\rangle_\xi\,\varphi_i(x)$, $\forall x \in \mathcal{X}$. Furthermore, one can rewrite the kernel in terms of eigenfunctions (one of the central conclusions from *Mercer's theorem*, Mercer, 1909):

$$\kappa(x,y) = \sum_{i=0}^\infty \varphi_i(x)\,\lambda_i\,\varphi_i(y)\,, \qquad \forall x,y \in \mathcal{X}\,. \tag{3.6}$$

The eigenvalues and eigenfunctions of $\hat{\mathrm{K}}$ also induce the inner product of the REPRODUCING KERNEL HILBERT SPACE $\mathcal{H}_\kappa$ (RKHS, Schölkopf and Smola, 2001): RKHS

$$\mathcal{H}_\kappa \;:=\; \left\{ f \in L^2(\mathcal{X},\xi) \;\middle|\; \|f\|^2_{\mathcal{H}_\kappa} = \langle f,f\rangle_{\mathcal{H}_\kappa} < \infty \right\}, \tag{3.7}$$

$$\langle f,g\rangle_{\mathcal{H}_\kappa} \;=\; \sum_{i=0}^\infty \langle f,\varphi_i\rangle_\xi\,\lambda_i^{-1}\,\langle\varphi_i,g\rangle_\xi\,, \qquad \forall f,g \in L^2(\mathcal{X},\xi)\,. \tag{3.8}$$

Lemma 3.1 (see e.g. Schölkopf and Smola, 2001) *RKHS $\mathcal{H}_\kappa$ induced by positive-semi-definite kernel $\kappa : L^\infty(\mathcal{X}\times\mathcal{X})$ (i) contains all functions $k_x(y) := \kappa(x,y), \forall x,y \in \mathcal{X}$, and (ii) has the "reproducing property" $\langle f,k_x\rangle_{\mathcal{H}_\kappa} = f(x)$, $\forall x \in \mathcal{X}$, $\forall f \in \mathcal{H}_\kappa$.*

Proof: can be found in Appendix C.3 on Page 236. □

RKHS provide the theoretical background to the KERNEL TRICK. Every linear algorithm, that can be formulated in terms of inner products between training samples, can be non-linearly extended by replacing the inner product with a positive-definite kernel κ. As the kernel can be written as an inner product $\kappa(x,y) = \sum_{i=0}^\infty \varphi_i(x)\lambda_i\varphi_i(y)$, this substitution is equivalent to a projection of the data into the the potentially infinite-dimensional space $\Psi := \{\boldsymbol{\psi}(x) \,|\, \psi_i(x) = \sqrt{\lambda_i}\varphi_i(x), \forall i \in$ KERNEL TRICK

[4] There is a plethora of equivalent definitions. See Schölkopf and Smola (2001) for an overview.

[5] Some kernels have only a finite set of non-zero eigenvalues and corresponding eigenfunctions. However, each finite basis can always be *completed* with additional orthonormal basis functions to yield a complete basis of $L^2(\mathcal{X},\xi)$. By setting the corresponding eigenvalues $\lambda_i = 0$, one can make sure that the choice of additional eigenfunctions φ_i does not affect the kernel.

$\mathbb{N}, \forall x \in \mathcal{X}\}$. A linear function in Ψ can be non-linear[6] in the original input space $\mathcal{X}$. Moreover, the REPRESENTER THEOREM (Wahba, 1990; Schölkopf and Smola, 2001) ensures that all solutions to any empirical cost function (see below) can be found within the span of the functions k_x from Lemma 3.1.

Theorem 3.2 (from Schölkopf and Smola, 2001) *Let $\kappa \in L^\infty(\mathcal{X} \times \mathcal{X})$ be a positive-definite kernel on the non-empty set $\mathcal{X}$ with corresponding RKHS $\mathcal{H}_\kappa$, and let $\hat{\mathrm{C}}$ be any empirical cost based only on training set $\{x_t, y_t\}_{t=1}^n \subset \mathcal{X} \times \mathbb{R}$ with a strictly monotonically increasing real-valued regularization $g : [0, \infty) \to \mathbb{R}$. Any solution $f^* \in \mathcal{H}_\kappa$ can be represented as linear combination of parameterized kernel functions,*

$$f^* \in \operatorname*{arg\,inf}_{f \in \mathcal{H}_\kappa} \hat{\mathrm{C}}[f](\{x_t, y_t\}_{t=1}^n) + g(\|f\|_{\mathcal{H}_\kappa}) \quad \Rightarrow \quad f^*(\cdot) = \sum_{t=1}^n a_t \, \kappa(\cdot, x_t)\,, \quad \exists \boldsymbol{a} \in \mathbb{R}^n\,.$$

Proof: can be found in Appendix C.3 on Page 237. □

Interested readers are referred to Schölkopf and Smola (2001), Shawe-Taylor and Cristianini (2004), and to my paper about kernel SFA in Appendix A.1 on Page 124.

3.2 Optimization

I want to demonstrate this thesis' approach to optimization, and machine learning in general, at the example of estimation of the *mean reward function* defined in Equation 2.5 on Page 12. Estimating a function from noisy samples is called REGRESSION. REGRESSION. I have published a paper on regression (with *linear factored functions*, Böhmer and Obermayer, 2015), which can be found in Appendix B.2 on Page 216. For regression one assumes a TRAINING SET of observed *samples* $\{x_t, a_t\}_{t=1}^n \subset \mathcal{X} \times \mathcal{A}$ and of observed *labels* $\{r_t\}_{t=1}^n \subset \mathbb{R}$, in our case rewards. The task is to predict the labels of a TEST SET of (potentially unseen) samples. All samples are in general assumed to be i.i.d. drawn from $x_t \sim \xi$ and $a_t \sim \pi$. This is also the case when an ergodic Markov chain is long enough to to converge to steady state distribution ξ. Recall that rewards r_t are drawn from $R(\cdot|x_t, a_t, x_{t+1})$, and are thus i.i.d. w.r.t. $p(\cdot|x_t, a_t) = \int_{\mathcal{X}} P(dx'|x_t, a_t)\, R(\cdot|x_t, a_t, x')$ for any infinite Markov chain. As both this distribution and the mean reward function always depend on states and actions, I will in the following use the notation $(x_t, a_t) =: z_t \in \mathcal{Z} := \mathcal{X} \times \mathcal{A}$, $\zeta(dz) := \xi(dx)\, \pi(da|x)$ and $L^2(\mathcal{Z}, \zeta) := L^2(\mathcal{X} \times \mathcal{A}, \xi\pi)$.

3.2.1 Cost functions

Depending on assumptions on the reward distribution, one can formulate different COST FUNCTIONS (e.g. using different norms). I describe here only the most common assumption: the Gaussian distribution $p(r|z) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{1}{2\sigma^2}[r - f(z)]^2)$. The mean function $f : \mathcal{Z} \to \mathbb{R}$ can be learned using BAYES' RULE:

$$\overbrace{p(f(z_t) \,|\, r_t)}^{\text{posterior}} = \frac{\overbrace{p(r_t \,|\, f(z_t))}^{\text{likelihood}} \overbrace{p(f(z_t))}^{\text{prior}}}{\underbrace{p(r_t)}_{\text{evidence}}}\,. \tag{3.9}$$

[6] Except the *linear kernel* $\kappa(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^\top \boldsymbol{y}, \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, which projects back into $\mathbb{R}^d$. This can be easily confirmed by setting $\lambda_i = 1$ and $\varphi_i(\boldsymbol{x}) = x_i$ for $1 \leq i \leq d$, and $\lambda_i = 0$ otherwise.

Bayes' rule is recursive: the posterior of time t becomes the prior at time $t+1$. The function $f \in L^2(\mathcal{Z},\zeta)$ that maximizes this *joint posterior* is a good candidate for the mean reward function in Equation 2.5. This optimization problem is equivalent[7] to the REGULARIZED LEAST-SQUARES cost function: LEAST SQUARES

$$\sup_f \underbrace{\prod_{t=1}^n \frac{p(r_t \,|\, f(z_t))}{p(r_t)}\, p(f)}_{\text{joint posterior}} \quad \equiv \quad \inf_f \tfrac{1}{2n} \sum_{t=1}^n \underbrace{\big(r_t - f(z_t)\big)^2}_{\text{least-squares error}} - \tfrac{\sigma^2}{n} \underbrace{\log\big(p(f)\big)}_{\text{prior on } f}. \tag{3.10}$$

The second part of the right hand side is the PRIOR, weighted by the scaled variance $\frac{\sigma^2}{n}$ of the rewards. The prior represents our belief how the reward *should* be, and is equivalent to a regularization term (see below). Note the similarities to the noise-induced term in Equation 3.4, which would be equivalent for $p(f) = \frac{1}{\sqrt{\pi}} \exp(-\|f\|_\zeta^2)$. A Gaussian prior on f performs therefore a mean-variance trade-off. It is easy to see that least squares (without the prior) estimates $\frac{1}{2}\|r-f\|_\zeta^2$ similar to Equation 3.1. However, Lemma 3.3 shows that least squares estimates are not *bias-free*. In the theoretical limit holds nonetheless $\lim_{n\to\infty} \inf_f \frac{1}{2n} \sum_{t=1}^n \big(r_t - f(z_t)\big)^2 \equiv \inf_f \frac{1}{2}\|f-r\|_\zeta^2$.

Lemma 3.3 *Let $\{r_t\}_{t=1}^n$ be i.i.d. normal distributed with means $\{r(z_t)\}_{t=1}^n$ and variance σ^2. For the least squares operator $\hat{\mathrm{L}}[f] := \frac{1}{2n}\sum_{t=1}^n \big(r_t - f(z_t)\big)^2$ holds:*

$$\mathbb{E}\big[\hat{\mathrm{L}}[f]\big] = \tfrac{1}{2}\|r-f\|_\zeta^2 + \tfrac{\sigma^2}{2} \qquad \text{and} \qquad \mathbb{E}\Big[\big(\hat{\mathrm{L}}[f] - \mathbb{E}[\hat{\mathrm{L}}[f]]\big)^2\Big] = \frac{\sigma^2}{n}\Big(\|r-f\|_\zeta^2 + \tfrac{\sigma^2}{2}\Big).$$

Proof: can be found in Appendix C.3 on Page 237. □

Solving Equation 3.10 (without prior) requires to choose a parameterized *function class* $\mathcal{F}$ and to find (one of the) function parameters that minimize $\hat{\mathrm{L}}$. I want to demonstrate this with the example of ORDINARY LEAST-SQUARES (OLS). Here the function class $\mathcal{F}_\phi := \{\boldsymbol{a}^\top \boldsymbol{\phi}(\cdot) \,|\, \boldsymbol{a} \in \mathbb{R}^p\}$ consists of all linear combinations of p basis functions $\{\phi_i \in L^2(\mathcal{Z},\zeta)\}_{i=1}^p$. The corresponding linear coefficients $\boldsymbol{a} \in \mathbb{R}^p$ are the p parameters that have to be optimized using the estimation operator $\hat{\mathrm{E}}$ from Section 3.1.1, where $\tilde{r}$ denotes that r is afflicted by zero mean i.i.d. noise: OLS

$$\boldsymbol{a}^* \quad := \quad \operatorname*{arg\,inf}_{\boldsymbol{a}\in\mathbb{R}^p} \hat{\mathrm{L}}[\boldsymbol{a}^\top\boldsymbol{\phi}(\cdot)] \quad = \quad \hat{\mathrm{E}}[\boldsymbol{\phi}\cdot\boldsymbol{\phi}^\top]^{-1}\, \hat{\mathrm{E}}[\boldsymbol{\phi}\cdot\tilde{r}] \quad \in \quad \mathbb{R}^p. \tag{3.11}$$

The notation $\hat{\mathrm{E}}[\boldsymbol{\phi}\cdot\boldsymbol{\phi}^\top]$ denotes here a matrix of $p\times p$ estimations $\hat{\mathrm{E}}[\phi_i\phi_j]$ and $\mathbf{A}^{-1}$ indicates the inverse of matrix $\mathbf{A} \in \mathbb{R}^{p\times p}$. Note that both estimations are bias-free and that in the limit of infinite samples the OLS solution $\boldsymbol{a}^\top\boldsymbol{\phi}(\cdot)$ converges against the PROJECTION OPERATOR $\hat{\Pi}_\zeta^\phi : L^2(\mathcal{Z},\zeta) \to L^2(\mathcal{Z},\zeta)$, i.e.

$$\hat{\Pi}_\zeta^\phi[f](z) \quad := \quad \langle f, \boldsymbol{\phi}^\top\rangle_\zeta \langle \boldsymbol{\phi}, \boldsymbol{\phi}^\top\rangle_\zeta^{-1} \boldsymbol{\phi}(z)\,, \qquad \forall f \in L^2(\mathcal{Z},\zeta)\,, \quad \forall z \in \mathcal{Z}\,. \tag{3.12}$$

During my PhD I have also studied CLASSIFICATION to showcase features learned by *slow feature analysis* (Böhmer et al., 2012). One can interpret the case with two CLASSIFICATION

[7] In this thesis, I call two optimization problems over domain $\mathcal{Y}$ *equivalent* ($\equiv$), when they are solved by the same element(s) $y \in \mathcal{Y}$. Here I used $\sup_y c(y) \equiv \inf_y -c(y)$, $\inf_y c(y) \equiv \inf_y \log(c(y))$, $\inf_y c(y) \pm a \equiv \inf_y c(y)$ and $\inf_y ac(y) \equiv \inf_y c(y)$, with $a > 0$ and $c(y) \geq 0, \forall y \in \mathcal{Y}$.

classes as regression with binary targets (e.g. $\{-1, 1\}$), but some specialized approaches can improve dramatically upon that. Examples are the *maximum margin* classifiers (support vector machines, Vapnik, 1995; Boser et al., 1992) and iterative improvements by *perceptrons* (Rosenblatt, 1962) and *logistic regression* (iteratively reweighted least squares, Rubin, 1983). Although a major branch of machine learning, classification is not relevant for the autonomous control sketched out in this thesis. Interested readers are referred to my paper in Appendix A.1 on Page 124.

3.2.2 Generalization

In practice, no finite training set from a continuous (infinite) input space $\mathcal{Z}$ can visit *every* sample $z \in \mathcal{Z}$. The function output *between* the training samples does not influence the operator $\hat{\mathrm{L}}$, though. As $L^2(\mathcal{Z}, \zeta)$ includes all bounded functions from $L^\infty(\mathcal{Z})$, there are always infinitely many functions $f \in \arg\inf_f \hat{\mathrm{L}}[f] \subset L^2(\mathcal{Z}, \zeta)$, that minimize $\hat{\mathrm{L}}$. The GENERALIZATION ERROR $\|f - r\|_\zeta^2$ between learned function f and target r can therefore be arbitrarily bad. Most f will not generalize well to unseen samples. Furthermore, as each training sample is almost aways visited only once, $\hat{\mathrm{L}}$ fits the observed *noisy reward* rather than the *mean reward*. Predictions will therefore be suboptimal, even on the training set, until each $z \in \mathcal{Z}$ has been visited multiple times. The combination of both effects is called OVER-FITTING, which is a major obstacle in machine learning. Over-fitting is usually detected when the costs on the training set are significantly lower than on a test set, which had been withheld from training.

OVER-FITTING

Generalization to unseen samples has to rely on the labels of "similar" training samples. For example, a k-NEAREST NEIGHBOR (k-NN, Cover and Hart, 1967) estimator on $\mathcal{Z} \subset \mathbb{R}^d$ predicts the mean label of the k training samples with the smallest Euclidean distance to the test sample. Similarity between samples always requires a METRIC, i.e. a DISTANCE FUNCTION $d : \mathcal{Z} \times \mathcal{Z} \to [0, \infty)$, in this case the Euclidean metric $\|\cdot\|_2$ in $\mathbb{R}^d$. However, such a metric may be hard to construct and often fails to generalize sensibly to unseen states. In general, one can define an arbitrary SIMILARITY FUNCTION $\kappa \in L^\infty(\mathcal{Z} \times \mathcal{Z})$. Every positive (semi)-definite similarity kernel κ induces[8] a (semi)-metric with (semi)-norm distance function

METRIC ON INPUT SPACE

$$d_\kappa(x, y) \quad = \quad \sqrt{\kappa(x, x) - 2\kappa(x, y) + \kappa(y, y)}\,, \qquad \forall x, y \in \mathcal{Z}\,. \tag{3.13}$$

This is of particular interest when input space represents high-level "situations". Here descriptions are often high-dimensional but follow intuitively noticeable patters, that can be formalized by an experienced designer.

[8] To be precise, norms require $\forall x, y, z \in \mathcal{Z}$: (i) positivity $d_\kappa(x, y) \geq 0$, (ii) positive definiteness $d_\kappa(x, y) = 0$ if and only if $x = y$, (iii) symmetry $d_\kappa(x, y) = d_\kappa(y, x)$, and (iv) the triangle inequality $d_\kappa(x, z) \leq d_\kappa(x, y) + d_\kappa(y, z)$. As the name suggests, positive-definite kernel functions fulfill (i)–(iv). Positive semi-definite kernels fulfill all but (ii), and induce a semi-norm.

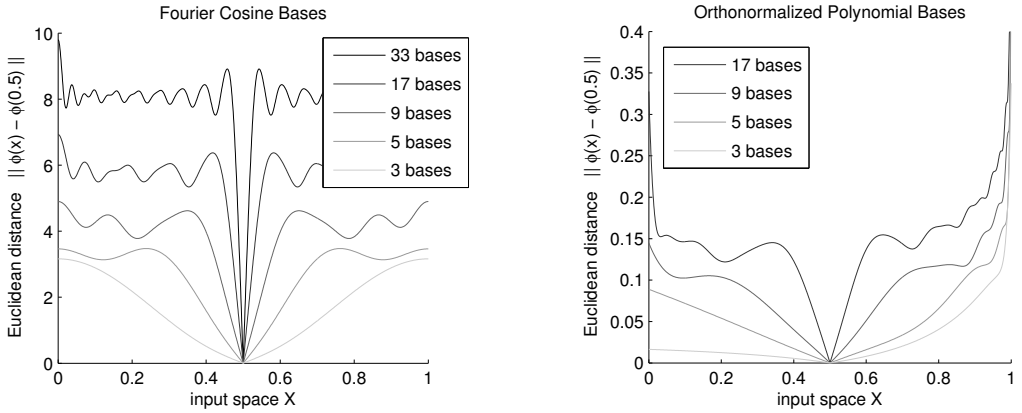


Figure 3.1: Bound on $|f(x) - f(0.5)|$ from Lemma 3.4, divided by $\|f\|_\zeta$. The left plot is based on the first Fourier cosine bases, the right on polynomials of degree 100. The former are orthonormal, the latter become increasingly similar and have been orthonormalized and eigenfunctions are included in order of their eigenvalues.

3.2.3 Induced metric

Not all input spaces provide such an intuition, though. For example, in Chapter 4 I will review works on camera images as input spaces. Images have thousands of pixels (dimensions) and the Euclidean metric does not generalize well. I will show how to construct a set of basis functions, which induce a metric on the input space.

Lemma 3.4 *For any set of orthonormal basis functions* $\{\phi_i\}_{i=1}^p \subset L^2(\mathcal{Z}, \zeta)$ *holds*

$$|f(x) - f(y)| \quad \leq \quad \|\boldsymbol{\phi}(x) - \boldsymbol{\phi}(y)\|_2 \ \|f\|_\zeta \,, \qquad \forall x, y \in \mathcal{Z} \,, \quad \forall f \in \mathcal{F}_\phi \,,$$

where $\mathcal{F}_\phi := \{\boldsymbol{a}^\top \boldsymbol{\phi}\}$ *denotes the space of linear combinations of basis functions.*

Proof: can be found in Appendix C.3 on Page 238. □

Lemma 3.4 restricts the possible function output "close" to the training samples, which in practice also averages out the i.i.d. distributed noise when the training samples are sufficiently "close by". The choice and number of bases functions implicitly CONSTRAINS therefore the admissible functions to be SMOOTH[9] w.r.t. a distance d_ϕ on $\mathcal{Z}$, which is equivalent to the Euclidean distance in *feature space* $\Phi := \{\boldsymbol{\phi}(z) \,|\, z \in \mathcal{Z}\}$, i.e. $d_\phi(x, y) := \|\boldsymbol{\phi}(x) - \boldsymbol{\phi}(y)\|_2$. Figure 3.1 shows this distance for Fourier and polynomial bases. Note that polynomial bases are not orthonormal, but can be orthonormalized (ensuring $\langle \phi_i, \phi_j \rangle_\zeta = \delta_{ij}$) like every set of basis functions in $L^2(\mathcal{Z}, \zeta)$. A statement similar to Lemma 3.4 can be made for arbitrary positive (semi-)definite similarity functions κ:

REPRESENTATION CONSTRAINS

Lemma 3.5 *Let* $d_\kappa(x, y) = \sqrt{\kappa(x, x) - 2\kappa(x, y) + \kappa(y, y)}$, $\forall x, y \in \mathcal{Z}$, *denote the (semi-)distance induced by positive (semi-)definite kernel* $\kappa \in L^\infty(\mathcal{Z} \times \mathcal{Z})$, *then*

$$|f(x) - f(y)| \quad \leq \quad d_\kappa(x, y) \ \|f\|_{\mathcal{H}_\kappa} \,, \qquad \forall x, y \in \mathcal{Z} \,, \quad \forall f \in \mathcal{H}_\kappa \,.$$

Proof: can be found in Appendix C.3 on Page 238. □

[9] In this thesis I use the word "smooth" in a figurative sense to indicate that the function output of samples "close by" is similar. Note that the mathematical definition is somewhat different.

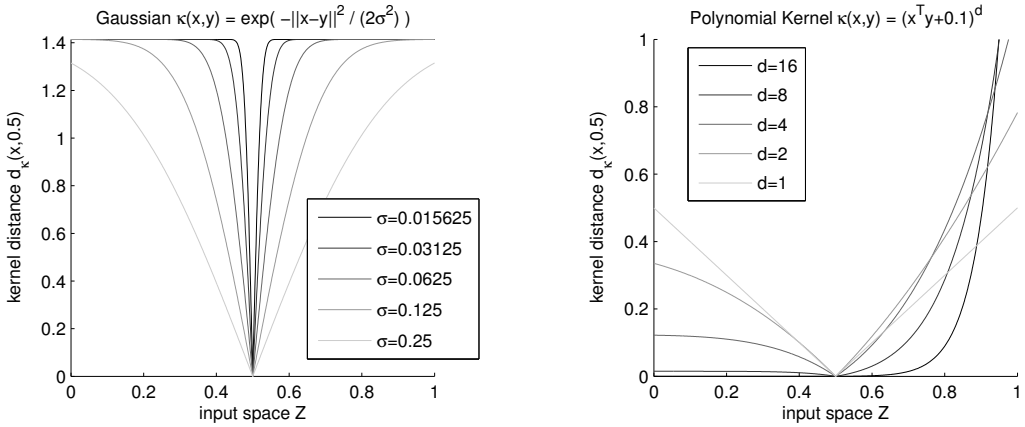


Figure 3.2: Bound on $|f(x)-f(0.5)|$ from Lemma 3.5, induced by Gaussian kernels (left) and polynomial kernels (right). Note that the Gaussian metric saturates at some distance and that the polynomial kernel has an unintuitive, "skewed" distance.

Here the metric on $\mathcal{Z}$ is the induced distance d_κ. Figure 3.2 shows this distance d_κ for Gaussian and polynomial kernels. Note that d_κ is also the Euclidean distance in the feature space $\Psi := \{\boldsymbol{\psi}(z) \,|\, \psi_i(z) = \sqrt{\lambda_i}\,\varphi_i(z)\,, \forall i \in \mathbb{N}\,, \forall z \in \mathcal{Z}\}$ spanned by the kernel's eigenvalues $\lambda_i \in \mathbb{R}$ and eigenfunctions $\varphi_i \in L^2(\mathcal{Z}, \zeta)$, that is,

$$d_\kappa(x,y) \;=\; \big\|\boldsymbol{\psi}(x) - \boldsymbol{\psi}(y)\big\|_2\,, \quad \psi_i(x) = \sqrt{\lambda_i}\,\varphi_i(x)\,, \quad \forall i \in \mathbb{N}\,, \quad \forall x, y \in \mathcal{Z}\,. \tag{3.14}$$

Any feature space, either specified explicitly by an orthonormal basis Φ or implicitly by a kernel Ψ, constrains therefore admissible functions to be smooth w.r.t. an induced metric on $\mathcal{Z}$. Assuming this metric generalizes well, one can avoid overfitting, by reducing the number of basis functions or changing the kernel parameters until the function is smooth enough, which is sometimes called PRUNING (Haykin, 1998).

3.2.4 Regularization

Adjusting the function class to perfectly fit the demands at hand is either very costly or simply impossible in practice. An alternative is to *allow* all considered functions, but to *prefer* smooth ones. This philosophy to prefer simple (here smooth) solutions goes back to OCCAM'S RAZOR and is called REGULARIZATION in machine learning (a term accredited to Andrey Tikhonov in 1963, see Haykin, 1998). Preferences can be formalized in *prior* distributions $p(f)$ over the function space, which yield a weighted regularization term $-\frac{\sigma^2}{n}\log p(f)$ as in Equation 3.10. The larger the REGULARIZATION CONSTANT σ^2, the smoother the solution. Literature contains a plethora of heuristic regularization terms, which encode prior knowledge about the expected function (see Haykin, 1998, for a comprehensive overview).

WEIGHT DECAY

I want to demonstrate this concept at the example of WEIGHT DECAY, a simple and popular technique, which is closely related to the presented framework. The intuition is that in smooth functions, samples similar (w.r.t. the induced metric) to training sample $z_t \in \mathcal{Z}$ should have similar labels r_t. One could therefore produce

VIRTUAL SAMPLES

VIRTUAL SAMPLES with the label r_t from a distribution centered around z_t. To allow an analytic solution for linear function spaces $\mathcal{F}_\phi := \{\boldsymbol{a}^\top\boldsymbol{\phi} \,|\, \boldsymbol{a} \in \mathbb{R}^p\}$, the "samples"

are drawn from a conditional Gaussian distribution[10] $\chi : \mathcal{Z} \times \mathcal{B}(\mathbb{R}^p) \to [0,1]$ in feature space Φ, rather than input space $\mathcal{Z}$. In the limit of infinitely many virtual samples $\tilde{\boldsymbol{\phi}} \in \mathbb{R}^p$, the least squares operator $\hat{\mathrm{L}}$ for function $f(\cdot) = \boldsymbol{a}^\top \boldsymbol{\phi}(\cdot) \in \mathcal{F}_\phi$ is

$$\tfrac{1}{2n} \sum_{t=1}^{n} \int \chi(d\tilde{\boldsymbol{\phi}}|\boldsymbol{z}_t) \left(\boldsymbol{a}^\top \tilde{\boldsymbol{\phi}} - r_t\right)^2 \quad = \quad \hat{\mathrm{L}}[\boldsymbol{a}^\top \boldsymbol{\phi}(\cdot)] \; + \; \tfrac{\sigma^2}{n} \|\boldsymbol{a}\|_2^2 . \tag{3.15}$$

The term *weight decay* derives from the equal punishment of all parameters, often called *weights*. Note that for orthonormal bases this regularization is equivalent to $\|f\|_\zeta^2$, or the prior $p(f) = \frac{1}{\sqrt{\pi}} \exp\left(-\|f\|_\zeta^2\right)$, and thus implements the mean-variance trade-off indicated in Equation 3.4. Minimizing Equation 3.15 is called RIDGE REGRESSION (Hoerl and Kennard, 1970), and has the solution RIDGE REGRESSION

$$\boldsymbol{a}^* \quad = \quad \left(\hat{\mathrm{E}}[\boldsymbol{\phi} \cdot \boldsymbol{\phi}^\top] + \tfrac{\sigma^2}{n} \mathbf{I}\right)^{-1} \hat{\mathrm{E}}[\boldsymbol{\phi} \cdot \tilde{r}] \quad \in \quad \mathbb{R}^p . \tag{3.16}$$

Virtual sampling in the feature space Ψ of positive (semi)-definite kernel κ yields the similar regularization term $\|f\|_{\mathcal{H}_\kappa}^2 = \boldsymbol{a}^\top \mathbf{K} \boldsymbol{a}$, with the KERNEL MATRIX $K_{ij} = \kappa(z_i, z_j), \forall i,j \in \{1,\ldots,n\}$. This regularization term is, for example, employed by GAUSSIAN PROCESSES (GP, Rasmussen and Williams, 2006). GP predict the mean function $f(\cdot) := \sum_{t=1}^{n} a_t \, \kappa(\cdot, z_t)$ by minimizing[11] the cost function $\hat{\mathrm{L}}[f] + \frac{\sigma^2}{n} \|f\|_{\mathcal{H}_\kappa}^2$, which yields GAUSSIAN PROCESSES

$$\boldsymbol{a}^* \quad = \quad \left(\mathbf{K} + \sigma^2 \, \mathbf{I}\right)^{-1} \boldsymbol{r} . \tag{3.17}$$

Weight decay is popular in practice primarily due to its simplicity. However, treating each parameter equally may not be the optimal regularization for the employed function class. For example, smoother basis functions (w.r.t. some underlying metric) should be regularized less, as should regions of $\mathcal{Z}$ with many training samples. In my publication Böhmer and Obermayer (2015), I extended virtual sampling to address these shortcomings. In the following I assume the input space $\mathcal{Z} \subset \mathbb{R}^d$ should be smooth w.r.t. the Euclidean metric in $\mathbb{R}^d$. Each training sample $\boldsymbol{z}_t \in \mathbb{R}^d$ shall generate an infinite set of virtual samples in $\mathbb{R}^d$ with Gaussian distribution χ:

$$\int \chi(d\boldsymbol{y}|\boldsymbol{z}_t) \, (\boldsymbol{y} - \boldsymbol{z}_t) = \mathbf{0} , \qquad \int \chi(d\boldsymbol{y}|\boldsymbol{z}_t) \, (\boldsymbol{y} - \boldsymbol{z}_t)(\boldsymbol{y} - \boldsymbol{z}_t)^\top = \boldsymbol{\Sigma} . \tag{3.18}$$

Without loss of generality[12] one can assume the covariance matrix $\boldsymbol{\Sigma}$ to be diagonal, with $\Sigma_{ij} = \sigma_i^2 \, \delta_{ij}, \forall i,j \in \{1,\ldots,d\}$. Furthermore, I assume that all admissible functions are *differentiable* (w.r.t. $\mathbb{R}^d$) and that $\nabla_k f := \frac{\partial}{\partial z_k} f \in L^2(\mathcal{Z}, \zeta)$ exists. The first order *Taylor approximation* of the virtual sampling least squares operator is

$$\tfrac{1}{2n} \sum_{t=1}^{n} \int \chi(d\boldsymbol{y}|\boldsymbol{z}_t) \Big(f(\boldsymbol{y}) - r_t\Big)^2 \quad \approx \quad \hat{\mathrm{L}}[f] \; + \; \sum_{k=1}^{d} \sigma_k^2 \underbrace{\tfrac{1}{2} \hat{\mathrm{E}}[(\nabla_k f)^2]}_{\hat{\mathrm{R}}_k[f]} . \tag{3.19}$$

The resulting regularization term $\sum_{k=1}^{d} \sigma_k^2 \, \hat{\mathrm{R}}_k[f]$ punishes un-smooth functions and the regularization can be adjusted for different input dimensions individually.

---

[10] The conditional Gaussian distribution $\chi(d\tilde{\boldsymbol{\phi}} \,|\, x)$ is centered around $\boldsymbol{\phi}(x) \in \mathbb{R}^p$ with variance $2\frac{\sigma^2}{n}$, which implies $\int \chi(d\tilde{\boldsymbol{\phi}}|x) \, \tilde{\phi}_i = \phi_i(x)$ and $\int \chi(d\tilde{\boldsymbol{\phi}}|x) \, \tilde{\phi}_i \, \tilde{\phi}_j = \phi_i(x) \, \phi_j(x) + 2\frac{\sigma^2}{n} \delta_{ij}, \; \forall i,j \in \{1,\ldots,p\}$.

[11] Gaussian processes actually learn distributions of functions. A full mathematical treatment of this complex topic is beyond the scope of this thesis, but under some simplifying assumptions (namely an invertible kernel matrix $\mathbf{K}$) the presented formalism yields the GP's *mean* function.

[12] Non-diagonal covariance matrices can be diagonalized by projecting the input samples onto the eigenvectors of $\boldsymbol{\Sigma}$. The the corresponding eigenvalues of $\boldsymbol{\Sigma}$ are the new diagonal elements σ_i^2.

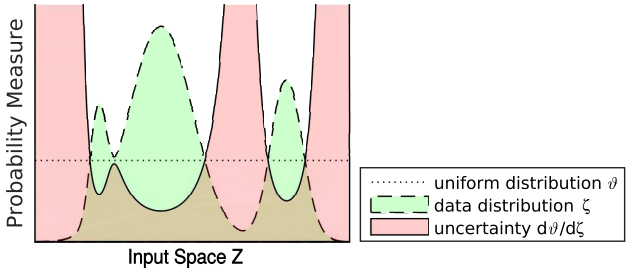


Figure 3.3: I interpret the Radon-Nikodym derivative $\frac{d\vartheta}{d\zeta}$ as *uncertainty measure* for our knowledge of $\mathcal{Z}$. Regularization enforces smoothness in uncertain regions. Adapted from Böhmer and Obermayer (2015).

The above virtual samples are distributed w.r.t. $p(B) := \frac{1}{n}\sum_{t=1}^{n}\chi(B|\boldsymbol{z}_t), \forall B \in \mathcal{B}(\mathcal{Z})$. The probability mass of $p(\cdot)$ is concentrated in regions with many training samples, and generalizing to test samples far away from the training set requires therefore large variances σ_k. On the other hand, large variances force the same output upon any set of training samples that are close to each other. This can unintentionally smooth out rough functions. k-NN, for example, approaches this problem by averaging the labels of the k nearest neighbors, irrespective of the distance. To achieve a similar effect, one can *scale* the virtual variances σ_k by the distance to the k nearest neighbors. This is an example of an UNCERTAINTY MEASURE, that allows generalization to sparsely samples regions of $\mathcal{Z}$, while maintaining the possibility of detailed functions in densly sampled regions. I proposed the inverse of ζ's *probability density function* as uncertainty measure, that is, the *Radon-Nikodym derivative*[13] $\frac{d\vartheta}{d\zeta}$ between the uniform distribution ϑ and the training distribution ζ (see Figure 3.3 and Böhmer and Obermayer, 2015, in Appendix B.2 on Page 216). Scaling the virtual variance ensures therefore strong regularization in sparsely sampled regions and weak regularization in strongly sampled regions. Moreover, in the limit of infinite training samples, the regularization term becomes $\hat{\mathrm{R}}_k[f] = \frac{1}{2}\|\nabla_k f\|^2_\vartheta$, which is independent of the sample distribution ζ, that is, the training set. For linear functions $f \in \mathcal{F}_\phi$, the solution of Equation 3.19 with this regularization operator is

UNCERTAINTY MEASURE

$$\boldsymbol{a}^* \;=\; \big(\hat{\mathrm{E}}[\boldsymbol{\phi}\cdot\boldsymbol{\phi}^\top] + \sum_{k=1}^{d}\sigma_k^2\,\mathbf{R}_k\big)^{-1}\hat{\mathrm{E}}[\boldsymbol{\phi}\cdot\tilde{r}]\,, \quad \mathbf{R}_k := \langle\nabla_k\boldsymbol{\phi}, \nabla_k\boldsymbol{\phi}^\top\rangle_\vartheta\,. \tag{3.20}$$

For some basis functions $\mathbf{R}_k \in \mathbb{R}^{p\times p}$ can be solved analytically. For example, one-dimensional Fourier cosine bases have $(\mathbf{R}_k)_{ij} = i^2\pi^2\delta_{ij}$, which yields an increasing regularization for parameters of higher frequencies and thus promotes smoothness. In Böhmer and Obermayer (2015), I exploit analytical solutions very effectively, as inner products of factored functions in $L^2(\mathcal{Z},\vartheta)$ reduce to one-dimensional integrals.

3.3 Reinforcement learning algorithms

So far the chapter has introduced the basic concepts of machine learning. In this section I apply these to reinforcement learning problems and derive some standard algorithms from literature.

[13] The Radon-Nikodym derivative $\frac{d\vartheta}{d\zeta} : \mathcal{Z} \to [0,\infty]$ is a function that "translates" between measures ϑ and ζ, i.e. $\int \zeta(dz)\,\frac{d\vartheta}{d\zeta}(z)\,f(z) = \int \vartheta(dz)\,f(z), \forall f \in L^2(\mathcal{Z},\zeta)$. The derivative is unique and exists iff ϑ is *absolutely continuous* w.r.t. ζ, i.e. $\zeta(dz) = 0 \Rightarrow \vartheta(dz) = 0, \forall dz \in \mathcal{B}(\mathcal{Z})$. For the uniform distribution ϑ, this is equivalent to the assumption that in the limit of infinite samples, each sample $z \in \mathcal{Z}$ will be drawn by ζ eventually. This is for example the case for an ergodic Markov chain with a strictly non-deterministic policy.

Algorithm 1 Model-based RL for non-stationarity finite state spaces

```
// Initialize current state x', Q-value q and models r and P
while alive do
  // Draw next sample from the environment
  x = x';   a ~ softmax(q_(index(x,:)));
  [x', r'] = sample_environment(x, a);
  // Update models r ∈ R^|X| and P ∈ R^(|X||A|×|X|)
  z = index(x, a);
  r_(z) = (1 − α) r_(z) + α r';
  P_(z,:) = (1 − α) P_(z,:);   P_(z,x') = P_(z,x') + α;
  // Perform policy iteration with the updated model
  v' = ∞;
  while ||v − v'||_∞ > ε do
    q = r + γPv;   v' = v;
    for i in {1, ..., |X|} do
      z = index(i, argmax(q_(index(i,:))));
      r^π_(i) = r_(z);   P^π_(i,:) = P_(z,:);
    end for
    v = (I − γP^π)\r^π;
  end while    // Policy iteration converged
end while      // Endless loop
```

3.3.1 Model based solutions for finite state spaces

The majority of RL publications assumes FINITE STATE SPACES $\mathcal{X} := \{x_1, \ldots, x_{|\mathcal{X}|}\}$. These can in principle be updated as described in Chapter 2. However, there exists an elegant way to express functions $f \in L^\infty(\mathcal{X})$ and linear operators $\hat{\mathrm{A}} : L^\infty(\mathcal{Y}) \to L^\infty(\mathcal{X})$ in terms of estimates in $L^2(\mathcal{X}, \xi)$:

$$x_i \equiv \boldsymbol{x}^i \in \mathbb{R}^{|\mathcal{X}|}, \quad \text{with } x^i_j := \delta_{ij}, \quad \forall x_i \in \mathcal{X} \tag{3.21}$$

$$f \equiv \boldsymbol{f} \in \mathbb{R}^{|\mathcal{X}|}, \quad \text{with } f(x) := \boldsymbol{f}^\top \boldsymbol{x}, \quad \forall f \in L^\infty(\mathcal{X}) \tag{3.22}$$

$$\hat{\mathrm{A}} \equiv \mathbf{A} \in \mathbb{R}^{|\mathcal{Y}|\times|\mathcal{X}|}, \text{ with } \hat{\mathrm{A}}[f](x) := \boldsymbol{f}^\top \mathbf{A} \boldsymbol{x}, \ \forall \hat{\mathrm{A}} : L^\infty(\mathcal{Y}) \to L^\infty(\mathcal{X}) \tag{3.23}$$

Estimation of a function, like the mean reward function $r(x, a) = r(z) \in L^\infty(\mathcal{Z})$, is equivalent to OLS (Equation 3.11) with the linear function $r(z) = \boldsymbol{r}^\top \boldsymbol{z}$:

$$\boldsymbol{r} = \mathbf{C}^{-1}\boldsymbol{b}, \quad \text{with} \quad \mathbf{C} := \sum_{t=1}^{n} \boldsymbol{z}_t \boldsymbol{z}_t^\top \quad \text{and} \quad \boldsymbol{b} := \sum_{t=1}^{n} \boldsymbol{z}_t r_t . \tag{3.24}$$

The empirical covariance matrix $\mathbf{C}$ is diagonal and represents how often a state-action pair has been visited. OLS is thus equivalent to batch learning as presented in Proposition 2.4 on Page 14. One can apply this principle directly to value iteration in Equation 2.11. Minimizing the OLS cost function of the value iteration operator $\boldsymbol{v}' = \hat{\Gamma}_\pi[\hat{\mathrm{B}}[\boldsymbol{v}]]$ applied on the previous value function $\boldsymbol{v} \in \mathbb{R}^{|\mathcal{X}|}$ yields

$$\boldsymbol{v}' = \hat{\Gamma}_\pi[\boldsymbol{r}] + \gamma \hat{\Gamma}_\pi[\mathbf{P}]\boldsymbol{v}, \quad \text{with} \quad \mathbf{P} := \mathbf{C}^{-1}\hat{\mathrm{E}}[\boldsymbol{z}_t \boldsymbol{x}_{t+1}^\top] \in \mathbb{R}^{|\mathcal{X}||\mathcal{A}|\times|\mathcal{X}|}. \tag{3.25}$$

Note that this requires to learn an explicit *model* of reward function and transition operator, and is called MODEL-BASED RL (Sutton and Barto, 1998). In non-stationary environment, one can also employ online-learning (Equation 2.10): MODEL-BASED RL

$$\boldsymbol{r} \leftarrow \boldsymbol{r} + \alpha_t\Big(\boldsymbol{z}_t\, r_t - \boldsymbol{z}_t\boldsymbol{z}_t^\top \boldsymbol{r}\Big) \quad \text{and} \quad \mathbf{P} \leftarrow \mathbf{P} + \alpha_t\Big(\boldsymbol{z}_t\boldsymbol{x}_{t+1}^\top - \boldsymbol{z}_t\boldsymbol{z}_t^\top \mathbf{P}\Big)\,. \tag{3.26}$$

The estimated models will oscillate around the true function/operator.

As the current policy π is known, it is easy to calculate $\boldsymbol{r}^\pi := \hat{\Gamma}_\pi[\boldsymbol{r}] \in \mathbb{R}^{|\mathcal{X}|}$ and $\mathbf{P}^\pi := \hat{\Gamma}_\pi[\mathbf{P}] \in \mathbb{R}^{|\mathcal{X}|\times|\mathcal{X}|}$. This allows to calculate the fix point $\boldsymbol{v}^* = (\mathbf{I} - \gamma\mathbf{P}^\pi)^{-1}\boldsymbol{r}^\pi$ by rearranging the equation and inverting the matrix $(\mathbf{I} - \gamma\mathbf{P}^\pi) \in \mathbb{R}^{|\mathcal{X}|\times|\mathcal{X}|}$, as demanded in Proposition 2.2. The fix-point of Q-learning (Equation 2.14) can not be computed analytically, as the operator $\hat{\mathrm{B}}[\hat{\Gamma}_*]$ is not linear. Given fix-point $\boldsymbol{v}^*$, however, the policy can be greedily improved by

$$\pi'(a|x) \;=\; \delta_{aa^*}\,, \qquad \text{with} \quad a^* \;\in\; \operatorname*{arg\,sup}_{a'\in\mathcal{A}} q(x,a')\,, \quad \boldsymbol{q} \;=\; \boldsymbol{r} + \gamma\mathbf{P}\boldsymbol{v}^*\,. \tag{3.27}$$

The improved policy changes the *rows* selected from $\boldsymbol{r}$ and $\mathbf{P}$ to generate next iteration's $\boldsymbol{r}^{\pi'}$ and $\mathbf{P}^{\pi'}$. The complete procedure is given in Algorithm 1. Note that there are other ways to update the Q-value function that do not require a matrix inversion (e.g. simulated sampling in Dyna-Q Sutton, 1990).

3.3.2 Linear RL algorithms

Most of literature is dealing with finite state spaces. This reflects more the solvability than the applicability to real-world problems, though. Many applications of interest, like robotics, facility control, finances and many more, require continuous state spaces. This thesis focuses therefore on continuous state spaces. More specific, I primarily discuss linear functions $f(\cdot) := \boldsymbol{a}^\top\boldsymbol{\phi}(\cdot) \in \mathcal{F}_\phi$ of p basis functions $\phi_i \in L^2(\mathcal{Z},\zeta)$, with OLS (see Equation 3.11) as the estimation method.

Linear functions can be updated online (in many algorithmic flavours, e.g. Sutton and Barto, 1998; Sutton et al., 2009), but I will concentrate here on batch learning, which is more sample efficient (see Section 2.2.3). There exist a host of algorithms for linear functions, like *Bellman residual minimization* (BRM, Schweitzer and Seidmann, 1985), *Gaussian-process temporal-difference learning* (GPTD, Engel et al., 2003), *fixed-point Kalman filtering* (FPKF, Choi and Roy, 2006) and *Kalman temporal-difference learning* (KTD Geist and Pietquin, 2010). However, I restrict myself here to the discussion of the most well known algorithm in this class,

LSTD namely LEAST SQUARES TEMPORAL DIFFERENCE LEARNING (LSTD, Bradtke and Barto, 1996). As in the previous section, I start[14] with an OLS estimation of the value or Q-value iteration operator, $\hat{\Gamma}_\pi[\hat{\mathrm{B}}[\cdot]]$ or $\hat{\mathrm{B}}[\hat{\Gamma}_\pi[\cdot]]$, onto an existing (Q-)value function $v(z) = \boldsymbol{a}'^\top\boldsymbol{\phi}(z) \in \mathcal{F}_\phi$.

$$\begin{aligned} &\inf_{f\in\mathcal{F}_\phi} \tfrac{1}{2}\sum_{t=1}^n \big(f(z_t) - r_t - \gamma\, v(z_{t+1})\big)^2 \qquad \Rightarrow \qquad \boldsymbol{a} = \mathbf{C}^{-1}\big(\boldsymbol{b} + \gamma\mathbf{D}\boldsymbol{a}'\big)\,, \\ &\text{with}\;\; \mathbf{C} := \sum_{t=1}^n \boldsymbol{\phi}(z_t)\,\boldsymbol{\phi}(z_t)^\top\,,\;\; \mathbf{D} := \sum_{t=1}^n \boldsymbol{\phi}(z_t)\,\boldsymbol{\phi}(z_{t+1})^\top\,,\;\; \boldsymbol{b} := \sum_{t=1}^n \boldsymbol{\phi}(z_t)\, r_t\,. \end{aligned} \tag{3.28}$$

[14] Minimizing the least-squares error $\|f - \hat{\mathrm{B}}[\hat{\Gamma}_\pi[f]]\|^2_{\xi_\pi}$ directly (as in BRM, Schweitzer and Seidmann, 1985) introduces a *double sampling bias* (Baird, 1995; Dann et al., 2014). LSTD is designed to avoid this with the method of *instrumental variables* (Bradtke and Barto, 1996). However, the presented derivation tends more towards the methods presented in the book of Bertsekas (2007), cast into the thesis' mathematical framework.

An online value iteration with this operator requires to estimate the matrix $\mathbf{C}^{-1}$ recursively using the *Matrix inversion lemma* and is called *least-squares policy evaluation* (LSPE, Nedić and Bertsekas, 2003). From a theoretical point of view, it is important to notice that, in the limit of infinite samples, the above procedure is equivalent to a *projection* of the value iteration operator onto the feature space spanned by the basis functions. In other words, one can use the projection operator of Equation 3.12 to define the *projected value- and Q-value iteration operators* $f = \hat{\Pi}_{\xi}^{\phi}[\hat{\Gamma}_{\pi}[\hat{\mathrm{B}}[v]]] \in L^2(\mathcal{X}, \xi)$ and $f = \hat{\Pi}_{\zeta}^{\phi}[\hat{\mathrm{B}}[\hat{\Gamma}_{\pi}[v]]] \in L^2(\mathcal{X} \times \mathcal{A}, \xi\pi)$, respectively.

Proposition 3.6 (see e.g. Bertsekas, 2007) *Both projected (Q-)value iteration operators* $\hat{\Pi}_{\xi}^{\phi}[\hat{\Gamma}_{\pi}[\hat{\mathrm{B}}[\cdot]]]$ *and* $\hat{\Pi}_{\xi\pi}^{\phi}[\hat{\mathrm{B}}[\hat{\Gamma}_{\pi}[\cdot]]]$ *are for* $\gamma < 1$ *contraction mappings under the norms of* $L^2(\mathcal{X}, \xi)$ *and* $L^2(\mathcal{X} \times \mathcal{A}, \xi\pi)$*, respectively.*

Proof: can be found in Appendix C.3 on Page 238. □

This projected operator is a contraction mapping, as shown in Proposition 3.6, which implies that the fix-point $f^*(\cdot) = \boldsymbol{a}^{*\top}\boldsymbol{\phi}(\cdot)$ exist and can be found by setting $\boldsymbol{a} = \boldsymbol{a}'$:

$$\boldsymbol{a}^* = (\mathbf{C} - \gamma\mathbf{D})^{-1}\boldsymbol{b} = (\mathbf{I} - \gamma\mathbf{P}^{\pi})^{-1}\boldsymbol{r}^{\pi}\,, \quad \text{with} \quad \boldsymbol{r}^{\pi} := \mathbf{C}^{-1}\boldsymbol{b}\,, \quad \mathbf{P}^{\pi} := \mathbf{C}^{-1}\mathbf{D}\,. \tag{3.29}$$

This is called the least-squares temporal difference (LSTD) solution. Note that the MODEL-FREE batch LSTD estimate of $\boldsymbol{a}^*$ (first equality) is also the model-based solution (second equality). LSTD solutions are therefore exactly equivalent to the fix-point of linear models based on the same training data (Parr et al., 2008). MODEL-FREE RL

Proposition 3.7 (from Tsitsiklis and Van Roy, 1997) *The difference between the LSTD fix-point solution* $f \in \mathcal{F}_{\phi}$ *and the true fix point* $v = \hat{\Gamma}_{\pi}[\hat{\mathrm{B}}[v]] \in L^2(\mathcal{X}, \xi)$ *or* $v = \hat{\mathrm{B}}[\hat{\Gamma}_{\pi}[v]] \in L^2(\mathcal{X} \times \mathcal{A}, \xi\pi)$*, with* $\hat{\mathrm{B}}[\cdot] := r + \gamma\hat{\mathrm{P}}[\cdot]$ *and* ξ *as the steady state distribution of* P *with policy* π*, is bounded from above by*

$$\left\|v - f^*\right\|_{\zeta} \quad \leq \quad \frac{1}{\sqrt{1-\gamma^2}}\left\|v - \hat{\Pi}_{\zeta}^{\phi}[v]\right\|_{\zeta}\,, \qquad \textit{with} \quad \zeta = \xi \quad \textit{or} \quad \zeta = \xi\pi\,.$$

Proof: can be found in Appendix C.3 on Page 239. □

Propositions 3.6 and 3.7 hold technically only in the limit of infinite training samples, but provide important insights into the convergence behavior of LSTD. In Section 4.2.3, for example, I use the bound in Proposition 3.7 as a cost function to learn the basis functions $\{\phi_i\}_{i=1}^{p}$ from training data.

Using LSTD to estimate Q-values allows *off-policy* learning (see Section 2.2.4). Here one uses training samples drawn from a Markov chain, which executes sampling policy π, to evaluate another (possibly improved) policy π' with the projected Q-value iteration operator $\hat{\Pi}_{\xi\pi}^{\phi}[\hat{\mathrm{B}}[\hat{\Gamma}_{\pi'}[\cdot]]]$. Both policies affect Equations 3.28 and 3.29 only in their choice of actions: the sampling policy π draws a_t and the evaluated policy π' draws a'_{t+1}. Using transitions $z_t := (x_t, a_t) \to (x_{t+1}, a'_{t+1}) =: z_{t+1}, a'_{t+1} \sim \pi'(\cdot|x_{t+1})$, only matrix $\mathbf{D}$ has to be recomputed to find the off-policy LSTD solution. LEAST-SQUARES POLICY ITERATION (LSPI, Lagoudakis and Parr, 2003, Algorithm 2 on Page 36 with `soft_lspi=false`) repeatedly estimates the next Q-value $q^{i+1} \in \mathcal{F}_{\phi}$ with LSTD, based on a fixed training set of transitions $\mathcal{D} = \{(x_t, a_t) \to x'_t, r_t\}_{t=1}^{n}$, by adjusting $a'_{t+1} := \arg\max_{a'} q^i(x_{t+1}, a')$. As for most linear algorithms, there exists also a kernelized version of LSPI (KLSPI, Xu et al., 2007). LSPI

Algorithm 2 Least-squares policy iteration (LSPI, Lagoudakis and Parr, 2003)

Input: $\mathbf{X} \in \mathbb{R}^{d_x \times n}$, $\mathbf{A} \in \mathbb{R}^{d_a \times n}$, $\boldsymbol{r} \in \mathbb{R}^{n \times 1}$, $\mathbf{X}' \in \mathbb{R}^{d_x \times n}$, $\gamma \in [0, 1)$, $\beta \in [0, \infty)$
$\mathbf{C} = \boldsymbol{\phi}(\mathbf{X}, \mathbf{A})\, \boldsymbol{\phi}(\mathbf{X}, \mathbf{A})^\top \in \mathbb{R}^{p \times p}$; // covariance matrix $\mathbf{C}$
$\boldsymbol{b} = \boldsymbol{\phi}(\mathbf{X}, \mathbf{A})\, \boldsymbol{r} \in \mathbb{R}^{p \times 1}$; // reward statistics $\boldsymbol{b}$
$\mathbf{Q} = \mathtt{rand}(|\mathcal{A}|, n) \in \mathbb{R}^{|\mathcal{A}| \times n}$; $\mathbf{Q}' = \infty$; // Q-value matrix $\mathbf{Q}$
while $\|\mathbf{Q} - \mathbf{Q}'\|_\infty > \epsilon$ **do**
  $\mathbf{Q}' = \mathbf{Q}$; $\mathbf{D} = \mathbf{0} \in \mathbb{R}^{p \times p}$;
  if ¬(soft_lspi) **then**
    $\mathbf{D} = \boldsymbol{\phi}(\mathbf{X}, \mathbf{A})\, \boldsymbol{\phi}(\mathbf{X}', \mathtt{argmax}(\mathbf{Q}))^\top$; // as in Lagoudakis and Parr (2003)
  else
    $\boldsymbol{\pi}' = \mathtt{softmax}(\mathtt{normalize}(\mathbf{Q}), \beta) \in \mathbb{R}^{|\mathcal{A}| \times n}$; // see Eqns. 2.16 &2.17
    for $k \in \{1, \ldots, |\mathcal{A}|\}$ **do**
      $\mathbf{D} = \mathbf{D} + \boldsymbol{\phi}(\mathbf{X}, \mathbf{A})\, [\boldsymbol{\phi}(\mathbf{X}', k) \cdot (\mathbf{1}\, \boldsymbol{\pi}'_{(k,:)})]^\top$; // soft-LSPI extension
    end for
  end if
  $\boldsymbol{a} = (\mathbf{C} - \gamma \mathbf{D}) \backslash \boldsymbol{b}$; // LSTD parameter-vector $\boldsymbol{a}$
  for $k \in \{1, \ldots, |\mathcal{A}|\}$ **do**
    $\mathbf{Q}_{(k,:)} = \boldsymbol{a}^\top \boldsymbol{\phi}(\mathbf{X}', k)$; // Q-value matrix $\mathbf{Q}$ updated
  end for
end while
Output: $\boldsymbol{a} \in \mathbb{R}^{p \times 1}$

3.3.3 Empirical convergence analysis

LSPI works in many tasks, but no convergence guarantees[15] exist (Dann et al., 2014). The critical parameter appears to be γ. Figure 3.4 demonstrates this at the example of two navigation tasks learned with 1500 bases, encoding the 3-dimensional continuous state space and 3 discrete actions[16]. The plots evaluate the policies learned by LSPI (black crosses) in a U-shaped room (left plot) and in a S-shaped room (right plot). Solving the tasks requires to plan 40-70 steps ahead, that is, requires a large discount factor γ. Both plots show that low γ do not solve the task,

DIVERGENCE

however, large γ are often DIVERGENT ($0.938 < \gamma < 0.996$ on the left and $\gamma > 0.938$ on the right). The exact underlying mechanisms are still unknown, but there are two potential sources of this error: value estimation or policy improvement.

[15] Dann et al. (2014) suggest to use *importance reweighting* of the training samples with Radon-Nikodym derivative $\frac{d\pi'}{d\pi}$ to change the effective sampling distribution, based on the idea of *importance sampling* (Glynn and Iglehart, 1989). In combination with greedy policies, this is equivalent to discarding all training transitions that do not agree with the evaluated policy. While reported to improve convergence in LSPI, reweighting can therefore be impractical for tasks with many actions. Additionally, without knowledge of the steady state distribution of the evaluated policy π' (which may not even exist), Lemma C.2 does not hold and convergence can not be guaranteed. Note that a softmax policy improvement could use importance reweighting much better.

[16] The 1500 bases encode all combinations of: 10 Fourier cosine bases for each spatial coordinate x and y, 5 full Fourier bases (containing both sine and cosine bases) for the agent's orientation θ and 3 *Kronecker-delta* functions for the action a. For $x, y, \theta \in [0, 1]$ and indices beginning at 0:

$$\phi_{ijlk}(x, y, \theta, a) \quad := \quad \cos(i\pi x)\, \cos(j\pi y)\, \delta_{ka} \begin{cases} \cos(l\pi\theta) & \text{, for even } l \\ \sin((l{+}1)\pi\theta) & \text{, for odd } l \end{cases}.$$

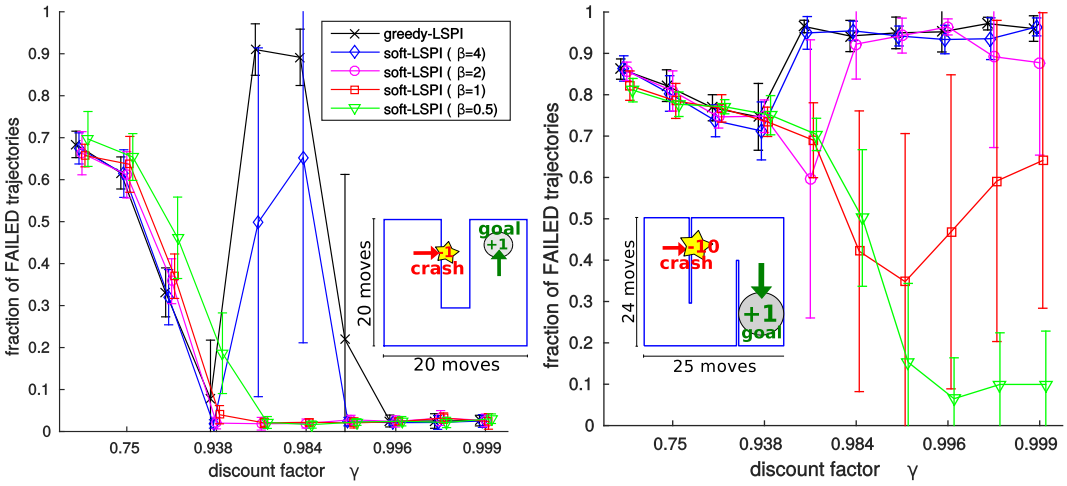


Figure 3.4: Greedy- and softmax-policies for LSPI, compared in two continuous navigation task (see sketched room layouts). The agent can move forward or rotate 45° left or right (i.e. 3 actions). Reaching the goal area is rewarded (+1) and crashing into a wall is punished (-1 or -10). Error-bars indicate mean and standard deviation (over 10 random-walk training sets with 50000 samples each) of the *fraction of failed test-trajectories*, starting at random positions, for different *discount factors* γ.

The first possibility is that LSTD does not yield a good value estimate. As the projection operator $\hat{\Pi}^{\phi}_{\xi}$ is not using the steady state distribution ξ' of the evaluated policy π', the projected value iteration operator may either not be convergent or converges to some arbitrary local minimum. The idea can be demonstrated by rewriting the contraction mapping property of $\hat{\Pi}^{\phi}_{\xi\pi}[\hat{\mathrm{B}}[\hat{\Gamma}_{\pi'}[\cdot]]]$ in terms of operator norms:

$$\left\|\hat{\Pi}^{\phi}_{\xi\pi}[\hat{\mathrm{B}}[\hat{\Gamma}_{\pi'}[f]]] - \hat{\Pi}^{\phi}_{\xi\pi}[\hat{\mathrm{B}}[\hat{\Gamma}_{\pi'}[g]]]\right\|_{\xi\pi} \quad \leq \quad \gamma \underbrace{\left\|\hat{\Pi}^{\phi}_{\xi\pi}\right\|_{\xi\pi}}_{=1} \underbrace{\left\|\hat{\mathrm{P}}[\hat{\Gamma}_{\pi'}]\right\|_{\xi\pi}}_{\leq \|\hat{\Gamma}_{\pi'}\|_{\xi}} \|f-g\|_{\xi\pi}\,. \tag{3.30}$$

As long as for all encountered policies π' the discount factor $\gamma < \|\hat{\Gamma}_{\pi'}\|_{\xi}^{-1}$, LSPI should converge to (or near) the same solution. However, for almost all practical applications[17] this bound is far too restrictive, and the empirical results in Figure 3.4 demonstrate that stable solutions exist for much larger γ.

By ensuring convergence of LSTD, one should therefore also ensure convergence of LSPI. I investigated a PROJECTION OPERATOR $\hat{\mathrm{G}}[f]$, which translates f into the eigensystem of operator $\hat{\Pi}^{\phi}_{\xi\pi}[\hat{\mathrm{P}}[\hat{\Gamma}_{\pi'}[\cdot]]]$ to effectively truncate the eigenvalues of that operator at $\frac{1}{\gamma}$: CONVERGENT PROJECTIONS

$$\begin{aligned} \hat{\mathrm{G}}[f] &= \langle f, \boldsymbol{\phi}\rangle_{\xi\pi}\langle \boldsymbol{\phi},\boldsymbol{\phi}\rangle_{\xi\pi}^{-1}\mathbf{G}\,\boldsymbol{\phi}(\boldsymbol{z})\,, \qquad \text{with} \qquad \mathbf{G} = \mathbf{U}\bar{\mathbf{\Lambda}}\mathbf{U}^{\top}\,, \\ \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\top} &:= \mathbf{D}^{\top}\mathbf{C}^{-1}\mathbf{D}\,, \quad \bar{\Lambda}_{ii} := \begin{cases} \frac{1}{\gamma\Lambda_{ii}} & \text{, if } \Lambda_{ii} > \frac{1}{\gamma} \\ 1 & \text{, otherwise} \end{cases} \text{, } \forall i\,. \end{aligned} \tag{3.31}$$

In theory, this ensures that the projected Q-value iteration operator $\hat{\Pi}^{\phi}_{\xi\pi}[\hat{\mathrm{B}}[\hat{\Gamma}_{\pi'}[\hat{\mathrm{G}}[\cdot]]]]$ is a contraction mapping. However, I was unable to confirm *any* changes in the

[17] Take the example in Figure 3.4, with 3 discrete actions sampled by a uniform policy π. The operator norm of *every* greedy policy π' is $\gamma < \|\hat{\Gamma}_{\pi'}\|_{\xi}^{-1} \leq \frac{1}{3}$ (the Radon-Nikodym derivative $\frac{d\pi'}{d\pi}$ is 3 for one action and 0 otherwise), which is not large enough for any non-trivial navigation task.

behavior of LSPI using the resulting LSTD variant $\boldsymbol{a}^* = (\mathbf{C} - \gamma\mathbf{DG})^{-1}\boldsymbol{b}$ in experiments. LSTD does therefore *not* appear to be the cause of LSPI divergence. With this in mind, I did not further investigate other existing projection methods, e.g., based on L_∞ projections (Guestrin et al., 2001a; Petrik and Zilberstein, 2011).

The second possibility for divergence is the greedy policy improvement step or at least an interaction thereof with value estimation. Perkins and Precup (2002) suggest that with a slowly changing non-deterministic policy improvement, policy iteration is a contraction mapping and thus guaranteed to converge[18]. I have tested their claim by using a softmax policy with slowly increasing inverse temperature β, similar to SIMULATED ANNEALING (Haykin, 1998). Empirically this increase has to be very slow and LSPI took an unacceptable number of iterations to converge. However, I have observed that even prematurely aborted policy iteration yields improved results.

SOFT-LSPI

In fact, the same result can be achieved by running LSPI with the inverse temperature at which the annealing aborts. I call this approach SOFT-LSPI (Algorithm 2 with `soft_lspi=true`). Here the stochastic policy is obtained by a softmax with inverse temperature β (Equation 2.16) based on normalized Q-values (Equation 2.17). The algorithm converges as fast as greedy LSPI, but Figure 3.4 shows that divergence can be greatly reduced by low inverse temperatures β. However, the parameter β must be chosen carefully, as small β also lower the performance for a given discount factor γ (e.g. $\beta = 0.5$, green triangles). In contrast, large β behave almost like greedy LSPI (e.g. $\beta = 4$, blue diamonds). Soft-LSPI extends therefore the abilities of greedy LSPI (as defined by Lagoudakis and Parr, 2003), by introducing another hyper-parameter β, that has to be chosen by the system-designer before training.

I conclude that the divergent behavior of LSPI is most likely caused by the policy improvement step, rather than the value estimation with LSTD. However, the problem can be strongly mitigated by soft-LSPI. A theoretical justification is still missing, as well as an autonomous adaptation of β to test the predictions of Perkins and Precup (2002) in feasible time.

3.3.4 Non-linear RL algorithms

This thesis aims to analyze and learn state(-action) representations for reinforcement learning in chapters 4 and 5. For the above discussed linear (Q-)value functions, state representation can simply be seen as basis function (a.k.a. feature-)selection and I will primarily focus on this interpretation. However, algorithms that learn non-linear functions have obviously other requirements for the state representation. In fact, the literature I reviewed appears to have a strong focus on deep neural networks to learn state representations for non-linear RL algorithms (Böhmer et al., 2015). For completeness sake, I want to introduce the basis of RL with artificial neural networks (ANN, Haykin, 1998).

NEURAL NETWORKS

An ANN is a set of linear functions, called NEURONS. Each neuron $h_j(z) := \sum_{i\in\rho_j} w_{ij}\, f(h_i(z))$ is connected to a set of parents $i \in \rho_j$ with a WEIGHT $w_{ij} \in \mathbb{R}$ and a TRANSFER FUNCTION $f : \mathbb{R} \to \mathbb{R}$. Each input dimension is modeled as a parent-less neuron, and one neuron h_{out} represents the ANN output. Recursive dependencies between neurons yields complicated behavior and optimization. The class of FEED-FORWARD ANN do therefore not allow cycles in the resulting dependency graph. This simplification allows fast FORWARD computation of the ANN's

[18] They attempt to show this for a batch value estimation algorithm closely related to LSTD.

output and fast BACK-PROPAGATION of the online gradient. Let in the following $E : \mathcal{Z} \times \mathbb{R} \to \mathbb{R}$ be a (e.g. quadratic) cost function of sample z_t and label y_t. The gradient of $E(z_t, y_t)$ w.r.t. the parameter w_{ij} of neuron h_j is

BACK-PROPAGATION

$$\frac{\partial E(z_t,y_t)}{\partial w_{ij}} = \frac{\partial E(z_t,y_t)}{\partial h_j(z_t)} f(h_i(z_t)) , \quad \frac{\partial E(z_t,y_t)}{\partial h_j(z_t)} = \sum_{j \in \rho_k} \frac{\partial E(z_t,y_t)}{\partial h_k(z_t)} w_{jk} f'(h_j(z_t)) . \quad (3.32)$$

The forward-sweep computes all neurons' outputs $h_i(z_t)$ of input sample z_t. For a quadratic cost function, the prediction error is $\frac{\partial E(z_t,y_t)}{\partial h_{\text{out}}(z_t)} = h_{\text{out}}(z_t) - y_t$. This error is passed to all parents h_j of h_{out}, which compute the local error $\frac{\partial E(z_t,y_t)}{\partial h_j(z_t)}$, as described in Equation 3.32, and pass it on to their respective parents. After the back-propagation has reached the input layer, all weights w_{ij} are updated by GRADIENT DESCEND, that is, $w_{ij} \leftarrow w_{ij} - \alpha_t \frac{\partial E(z_t,y_t)}{\partial w_{ij}}$.

GRADIENT DESCEND

Many textbooks propose SIGMOID transfer-functions like the *hyperbolic tangent* (Haykin, 1998; Bishop, 2006). However, in practice these can have very shallow derivatives f'. Passing these on to parents causes VANISHING GRADIENTS (Hochreiter et al., 2001) and prevents efficient training of deep networks. Recently many authors appear to have solved this problem by using the transfer function $f(z) = \max(0, z)$, though. However, any non-linear transfer function induces a non-convex cost-function with many local minima. For a long time this practically restricted ANN to shallow networks. These are still used (e.g. Jonschkowski and Brock, 2014) but get more and more replaced by DEEP NEURAL NETWORKS (Hinton and Osindero, 2006). Here a deep layered neural network's weights are *initialized* by training a layer-wise AUTO-ENCODER. Each "layer" uses the previous layers' n *hidden neurons* as input neurons h^{i}_k, which are fully connected to this layer's set of m hidden neurons $h^{\text{h}}_j(\boldsymbol{z}) = \sum_{k=1}^{n} w^{\text{h}}_{kj} f(h^{\text{i}}_k(z))$. These hidden neurons are the (fully connected) parents of the layer's n output neurons $h^{\text{o}}_i(\boldsymbol{z}) = \sum_{j=1}^{m} w^{\text{o}}_{ji} f(h^{\text{h}}_j(z))$. By constraining the number of hidden neurons to be smaller than the number of inputs, $m \ll n$, training the network to predict its own inputs leads to an efficient coding of that input in the hidden layer. Hidden and output weights are often tied together:

VANISHING GRADIENTS

DEEP NEURAL NETWORKS

AUTO-ENCODER

$$\mathcal{C}_{\text{ae}}(\boldsymbol{z}) = \tfrac{1}{2} \sum_{i=1}^{n} \left(h^{\text{o}}_i(\boldsymbol{z}) - z_i \right)^2 \quad \text{s.t.} \quad w^{\text{h}}_{ij} = w^{\text{o}}_{ji} , \quad \forall i, j . \quad (3.33)$$

After training of one layer has converged, the output neurons are discarded and the next (smaller) layer is trained with the hidden neurons as inputs. Each layer compresses the input samples further until some sufficiently *low-dimensional representation* is reached. Using this deep network as initialization for traditional back-propagation appears to avoid most local minima and has by now broken most benchmark records previously held by support vector machines (SVM, Schölkopf and Smola, 2001).

Authors that applied ANN methods to reinforcement learning (e.g. Mnih et al., 2015), report an additional problem with online-updates. Updating one state-action pair at a time appears to cause unpredictable changes in other, sometimes completely unrelated states. Online learning still converges, but requires an unfeasible amount of samples. Riedmiller (2005) has therefore proposed an off-policy batch Q-value iteration approach based on ANN called NEURAL FITTED Q-LEARNING (NFQ). The algorithm treats the Q-value iteration like a regression problem and learns one application of the Q-value iteration operator (Equation 2.12) with an efficient batch gradient-descend procedure (RPROP, Riedmiller and Braun, 1993). Using batch

NFQ

learning to average the gradients of the entire data set appears to stabilize the optimization and allows efficient reuse of observed transitions. However, due to the highly nonlinear nature of neural networks, no convergence guarantees exist for NFQ.

Other approaches to reinforcement learning with neural networks exist, like *actor-critic* algorithms (Konda and Tsitsiklis, 2003) and *direct policy search* (Deisenroth et al., 2013), but a comprehensive review is beyond the scope of this thesis.

3.4 Challenges to machine learning

Most of the recent successes in artificial intelligence are based on machine learning techniques, and the field seems to advance at a rapid pace. Only a decade ago the most sophisticated ML techniques were based on kernel methods, in particular support vector machines. Nowadays these are getting outcompeted in nearly every field by deep neural networks. Many colleagues I talked to during my PhD studies expressed their believe that we only have to scale up the existing methods, primarily by providing more samples, to learn almost anything. Despite impressive successes in recent years, however, I believe that the construction of truly autonomous adaptive systems requires us to challenge some of the basic assumptions underlying ML.

3.4.1 Correcting deductive errors

In many practical experiments with deductive models or batch learning algorithms, I observed some irritating oscillatory behavior. For example, Figure 3.5 shows some test-trajectories of a robot in a visual navigation task (Böhmer et al., 2013). The numbers indicate how many actions were executed to reach the gray goal area in the lower right corner. Some of their differences are due to different starting orientations, which may require to rotate the robot first, but note the large number of actions and unsuccessful trajectories on the left half of the figure. These are due to

OSCILLATIONS

OSCILLATIONS, which are caused by incorrect rotations: after a faulty rotation, the robot will rotate (correctly) back to (or near to) the original state and repeat the incorrect decision. In rare occasions, oscillations can also occur between three or more states. The only way to escape such a cycle is to be disrupted by exploration (e.g. a softmax) or system-inherent noise.

This phenomenon usually occurs far away from the goal and at locations where the largest Q-values are similar. Here small estimation errors can change the greedy policy and cause oscillations. Online estimation in finite state spaces can simply

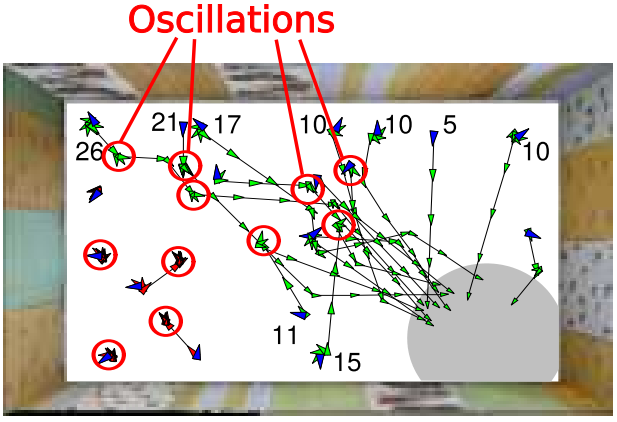


Figure 3.5: Modified result figure from Böhmer et al. (2013). A robot (green markers) navigates from blue staring positions to the gray goal area. The numbers indicate the required actions. Some are unnecessary high due to oscillations (marked with red circles). Oscillations are caused by incorrect rotations and can only be escaped by system noise.

update the Q-values and eventually escape the cycle. In continuous state spaces, however, the value function must be approximated, which induces unavoidable estimation errors. Model based and batch approaches are more sample-efficient (and often equivalent deductive mechanisms), but lack a mechanism to correct themselves. In Section 5.1.1, I derive a method to combine deductive and inductive Q-value estimation and eventually escape these oscillations. Alternatively, one could aim to find the *decision boundaries* induced by representation and policy. It might be possible to derive an IMPORTANCE SAMPLING approach (Glynn and Iglehart, 1989), that compensates by introducing additional punishment, or one could try to adjust the inverse temperature β or the discount factors of critical states.

Last but not least, it would be a great help if one could modify the *representation* at critical states, for example, by including a Gaussian basis function centered around it. Other approaches from literature select a candidate basis either during training (by sparse L_1 regularization, Kolter and Ng, 2009; Johns et al., 2010), or iteratively by selecting the basis function that fits the Bellman error best (called MATCHING PURSUIT, Painter-Wakefield and Parr, 2012). The latter selection scheme can be augmented to respect the basis function's spatial derivatives (Wookey and Konidaris, 2015). Chapter 4 discusses the role of representation further.

3.4.2 Structure learning

Most autonomous system will interact with the environment, which prevents the test samples to be drawn from the same distribution as the training samples. This changes the definition of the L_2 generalization error, but more importantly requires predictions in parts of the state space unseen during training. It is my belief that machine learning for autonomous control has to address this problem by STRUCTURE LEARNING to become applicable in practical tasks. I want to illustrate this at the example of taking my three years old nephew to the zoo. If I show my nephew a crudely drawn picture of an animal, he has a pretty good chance to identify it in real life. This is an impossible feat under the standard assumptions of inductive learning. My nephew should see thousands of low-level features (or pixels) in the drawing, each looking very different when compared to the real animal. He must therefore employ prior knowledge to both identify the animal's characteristics in a cartoon and to transfer these characteristics to the features of real-life animals. Engineers, faced with the same problem, have to tweak the sample representation, the regularization and the cost function until an animal can be detected. However, prior knowledge defined this way rarely carries over to other tasks and human intuition often fails to choose it properly in complex tasks. To achieve robust and general AI, prior knowledge must be integrated into ML techniques *autonomously*. But how?

Confident estimates in machine learning require to sample "near" each state multiple times (see Section 3.2.2). When there are not enough samples to cover the entire state space, the unseen states can indirectly be trained using regularization or a more compact state representation. Both techniques depend crucially on a distance metric $d(\cdot,\cdot)$: regularization can be seen as virtual sampling w.r.t. d (see Section 3.2.4) and the choice of basis functions induces constrains w.r.t. d (see Section 3.2.3). My nephew must therefore use his intuition to autonomously choose a suitable similarity/distance function, under which (i) each possible state is "visited" sufficiently often and (ii) states with different targets can be distinguished.

There exist several approaches to formalize and learn such prior knowledge.

METRIC LEARNING

METRIC LEARNING aims to learn a distance function directly from similar and dissimilar examples (Xing et al., 2002). The techniques has been reported to improve subsequent tasks (e.g. Parikh and Grauman, 2011), but suffers from the same restrictions as all inductive learning methods. In opposition to this approach, I am convinced that generalization to unseen regions of the state space requires *deductive* reasoning. This requires to learn structural rules, which may allow to deduce the similarity of unseen states to the training set.

ILP

INDUCTIVE LOGICAL PROGRAMMING (ILP) and STATISTICAL RELATIONAL LEARNING (SRL) address this by learning a symbolic representation of samples, which has a probabilistic component and can be estimated from data (Dietterich et al., 2008). With a proper symbolic description, the learned generalization abilities are unrivaled, but inadequate problem descriptions can lead to drastic misconceptions. Learning adequate symbols and relational predicates, as well as GROUNDING them on sensory observations, is still a largely unresolved problem (not due to a lack of trying, e.g. Coradeschi et al., 2013; Jetchev et al., 2013). To the best of my knowledge, there exist no method that allows to cast observed similarities into a symbolic representation, in order to generalize to unseen samples. In Chapter 5, I investigate how to use FACTORED FUNCTIONS to develop algorithms that solve large, FACTORED MDP (Boutilier et al., 1999; Guestrin et al., 2004) based on a combination of symbolic state description with continuous transition models (see also my publications: Böhmer and Obermayer, 2013, 2015, in the appendix on Pages 209 and 216, respectively).

Chapter 4

Representation for Control

The word "representation" is used ambiguously in control, almost as ambiguous as the word "control" itself. It refers to some kind of intermediate data format that generalizes sensor observations and motor actions. Representation ought to allow precise control as well as generalize well to unseen observations, a challenge that will appear again and again during the course of this chapter. Although other things can be represented as well, in this thesis I focus exclusively on the representation of *states* and *actions*. States and actions are a loosely defined concepts as well (see Section 2.1.4), which are only restricted by stationary transition probabilities, called the MARKOV PROPERTY. Autonomous extraction of states from observations runs the risk of including many distractors and therefore constructing very large state spaces. To obtain a compact state space for planing or learning, the system designer must in practice often choose a task-relevant subspace. This may be the entire the sensor/motor apparatus the agents body can experience, but is often a deliberate design choice that estimates unobservable quantities, for example, estimating a robots location with SLAM (see Page 9) or object dependent SIFT features (Lowe, 1999).

Given a task-relevant state space $\mathcal{X}$, "representation" has a another meaning: as patterns of bits represent numbers in a computer, so must the value function somehow be represented. On the one hand, discrete state spaces are often encoded as vectors of zeros and ones, which span the function space $L^\infty(\mathcal{X})$ (see Section 3.3.1). Continuous spaces, on the other hand, require parameterizable function spaces (see Section 3.1), which are prone to over-fitting. Each function space generalizes values to unseen states w.r.t. an induced metric. In Section 3.2.3 I show this for linear and kernel functions, but the same principle applies to artificial neural networks and other function spaces. To be exact, function spaces only provide constrains on representable functions, further generalization must be enforced by regularization w.r.t. to some metric (which can, but must not, be the same; see Section 3.2.4). When designing an autonomous system, the question arises which function space

REQUIREMENTS FOR GOOD REPRESENTATIONS

- is able to represent all value-functions/policies the agent may need to adopt,
- generalizes reasonably well to unseen states, and
- can prevent over-fitting and divergence with very few hyper-parameters.

Section 4.1 surveys available literature on representation learning and feature selection in RL. I thoroughly analyze basis functions as state representation for linear RL algorithms in Section 4.2. One key result indicates that MDP induce a particular metric on value functions, and that feature spaces learned by *slow feature analysis*

(SFA, Wiskott and Sejnowski, 2002) approximate such a metric. To test this prediction in continuous state spaces, I derive a kernelized slow feature analysis algorithm (RSK-SFA, Böhmer et al., 2012) in Section 4.3. Finally, I evaluate and compare the kernelizable approaches to learn basis functions for LSPI in Section 4.4, and discuss open questions in Section 4.5.

4.1 Representation learning

Learning and planning algorithms are fairly task independent, but success in a given task depends largely on the chosen state representation. System designers have to choose a set of representative *features* ϕ_i from sensor measurements and estimated quantities. Prominent examples are TD-Backgammon (Tesauro, 1994), autonomous helicopter control (Abbeel et al., 2007) and robot soccer (Riedmiller et al., 2009). Well chosen feature spaces Φ have a one-to-one correspondence to the task-relevant subspace $\mathcal{X}$, that is, are ISOMORPHISMS of $\mathcal{X}$. Badly chosen features can either not differentiate between critical states or include so many distractors that both learning and planning becomes infeasible. *Learning* a well suited feature space autonomously from experiences would simplify the design process immeasurably.

4.1.1 Finding task-relevant subspaces

The underlying state spaces $\mathcal{X}$ of practical applications can be very large. Humanoid robots, for example, can have more than 30 degrees of freedom, which usually implies 30 state and 30 action variables. Both planning and learning require (almost) every state and action to be sampled, and autonomous control on $\mathbb{R}^{60}$ is therefore often infeasible. However, these spaces often exhibit task- and environment specific structure. Due to coupled joints and external constrains, the set of possible configurations $\mathcal{X}$ is in practice much smaller, i.e. $\mathcal{X} \subset \mathbb{R}^{60}$. Defining a compact state-representation based on the performed task and current environment can therefore allow to plan robotic control autonomously (e.g., using *topology-based representations*, Ivan et al., 2013). To increase autonomy, many authors aim to learn such a low-dimensional representation of $\mathcal{X}$. Linear dimensionality reduction (like PCA) usually does not capture the complex manifold of states well, as the embedding of $\mathcal{X}$ in $\mathcal{R}^d$ is often highly non-linear. For example, Jenkins and Matarić (2004) learned the manifold directly with ISOMAP (Tenenbaum et al., 2000), Mahadevan and Maggioni (2007) encoded spectral distances within the manifold in a nearest-neighbor extension of PVF and Höfer et al. (2010) used non-linear SFA for the same purpose.

For practical systems with many degrees of freedom, the extracted state spaces $\mathcal{X}$ are still too large for planning or learning, though. However, solving a specific task often requires only a TASK-RELEVANT SUBSPACE $\mathcal{X}_T \subset \mathcal{X}$ (see Page 9). Such a subspace ignores some dimensions of the state-manifold and is thus no longer isomorphic to the observations in configuration space $\mathbb{R}^d$. How to select and ignore these unnecessary state-dimensions is an open question. Jonschkowski and Brock (2014) modified an objective similar to SFA with additional terms to promote predictability of rewards and causality of decisions. The resulting representation of a visual task (see Section 4.1.3) still encoded spectral distances on the manifold, but preferred controllable and rewarded state-dimensions. Although it appears thus possible to learn task-relevant subspaces, doing so with inductive techniques requires samples from the entire state space $\mathcal{X}$. Alternatively, one can also use *demonstrations* of the

TASK-RELEVANT SUBSPACE

intended task, similar to imitation learning in Section 2.3.3. Examples are kernelized manifold encoding (Bitzer et al., 2010) and handcrafted feature spaces (Jetchev and Toussaint, 2014).

Another approach to conquer high-dimensional state spaces $\mathcal{X}$ is to learn general STATE ABSTRACTIONS (Boutilier et al., 1999). One example are *factored MDP*, which assume sparse transition dependencies between state variables (Poupart et al., 2002; Guestrin et al., 2004; Böhmer and Obermayer, 2013). These spaces have the potential for *deductive generalization* and are the focus of Chapter 5. Other authors hope to find large super-structures, similar to *options* in Section 2.3.1, and exploit them either hierarchically or by excluding non-task-relevant states from learning. For example, Foster and Dayan (2002) merge states with similar values in multiple tasks into large abstract states, Bellemare et al. (2013) use recursive quad-trees and Yu et al. (2012) decompose factored MDP into hierarchically controllable options. ABSTRACT STATES

However, this chapter will focus in the following on the construction of basis functions to allow generalization with linear algorithms like LSPI (see Section 3.3.2).

4.1.2 Learning basis functions

Learning generally implies[1] inductive reasoning (see Page 3). An inductively learned representation can only generalize to situations similar to those already experienced in the training set. Learning isomorphic representations Φ of task-relevant state space $\mathcal{X}$ require therefore to sample the *entire* set $\mathcal{X}$ densely. Which states are considered "close by" depends on the metric used for generalization (see Section 3.2.2). Classical machine learning would therefore simply *shift*[2] the burden of designing a feature space from reinforcement learning to representation learning. Given enough samples, this can still improve generalization performance, though. Furthermore, representation learning may be able use data that is not available for control (Böhmer et al., 2015). For example, samples drawn from previous RL tasks in the same environment can not be used to learn a control, as the experienced rewards are no longer valid. Observed state transitions, on the other hand, remain the same and can be used to learn a representation Φ. If this representation generalizes values well, a reliable control can be learned inductively with comparatively few correctly rewarded samples.

The most straight forward approach to state representation is to find a suitable set of basis functions $\{\phi_i\}_{i=1}^p$ and adjust it to the situation at hand. These functions can either be used as bases for linear functions or as feature space Φ for non-linear function classes. Given a compact (usually hand-crafted) state space $\mathcal{X}$, one can either use a non-parametric function class (e.g. a RKHS in Jakab and Csató, 2012; Kveton and Theocharous, 2012) or a parameterizable function class on $\mathcal{X}$. Parameterizable functions are far more common and include, for example, linear CMAC architectures (Sutton, 1996), manifold discretization (Smart, 2004) and trigonometric Fourier bases (Konidaris et al., 2011, which we used for demonstrations in Chapter 3). These function spaces are UNIVERSAL FUNCTION APPROXIMATORS and not task-specific. This can pose a serious problem for linear functions in multi-dimensional

[1] To the best of my knowledge, there exist no deductive technique to learn representations of continuous state-action spaces. This is a promising field and further discussed in Section 4.5.3.

[2] Note the similarities to deep neural networks, which learn representations of the input in each layer. Training with auto-encoders simply shifts the burden of representation to the next layer.

state spaces $\mathcal{X} \subset \mathbb{R}^d$, which suffer a curse of dimensionality: the number of "universal" basis functions ϕ_i grows exponential in the number of state-dimensions d. Many of these functions do not contribute much, and some authors therefore try to select the most useful features only, which is called FEATURE SELECTION. One can either introduce sparsity constrains during optimization (e.g. L_1 regularization, Kolter and Ng, 2009) or select the basis functions greedily, based on their similarity to the current prediction error (Painter-Wakefield and Parr, 2012) and heuristic prior knowledge (e.g. some definition of smoothness, Wookey and Konidaris, 2015). Another possibility is to start with a limited number of flexible basis functions and adjust their parameters to better match the (Q-)value function at hand. Due to the non-convex optimization problem, this requires gradient descend (see Page 39) w.r.t., for example, the centers and covariance matrices of Gaussian RBF-bases Menache et al. (2005) or the frequencies and phases of Fourier bases Mahadevan et al. (2013).

FEATURE SELECTION

Individual basis functions provided by a universal function approximator do not always improve the (Q-)value significantly, and one can also look for linear combinations thereof to improve performance. Initially literature focused on heuristic cost functions (e.g. *action respecting embeddings*, Bowling et al., 2005, 2006; Biggs et al., 2008), until Parr et al. (2008) provided a general classification into *reward-based* and *subspace invariant* features. REWARD-BASED FEATURES include techniques like state-aggregation (Keller et al., 2007), but recent publications focus on BELLMAN ERROR BASIS FUNCTIONS (BEBF, Parr et al., 2007, details on Page 157 of this thesis), which are a equivalent to KRYLOV BASES (Petrik, 2007). The next BEBF feature is defined as the TD-error of the current solution and therefore propagates reward one step further into the past. It is easy to see that large discount factors γ require many BEBF features[3], as distant rewards influence the value significantly.

REWARD-BASED FEATURES

The second possibility is to encode the transition structure with no regard for the experienced reward, called SUBSPACE-INVARIANT FEATURES. Here the features aim to remain invariant under an application of the transition operator, i.e. $\hat{\mathrm{P}}^\pi[\phi_i](x) \approx \phi_i(x), \forall x \in \mathcal{X}$. Future reward can therefore be propagated far into the past without incurring too much error. The most prominent feature construction method of this class are PROTO-VALUE FUNCTIONS (PVF, Mahadevan and Maggioni, 2007; Johns and Mahadevan, 2007, details on Page 158 of this thesis). These are eigenvectors of the *normalized graph Laplacian* of a neighborhood graph, that has been learned by random walks through the state space. The features have proven to be useful in transferring information between different tasks (Ferguson and Mahadevan, 2006; Ferrante et al., 2008), which is an example of the above mentioned exploitation of prior information. Another subspace-invariant method is SLOW FEATURE ANALYSIS (SFA, Wiskott and Sejnowski, 2002, details on Page 159 of this thesis), which has recently been suggested due to its similarity to PVF (Sprekeler, 2011). Here the features are generalized eigenvectors of the covariance matrix of temporal derivatives, which are constrained to unit variance on the training set. Franzius et al. (2007) show that SFA features in navigation experiments (like the one shown in Figure 4.2) approximate Fourier bases in the underlying state space. On the one hand, these features still suffer the curse of dimensionality, but are on the other hand universal function approximators, independent of the complexity of the input space. This insight allows to learn state-representations in high-dimensional observation spaces.

SUBSPACE-INVARIANCE

[3] It has been reported that the number of features can be reduced by including the average reward as the first feature, called BELLMAN AVERAGE REWARD BASES (BARB, Mahadevan and Liu, 2010).

a) Robot b) Robot View c) Ceiling View

Figure 4.1: (a) the robot used in our real-world experiment, (b) a first-person perspective image the robot uses as current state/observation $\boldsymbol{z} \in \mathcal{Z}$ and (c) the top-down view of a camera used to track the robots position and orientation $\boldsymbol{x} \in \mathcal{X}$.

4.1.3 Isomorphic state and observation spaces

Basis functions discussed in the above subsection require a low-dimensional state space $\mathcal{X}$ or suffer the curse of dimensionality. These spaces need to be hand-crafted from a set of sensor measurements and high-level estimators. In the interest of autonomy, one would like to learn basis functions directly on the high-dimensional OBSERVATION SPACE[4] $\mathcal{Z}$. In this thesis I will demonstrate this idea at the example of observations from a fist-person perspective camera mounted on the head of a wheeled robot, depicted in Figure 4.1. Each camera pixel (and color channel) is an independent dimension in observation space[5] $\mathcal{Z} \subset \mathbb{R}^d$. A 320x200 RGB image has therefore a dimensionality of $d = 192000$. Navigation tasks with this robot are an example of VISUAL TASKS (Jodogne and Piater, 2007). On the one hand, using universal approximators like Fourier bases to learn a Q-value function, and thus an autonomous control on $\mathcal{Z}$, is out of the question due to the large number of pixels. The task-relevant state space $\mathcal{X}$, on the other hand, is only composed of the robots position and orientation. If the environment is visually rich enough, this three dimensional continuous state space $\mathcal{X}$ is embedded as a three dimensional manifold $\mathcal{Z}$ in the space of all images $\mathbb{R}^d$. By using a non-linear function class to construct basis functions on this observable manifold, instead of the space of all images $\mathbb{R}^d$, linear RL algorithms like LSPI become feasible. The autonomous visual navigation control is sketched in Figure 4.2 on Page 48.

VISUAL TASKS

Visual tasks assume the existence of a one-to-one correspondence between $\mathcal{X}$ and $\mathcal{Z}$, which are therefore ISOMORPHISMS. Without prior information about the reward at some observation $\boldsymbol{z} \in \mathcal{Z}$, each feature vector $\boldsymbol{\phi}(\boldsymbol{z}) \in \mathbb{R}^p$ must be distinguishable. This implies that the feature space $\Phi := \{\boldsymbol{\phi}(\boldsymbol{x}) \,|\, \boldsymbol{z} \in \mathcal{Z}\}$ should be an isomorphism of $\mathcal{X}$ as well. Isomorphic spaces differ only in their associated metric, that is, different isomorphic feature spaces Φ differ only their Euclidean distance metric $d_\phi(x, y) = \|\boldsymbol{\phi}(x) - \boldsymbol{\phi}(y)\|_2$. Note that this metric also constrains the representable functions and defines their "smoothness" (see Section 3.2.3). Feature construction methods should[6] therefore primarily differ in the metric they induce.

In the following I review all ISOMORPHIC REPRESENTATION LEARNING publications in high-dimensional observation spaces I am aware of. Some applicable feature

ISOMORPHIC REPRESENTATIONS

[4] Not to be confused with the space $\mathcal{Z} := \mathcal{X} \times \mathcal{A}$, I defined in Chapter 3, although the observation space $\mathcal{Z}$ can also include potential actions to learn basis functions for Q-values.

[5] Here $\mathcal{Z}$ is the set of all possible observations, which is a subset of all possible images in $\mathbb{R}^d$.

[6] Which metric benefits value estimation is analyzed in more detail in Section 4.2 on Page 49.

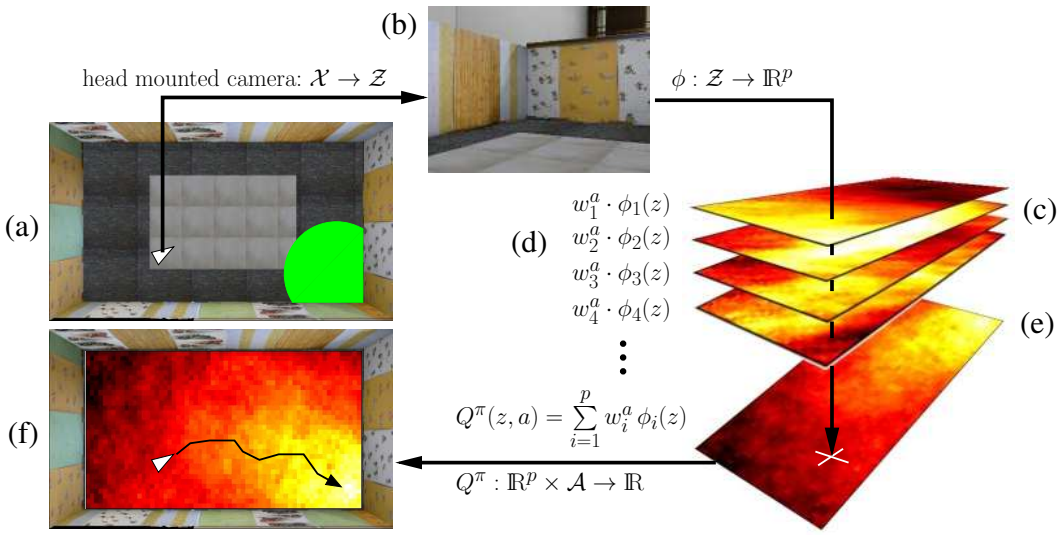


Figure 4.2: Sketch of the autonomous visual navigation experiment in Böhmer et al. (2013). At its position $\boldsymbol{x} \in \mathcal{X}$ in the environment (a), the robot takes a first-person picture $\boldsymbol{z} \in \mathcal{Z}$ (b), which is the input of p learned features $\phi_i(\boldsymbol{z})$. The feature vector $\boldsymbol{\phi}(\boldsymbol{z})$ (c) is for each discrete action $a \in \mathcal{A}$ weighted by LSPI coefficients $\boldsymbol{w}^a \in \mathbb{R}^p$ (d) to yield the Q-value estimation (e). The action with the highest Q-value is executed (f), until the robot arrives at the rewarded goal area (green circle in (a)).

construction methods in literature use linear functions in $\mathcal{Z}$, for example, *predictive projections* (Sprague, 2009) and *incremental SFA* (Luciw and Schmidhuber, 2012). For complex visual scenes, like first-person perspective images, this functions class is not sufficient, though. The majority of authors employs therefore nonlinear functions classes, that is, RKHS (see Section 3.1.2) or neural networks (see Section 3.3.4). Examples using Gaussian RBF-kernels are: action respecting embeddings (Bowling et al., 2005), nearest-neighbor PVF (Mahadevan and Maggioni, 2007), dimensionality reduction of demonstrations (Bitzer et al., 2010) and my own kernel SFA (Böhmer et al., 2013). Neural networks are usually *convolutional deep neural networks*, which define *receptive fields* and share weights to regularize the solution similar to computer vision (e.g. SIFT, Lowe, 1999). Here authors implemented SFA (Legenstein et al., 2010), auto-encoders (Lange et al., 2012; Mattner et al., 2012) and learned locally linear transitions (called *embed to control*, Watter et al., 2015). Deep ANN architectures are generally preferred, as intermediate layers generate increasingly more compact task-representations and learning the Q-value directly with a deep ANN (called *end-to-end* RL) has yielded impressive results in autonomously playing ATARI video games (Mnih et al., 2015). The only *shallow* architecture I am aware of optimizes a heuristic combination of slowness, predictability and other objectives (Jonschkowski and Brock, 2014). However, this thesis focuses on kernel methods. Readers interested in neural network approaches are referred to my review paper Böhmer et al. (2015) on Page 198 of this thesis.

VISUAL CONTROL EXPERIMENTS

Table 4.1 summarizes and compares the above mentioned literature with VISUAL CONTROL EXPERIMENTS. Although, on the one hand, the presented algorithms do not always seem similar on the surface, the learned objectives are predominantly based on successive samples and encode properties of the *transition* model. Notable

Publication	Perspective	Realism	Objective	Fun.Class
Böhmer et al. (2013)	first-person	sim.+robot	transition	kernel
Jonschkowski and Brock (2014)	t.-d.+f.-p.	simulation	transition	shallow
Lange et al. (2012)	top-down	real slot-car	compression	deep
Legenstein et al. (2010)	top-down	toy-example	transition	deep
Luciw and Schmidhuber (2012)	top-down	toy-example	transition	linear
Mattner et al. (2012)	sideways	real pole	compression	deep
Mnih et al. (2015)	sideways	atari-games	end-to-end	deep
Sprague (2009)	first-person	real robot	transition	linear
Watter et al. (2015)	sideways	simulation	transition	deep

Table 4.1: Overview over visual control experiments and their representation learning objectives in literature. Key words are further explained in the main text.

exceptions *compress* observations without consideration of the temporal structure: deep neural network auto-encoders and *principal component analysis* (PCA, Pearson, 1901), which is a linear auto-encoder[7] as well. Experiments, on the other hand, differ widely. Here I want to differentiate three classes of complexity. (i) Many experiments take a *top-down* perspective on the agent and its environment. Similar to discrete state representations, individual pixels encode individual positions almost exclusively. Estimation of individual states requires only locally overlapping groups of pixels and linear methods work fairly well. The biggest challenge is sensory noise. (ii) Some experiments watch a controllable pole or an arm *sideways*. Individual pixel values are also tightly associated with specific states, but estimating some arm-position may require to combine multiple pixels values. (iii) In the by far most difficult class of experiments, agents navigate in a small room using *first-person* perspective images from the agents point of view (as in Figure 4.1). There is no association of individual pixels with the underlying state, which is the agents position and orientation. Moreover, different actions lead to radically different Euclidean distances in $\mathbb{R}^d$. Forward movements affect only pixels of close-by objects, whereas rotations usually change every pixel. Representation learning requires therefore a highly non-linear function class or uniformly colored walls as in Jonschkowski and Brock (2014).

In Section 4.4 I summarize the experimental results of my publication (Böhmer et al., 2013). Here I compare *transition-dependent* kernel SFA features with the non-linear *compression* method *kernel PCA* (KPCA Schölkopf et al., 1998) in the (in my eyes) by far most realistic *first-person* robot navigation experiment to date.

4.2 Theoretical analysis

In this section I analyze the properties for good representations I mentioned on Page 43 theoretically. I restrict myself here to the case of isomorphic feature spaces Φ and linear (Q-)value estimation with LSPI. I answer in particular three questions:

(i) Which distance metric generalizes (Q-)values best (Section 4.2.1)?,
(ii) Which features induce this metric (Section 4.2.2)?
(iii) Can all anticipated value functions be represented (Section 4.2.3)?

[7] Linear PCA can be interpreted as the features that allow the smallest average linear reconstruction error, i.e., $\inf_{\mathbf{W}} \inf_{\mathbf{A}} \mathbb{E}\left[\|\boldsymbol{z} - \mathbf{A}^\top \mathbf{W}^\top \boldsymbol{z}\|_2^2 \,\middle|\, \boldsymbol{z} \sim \zeta\right] \equiv \sup_{\mathbf{W}} \mathbf{tr}(\mathbf{W}^\top \mathbb{E}[\boldsymbol{z}\boldsymbol{z}^\top \,|\, \boldsymbol{z} \sim \zeta]\, \mathbf{W})$ s.t. $\mathbf{W}^\top \mathbf{W} = \mathbf{I}$.

The key results of this section are that for a fixed policy (i) a specific *diffusion distance* generalizes values to unseen states, (ii) *slow feature analysis* (SFA) approximates feature spaces with this metric, and (iii) SFA features can are best suited to approximate *all* (Q-)value fix-points in the *same environment*.

4.2.1 Diffusion distances

To analyze properties of feature spaces $\Phi := \{\boldsymbol{\phi}(x) \,|\, x \in \mathcal{Z}\}$, I assume in the following that Φ is *isomorphic* to the task-relevant state(-action) or observations space $\mathcal{Z}$. We have seen in Section 4.1.3, that such feature spaces only differ in their induced metric $d_\phi(x,y) = \|\boldsymbol{\phi}(x) - \boldsymbol{\phi}(y)\|_2, \forall x,y \in \mathcal{Z}$. Throughout this thesis I have shown that generalization of linear (Q-)value estimators like LSTD (Section 3.3.2) depends crucially on the induced metric d_ϕ. On the one hand, Lemma 3.4 on Page 29 shows that Φ induces constrains that enforce "smoothness" of all representable linear functions $f \in \mathcal{F}_\phi = \{\boldsymbol{a}^\top \boldsymbol{\phi}(\cdot) \,|\, \boldsymbol{a} \in \mathbb{R}^p\} \subset L^2(\mathcal{Z}, \zeta)$ w.r.t. induced distance $d_\phi(\cdot,\cdot)$. On the other hand, using additional *weight-decay* regularization (Page 30) approximates virtual samples with a Gaussian distribution based on this distance metric to counter over-fitting. The selection of features ϕ_i is therefore crucial for generalization. But which metric is best suited to generalize (Q-)values?

DIFFUSION MAPS

Discrete feature construction methods like *proto-value functions* (PVF, Mahadevan and Maggioni, 2007) are based on DIFFUSION MAPS in weighted graphs (Coifman et al., 2005). The underlying idea is the diffusion of "heat" in the graph, where each edge's weight represents heat conductance. The DIFFUSION DISTANCE D_t between nodes is here defined by inducing heat in each node independently and calculating the average in heat differences at time t. Formally, the diffusion distance of the nodes i and j from a symmetric graph with m nodes and nonnegative weight (heat conductance) matrix $\mathbf{G} = \mathbf{G}^\top \in \mathbb{R}^{m \times m}$ is defined as

$$D_t(i,j) := \sqrt{\sum_{k=1}^{m} \frac{1}{\zeta_k} \left((\mathbf{P}^t)_{ik} - (\mathbf{P}^t)_{jk}\right)^2}, \qquad \text{with} \quad P_{ij} := \frac{G_{ij}}{\sum_{k=1}^{d} G_{ik}}. \tag{4.1}$$

Here $\mathbf{P} \in \mathbb{R}^{m \times m}$ denotes a transition matrix, $\mathbf{P}^t = \mathbf{P} \cdots \mathbf{P}$ the product of t transition matrices, and $\boldsymbol{\zeta} \in \mathbb{R}^m$ the steady state distribution of transition matrix $\mathbf{P}$. Note that diffusion distances can thus also be seen as comparing probabilities to arrive in states after a random walk of t steps following transition model $\mathbf{P}$. Small time parameters t reflect therefore local connectivity, whereas large t reveal global structures of the graph. Intuitively speaking, nodes with small differences are connected by many Markov chains of length t. This indicates some similarity between them, which is exploited in SPECTRAL CLUSTERING (Bach and Jordan, 2003). However, heat diffusion is symmetric and spectral clustering relies on symmetric graphs $\mathbf{G}$. In PVF, for example, the graph represents observed transitions between states and has to be symmetrized before diffusion distances are computed.

CONTINUOUS STATE SPACES

In Böhmer et al. (2013) (see Page 145 of this thesis) I translated the above concept of diffusion distances to CONTINUOUS state(-action) or observation spaces $\mathcal{Z}$. The sum in Equation 4.1 translates in the limit to a Lebesgue integral with uniform probability measure $\vartheta : \mathcal{B}(\mathcal{Z}) \to [0,1], \int \vartheta(dz) = 1$. Differences between probability distributions $P^\pi(\cdot|x)$ and $P^\pi(\cdot|y)$ can not directly be calculated, but one can instead calculate the differences between their Radon-Nikodym derivatives $\frac{dP^\pi(\cdot|x)}{d\vartheta}$ and $\frac{dP^\pi(\cdot|y)}{d\vartheta}$. However, derivatives of point-distributions do not have bounded L_2 norms and formally I have to make Assumption 4.1 to ensure $\frac{dP^\pi(\cdot|x)}{d\vartheta} \in L^2(\mathcal{Z}, \zeta), \forall x \in \mathcal{Z}$.

Assumption 4.1 (Assumption 1 in Böhmer et al., 2013) *If state space $\mathcal{X}$ is continuous, the transition model knows no finite set of future states, i.e.*

$$P(B|x,a) = 0\,, \quad \forall B \in \{B \in \mathcal{B}(\mathcal{X}) \mid |B| < \infty\}\,, \qquad \forall x \in \mathcal{X}, \quad \forall a \in \mathcal{A}\,.$$

Lemma 4.2 shows that, under some mild assumptions, diffusion distances in Equation 4.1 are in continuous spaces $\mathcal{Z}$ equivalent to a norm in $L^2(\mathcal{Z},\zeta)$:

Lemma 4.2 *For transition model $P^\pi : \mathcal{Z} \times \mathcal{B}(\mathcal{Z}) \to [0,1]$ and any distribution $\zeta : \mathcal{Z} \to [0,1]$, with respect to which the uniform distribution ϑ in $\mathcal{Z}$ is absolutely continuous, i.e. $\vartheta \ll \zeta$, holds under Assumption 4.1, $\forall x, y \in \mathcal{Z}, \forall t \in \mathbb{N} \setminus \{0\}$:*

$$\int \vartheta(dz)\, \frac{d\zeta}{d\vartheta}^{-1}(z) \left(\frac{d(P^\pi)^t(\cdot|x)}{d\vartheta}(z) - \frac{d(P^\pi)^t(\cdot|y)}{d\vartheta}(z) \right)^2 \quad = \quad \left\| \frac{d(P^\pi)^t(\cdot|x)}{d\zeta} - \frac{d(P^\pi)^t(\cdot|y)}{d\zeta} \right\|_\zeta^2 .$$

Proof: can be found in Appendix C.4 on Page 239. □

For $t > 0$ this yields the following definition of CONTINUOUS DIFFUSION DISTANCES: DIFFUSION DISTANCES

Definition 4.3 (Definition 1 in Böhmer et al., 2013) *Let $\mathcal{Z}$ be a state(-action) space $\mathcal{X}(\times\mathcal{A})$ with ergodic transition model P^π following Assumption 4.1 and steady state (-action) distribution ζ. The diffusion distance $d_t : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}^+$ is defined as*

$$d_t(x,y) := \left\| \frac{d(P^\pi)^t(\cdot|x)}{d\zeta} - \frac{d(P^\pi)^t(\cdot|y)}{d\zeta} \right\|_\zeta , \qquad \forall x, y \in \mathcal{Z}\,, \quad \forall t \in \mathbb{N} \setminus \{0\}\,,$$

where $(P^\pi)^t(\cdot|x) : \mathcal{B}(\mathcal{Z}) \to [0,1]$ denotes the state distribution after t steps with transition model P^π, starting at state $x \in \mathcal{Z}$.

Diffusion distances d_t are of great interest for generalization. I show in Lemma 4.4 that the differences between (Q-)values for a transition model P^π can be *bounded* by the infinite sum of discounted diffusion distances. Note the striking similarity to the "smoothness" bound on linear functions in Lemma 3.4 on Page 29.

Lemma 4.4 *Let $\mathcal{Z}$ be the state or state-action space of an MDP with reward function $r \in L^2(\mathcal{Z},\zeta)$, ergodic transition model $P^\pi : \mathcal{Z} \times \mathcal{B}(\mathcal{Z}) \to [0,1]$ and steady state(-action) distribution ζ. For (Q-)value fix-point $v = r + \gamma \hat{\mathrm{P}}^\pi[v]$ holds*

$$|v(x) - v(y)| \quad \leq \quad \underbrace{\left(\sum_{t=0}^{\infty} \gamma^t\, d_t(x,y) \right)}_{d_\gamma^\pi(x,y)} \|r\|_\zeta\,, \qquad \forall x, y \in \mathcal{Z}\,.$$

Proof: can be found in Appendix C.4 on Page 240. □

I conclude that linear value estimation, if necessary with weight decay regularization, generalizes OPTIMALLY to unseen states when the employed feature space Φ satisfies $\|\boldsymbol{\phi}(x) - \boldsymbol{\phi}(y)\|_2 = d_\gamma^\pi(x,y), \forall x, y \in \mathcal{Z}$. However, the discounted diffusion distances $d_\gamma^\pi(\cdot,\cdot)$ depend on the policy π, which is changing during the course of policy iteration, e.g. in LSPI. The feature space Φ must thus either change with the policy (called *representation policy iteration*, Mahadevan and Maggioni, 2007) or react robustly to policy changes. It is not clear yet how different policies affect diffusion distances, but I observed empirically that feature spaces based on a uniform policy can estimate value function of other policies quite well. OPTIMAL METRIC

4.2.2 Slow representations

Knowing the optimal distance metric d_γ^π for (Q-)value estimation is only half the battle. One must also be able to find (or construct) a feature space Φ that at least approximates this metric.

SELF-ADJOINT OPERATORS

Lemma 4.5 shows that for SELF-ADJOINT[8] transition operators $\hat{\mathrm{P}}^\pi : \mathcal{Z} \to \mathcal{B}(\mathcal{Z})$, the scaled eigenfunctions span a feature space Φ with $d_\gamma^\pi(x,y) \leq d_\phi(x,y), \forall x,y \in \mathcal{Z}$.

Lemma 4.5 *Given the eigenfunctions $\varphi_i \in L^2(\mathcal{Z},\zeta)$ and eigenvalues $\lambda_i \in \mathbb{R}$ of a self-adjoint transition operator in $L^2(\mathcal{Z},\zeta)$, the distance $d_\gamma^\pi(x,y) := \sum_{t=0}^\infty \gamma^t d_t(x,y)$, $\forall x,y \in \mathcal{Z}$, is bounded by the Euclidean distance between scaled eigenfunctions, i.e.*

$$d_\gamma^\pi(x,y) \;\leq\; \tfrac{1}{\sqrt{1-\gamma}} \big\| \tilde{\boldsymbol{\psi}}(x) - \tilde{\boldsymbol{\psi}}(y) \big\|_2 \,, \quad \tilde{\psi}_i(x) := \tfrac{1}{\sqrt{1-\gamma\lambda_i^2}}\, \varphi_i(x) \,, \quad \forall i \in \mathbb{N}, \;\; \forall x,y \in \mathcal{Z} \,.$$

Proof: can be found in Appendix C.4 on Page 241. □

Euclidean distances in scaled eigenspaces of self-adjoint transition operators $\hat{\mathrm{P}}^\pi$ are therefore reasonable approximations of the optimal distance d_γ^π to generalize values. Moreover, using the p eigenfunctions φ_i with the largest absolute eigenvalues $|\lambda_i|$ yields the best possible p-dimensional approximation. The restriction to self-adjoint transition models is analogous to the assumption of symmetric heat diffusion in graph diffusion distances of Equation 4.1. Note that transition models are rarely self-adjoint, though. Proposition 4.6 shows that SLOW FEATURE ANALYSIS (SFA, Wiskott and Sejnowski, 2002) extracts instead eigenfunctions of the *symmetrized* (and thus self-adjoint) transition operator $\hat{\mathrm{P}}_s^\pi := \frac{1}{2}\hat{\mathrm{P}}^\pi + \frac{1}{2}(\hat{\mathrm{P}}^\pi)^*$. SFA optimizes[9]

SFA

$$\begin{aligned} \inf_{\{\phi_i\}_{i=1}^p} \sum_{i=1}^p \hat{\mathcal{S}}[\phi_i] \quad &:= \quad \sum_{i=1}^p \tfrac{1}{n} \sum_{t=1}^n \Big(\phi_i(z_{t+1}) - \phi_i(z_t) \Big)^2 \,, \\ \text{s.t.} \quad \hat{\mathrm{E}}[\phi_i] = 0 \,, \qquad &\hat{\mathrm{E}}[\phi_i \cdot \phi_j] = \delta_{ij} \,, \quad \forall i,j \in \{1,\ldots,p\} \,, \end{aligned} \tag{4.2}$$

given the Markov chain $\{z_t\}_{t=1}^{n+1}$. Details can be found on Page 159 of this thesis.

Proposition 4.6 *In the limit of an infinite ergodic Markov chain in state or state-action space $\mathcal{Z}$, following transition model P^π with steady state distribution ζ, the first p SFA features in $L^2(\mathcal{Z},\zeta)$ are the eigenfunctions φ_i of the symmetrized transition operator $\hat{\mathrm{P}}_s^\pi := \frac{1}{2}\hat{\mathrm{P}}^\pi + \frac{1}{2}(\hat{\mathrm{P}}^\pi)^*$ in $L^2(\mathcal{Z},\zeta)$, corresponding to the p largest eigenvalues $\lambda_i = 1 - \frac{1}{2}\hat{\mathcal{S}}[\varphi_i]$, except the first constant eigenfunction $\varphi_0(x) = 1, \forall x \in \mathcal{Z}$.*

Proof: can be found in Appendix C.4 on Page 241. □

Note that SFA features have almost always[10] *nonnegative* eigenvalues, which excludes some potentially useful features. However, I conclude that SFA constructs a feature space Φ that *approximates* a bound on the optimal metric $d_\gamma^\pi \lessapprox d_\phi$. This insight has interesting consequences for "smoothness" constrains and regularization in LSTD.

---

[8] For a self-adjoint operator $\hat{\mathrm{A}} : L^2(\mathcal{Z},\zeta) \to L^2(\mathcal{Z},\zeta)$ holds $\langle f, \hat{\mathrm{A}}[g] \rangle_\zeta = \langle \hat{\mathrm{A}}[f], g \rangle_\zeta, \forall f,g \in L^2(\mathcal{Z},\zeta)$.

[9] Here $\hat{\mathrm{E}}[\cdot]$ denotes the empirical average operator defined on Page 24, i.e. $\hat{\mathrm{E}}[\phi_i] := \frac{1}{n}\sum_{t=1}^n \phi_i(z_t)$.

[10] $\hat{\mathrm{P}}^\pi$'s eigenvalues $|\lambda_i| \leq 1$ can also be negative. However, eigenfunctions with nonnegative eigenvalues are slower and as $\lim_{i\to\infty} |\lambda_i| = 0$, every *finite* set of SFA features for *infinite* state(-action) spaces will correspond to nonnegative eigenvalues.

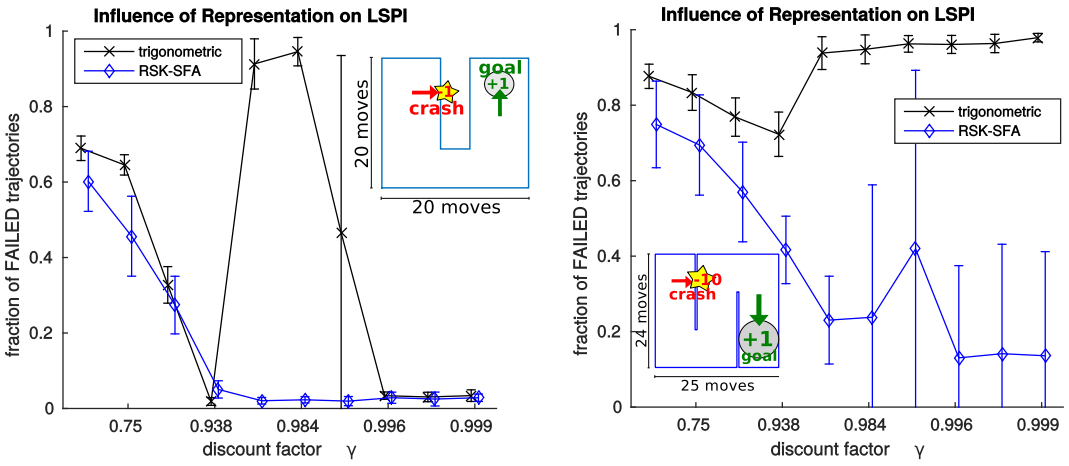


Figure 4.3: Influence of representation on greedy LSPI policies in the tasks shown in Figure 3.4 on Page 37. Large standard deviations are usually caused by some training sets converging and some diverging. In comparison with trigonometric bases (black crosses, see Footnote 16 on Page 36), RSK-SFA features (blue diamonds, learned from the same training set, see Section 4.4.1 on Page 65) significantly reduce therefore divergence for some, but not for all training sets.

On the one hand, scaling the SFA basis functions does not change the LSTD solution[11]. Estimating the (Q-)value function with LSTD based on SFA features should therefore generalize values of unseen states w.r.t. roughly the distance metric d_γ^π of Lemma 4.5. This affects the stability of LSPI as shown in Figure 4.3. Note that the originally divergent regions of the tested parameter space perform significantly better with approximated SFA features. However, the large standard deviations stem from strongly varying behavior for different training sets. For some sets LSPI found almost optimal policies, whereas for others the solution diverged completely. The latter could be a sign of over-fitting and may be addressed by regularization.

Regularization of LSTD WITH WEIGHT DECAY, on the other hand, reveals an influence of feature scaling. Recall the regularized ordinary least-squares (OLS) of Section 3.2.4. Lemma 4.5 shows that the Euclidean distances in the *scaled* eigenspace $\{\varphi_i\}_{i=0}^\infty$ of $\hat{\mathrm{P}}^\pi$ approximate the metric distance d_γ^π, which generalizes value functions to unseen states. SFA features $\phi_i \approx \varphi_i \in L^2(\mathcal{Z},\zeta)$ are *unscaled* approximations of these eigenfunctions. However, solving OLS in SFA feature spaces $\{\phi_i\}_{i=1}^p$ can also generalize w.r.t. d_γ^π by using the regularization term: REGULARIZED LSTD

$$\inf_{\boldsymbol{a}} \hat{\mathrm{L}}[\boldsymbol{a}^\top\boldsymbol{\phi}] + \frac{\sigma^2}{n}\boldsymbol{a}^\top\boldsymbol{\Lambda}\boldsymbol{a}\,, \qquad \text{with} \quad \Lambda_{ij} := \delta_{ij}\Big(1-\gamma\big(\underbrace{1-\tfrac{\hat{S}[\phi_i]}{2}}_{\lambda_i}\big)^2\Big)\,, \quad \forall_{i,j\in\{1,\ldots,p\}}\,. \tag{4.3}$$

Deriving the regularized solution of LSTD analogous to Section 3.3.2, the weights $\boldsymbol{a}^* \in \mathbb{R}^p$ of fix-point $f(x) := \boldsymbol{a}^{*\top}\boldsymbol{\phi}(x)$ with regularization parameter $\lambda := \sigma^2$ are

$$\boldsymbol{a}^* \quad = \quad \Big((\mathbf{C} + \tfrac{\lambda}{n}\boldsymbol{\Lambda}) - \gamma\mathbf{D}\Big)^{-1}\boldsymbol{b}\,. \tag{4.4}$$

This can be implemented as a once-only change to the covariance matrix $\mathbf{C}$. However, I chose to recompute the diagonal scaling matrix $\boldsymbol{\Lambda}$ after each policy improve-

[11] This can be seen by setting $\boldsymbol{\phi} \leftarrow \mathbf{A}\boldsymbol{\phi}$ with any invertible matrix $\mathbf{A} \in \mathbb{R}^{p\times p}$. The LSTD solution does not change: $f(x) = \boldsymbol{b}^\top\mathbf{A}^\top(\mathbf{A}\mathbf{C}\mathbf{A}^\top - \gamma\mathbf{A}\mathbf{D}^\top\mathbf{A}^\top)^{-1}\mathbf{A}\boldsymbol{\phi}(x) = \boldsymbol{b}^\top(\mathbf{C}-\gamma\mathbf{D}^\top)^{-1}\boldsymbol{\phi}(x)\,, \forall x \in \mathcal{Z}$.

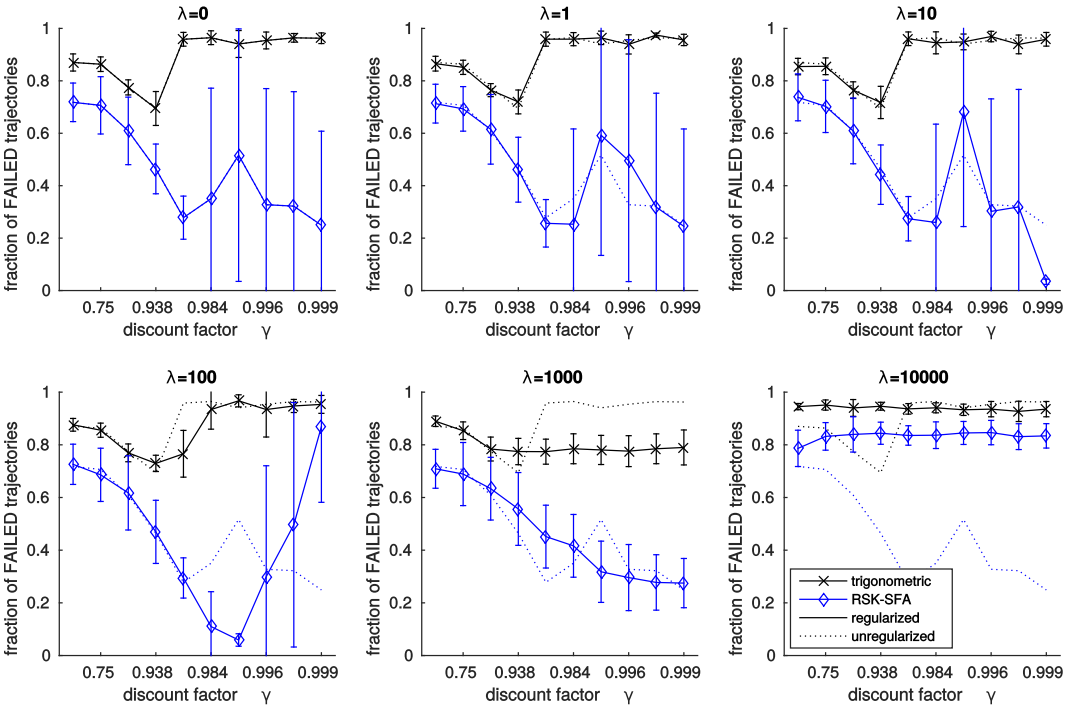


Figure 4.4: Mean performance (i.e., fraction of failed test trajectories) of the experiment on the right side of Figure 4.3 with varying regularization parameters λ. Trigonometric features were regularized with standard *weight decay* (see Section 3.2.4) and RSK-SFA features were regularized by Equation 4.3. Note that SFA features show only for few parameter combinations a significant improvement. However, the standard deviation for large γ can be greatly reduced by regularization, which indicates improved reliability at similar mean performance.

ment, which is not very expensive. Figure 4.4 demonstrates the effects of regularization on trigonometric and SFA representations at the simulated navigation example with a three dimensional state space introduced in Section 3.3.3. On the one hand, note that significant improvements in performance are rare, but so are significant impairments[12]. On the other hand, regularization *can* be used to greatly decrease the standard deviation between training sets for large γ.

I conclude that using LSTD with SFA features generalizes values to unseen states w.r.t. an approximation of the optimal metric d_γ^π. LSPI performs significantly more robust with SFA features and additional weight-decay regularization can help to further stabilize the solution. The latter requires expensive fine-tuning of regularization parameter λ using cross-validation, though.

4.2.3 Anticipated value functions

I have established why SFA features are ideally suited to *generalize* values in linear algorithms like LSTD. Last but not least, I want to show whether or not these features can also *represent* all expectable value functions.

The answer to this question depends primarily on what the agent can *expect*. For example, the best feature $\phi_0 \in L^2(\mathcal{Z}, \zeta)$ to approximate the fix point $v^\pi \in L^2(\mathcal{Z}, \zeta)$ of

[12] With the obvious exception of massively over-regularized problems, for example for $\lambda = 10000$.

the Bellman equation (see Page 12) is trivially the value function itself, i.e. $\phi_0 := v^\pi$. However, the agent does neither know this fix-point in advance, nor does it have access to all the necessary information about the MDP. A set of "optimal features" for LSPI would incorporate this uncertainty by, for example, minimizing the average approximation error of all possible value functions. Proposition 3.7 on Page 35 guarantees the approximation error of the LSTD solution f^* is bounded:

$$\left\|v^\pi - f^*\right\|_\zeta \quad \leq \quad \frac{1}{\sqrt{1-\gamma^2}} \left\|v^\pi - \hat{\Pi}^\phi_\zeta[v^\pi]\right\|_\zeta . \tag{4.5}$$

For the special case of self-adjoint transition operators and corresponding SFA features, I can improve this bound in Proposition 4.7 dramatically.

Proposition 4.7 (Corollary 9 in Böhmer et al., 2013) *The approximation error of the LSTD solution f^* to the true fix-point v^π for MDPs with self-adjoint transition operators using any corresponding SFA features $\{\phi_i\}_{i=1}^p$ is*

$$\left\|v^\pi - f^*\right\|_\zeta \quad = \quad \left\|v^\pi - \hat{\Pi}^\phi_\zeta[v^\pi]\right\|_\zeta .$$

Proof: Follows from Lemma 8 in Böhmer et al. (2013, Page 189 of this thesis). □

In both cases one can guarantee a small LSTD approximation error by minimizing the *projection error* of fix-point v^π. In Definition 4.8 I specify OPTIMAL FEATURES for LSTD by minimizing the average[13] projection error w.r.t. the anticipated distribution of the remaining unknowns: the MDP $m \in \mathcal{M}$ and the policy $\pi \in \Omega$. Here $\mathcal{M}$ denotes the set of all MDP and Ω denotes the set of all policies.

OPTIMAL FEATURES

Definition 4.8 (Definition 12 in Böhmer et al., 2013) *A set of p basis functions $\{\phi_i\}_{i=1}^p \subset L^2(\mathcal{Z}, \zeta)$ is called* OPTIMAL *w.r.t. the distributions $\rho : \mathcal{B}(\mathcal{M}) \to [0,1]$ (over MDP) and $\omega : \mathcal{B}(\Omega) \to [0,1]$ (over policies), if they are a solution to*

$$\inf_{\{\phi_i\}_{i=1}^p} \mathbb{E}\Big[\underbrace{\|v^\pi_m - \hat{\Pi}^\phi_\zeta[v^\pi_m]\|^2_\zeta}_{\text{(bound)}} \;\Big|\; \begin{matrix} m \sim \rho(\cdot) \\ \pi \sim \omega(\cdot) \end{matrix} \Big] ,$$

where v^π_m denotes the (Q-)value fix point $v^\pi_m = r + \gamma \hat{\mathrm{P}}^\pi[v^\pi_m]$ of MDP $m \in \mathcal{M}$.

In Theorem 13 of Böhmer et al. (2013), which can be found on Page 190 of this thesis, I show that SFA minimizes an upper bound on this optimization criteria for all tasks in the *same environment*. I want to give here an extended version of this theorem that provides even more insight in the relationship between optimization problem and anticipated value functions. In order to do that, however, I first need to generalize SFA with a discount factor and regularization kernel to γ-SFA.

Definition 4.9 DISCOUNTED SLOW FEATURE ANALYSIS (γ-SFA) *for training set $\{x_t\}_{t=1}^{n+1}$ in state(-action) space $\mathcal{Z}$, discount factor $\gamma \in (0,1]$ and positive semi-definite kernel $K \in L^\infty(\mathcal{Z} \times \mathcal{Z})$ is defined as, $\forall i,j \in \{1,\dots,p\}$:*

γ-SFA

$$\inf_{\{\phi_k\}_{k=1}^p} \sum_{k=1}^p \underbrace{\tfrac{1}{n}\sum_{t=1}^n \big(\phi_k(x_t) - \gamma\, \phi_k(x_{t+1})\big)^2}_{\text{discounted slowness}}, \quad s.t. \quad \underbrace{\tfrac{1}{n^2}\sum_{s,t=1}^n \phi_i(x_s)\, K(x_s,x_t)\, \phi_j(x_t)}_{\text{regularized variance and decorrelation}} \;=\; \delta_{ij} .$$

[13] In Böhmer et al. (2013) I discuss other definitions of optimality as well (Page 183 of this thesis).

This is a generalized eigenvalue problem, and can be solved in the same way as SFA. In my experiments I only needed to remove the the eigenfunctions φ_i with infinite eigenvalues $\lambda_i = \pm\infty$. All other eigenfunctions had eigenvalues $\lambda_i > 0$. Note that the solution generally depends on discount factor γ, except for one special case of the kernel[14] operator $\hat{\mathrm{K}}[f](x) = \int \zeta(dy)\, K(x,y)\, f(y), \forall x \in \mathcal{X}, \forall f \in L^2(\mathcal{Z}, \zeta)$:

Proposition 4.10 *γ-SFA and slow feature analysis (Wiskott and Sejnowski, 2002) have the same solutions for all $\gamma \in (0,1]$, if (i) the infinite training set is drawn from an ergodic Markov chain and (ii) the kernel operator $\hat{\mathrm{K}}$ is the identity operator $\hat{\mathrm{I}}$.*
Proof: can be found in Appendix C.4 on Page 241. □

SFA is therefore a special case of γ-SFA *without* regularization and for arbitrary γ.

Multiple tasks in the same environment share the same state(-action) space $\mathcal{Z}$ and the same transition model $\hat{\mathrm{P}}$. Corresponding MDP differ therefore only in their mean-reward function $r \in L^2(\mathcal{Z}, \zeta)$. The covariance of these reward functions can be formalized as a positive semi-definite kernel $K \in L^\infty(\mathcal{Z} \times \mathcal{Z})$, i.e. $K(x,y) = \mathbb{E}[r(x)\, r(y)], \forall x, y \in \mathcal{Z}$. Proposition 4.11 shows that γ-SFA features *approximate* the optimal features of Definition 4.8 for *all anticipated tasks* in the same environment:

Proposition 4.11 *γ-SFA optimizes an* UPPER BOUND *on the objective in Definition 4.8 under the assumption that: (i) the infinite training set in state(-action) space $\mathcal{Z}$ is drawn by an ergodic Markov chain with steady state(-action) distribution ζ, (ii) the set of anticipated policies ω contains only the sampling policy π and (iii) MDP in ρ have the same transition model P and reward functions are distributed with zero mean and positive-definite covariance kernel function $K \in L^\infty(\mathcal{Z} \times \mathcal{Z})$.*

Proof: can be found in Appendix C.4 on Page 242. □

Which tasks are considered "anticipated" can be specified by the covariance kernel K. In particular, SFA approximates features for *all* tasks in the same environment, that is, all state(-action)s $x \in \mathcal{Z}$ can experience reward/punishment equally and independent of each other.

For discrete state(-action) spaces $\mathcal{Z}$, the γ-SFA features can be computed analytically as the eigenvectors $\boldsymbol{\phi}_i$ to the smallest generalized eigenvalues λ_i of:

$$\mathbf{A}\boldsymbol{\phi}_i \overset{!}{=} \lambda_i\, \mathbf{B}\boldsymbol{\phi}_i\,, \qquad \mathbf{A} := (1+\gamma^2)\boldsymbol{\Xi} - \gamma\boldsymbol{\Xi}\mathbf{P} - \gamma\mathbf{P}^\top\boldsymbol{\Xi}\,, \qquad \mathbf{B} := \boldsymbol{\Xi}\mathbf{K}\boldsymbol{\Xi}\,. \tag{4.6}$$

Here $\boldsymbol{\Xi}$ is a diagonal matrix of the steady state(-action) distribution ζ, P_{xy} denotes the transition probabilities from state(-action) x to y following sampling policy π, and K_{xy} the kernel between state(-action) x and y. These features are theoretical SFA solutions for $\mathbf{K} = \mathbf{I}$. Figure 4.5 shows theoretical SFA features of a discrete navigation environment. Note the features remain the same, irrespective of the discretization's resolution. Figure 4.6 shows general γ-SFA features with $\gamma = 1$ and different covariance kernels that encode anticipated rewards. Note that the features in the top row resemble a full Fourier base in the potentially rewarded room and, like the value function, drop exponentially outside. Features in the middle row are restricted to horizontal Fourier bases due to the anticipated vertical correlation of rewards, and features in the last row resemble mirrored versions of the first.

In conclusion, γ-SFA features reflect the anticipated correlation structure of rewards in the environment and represent all possible (Q-)value functions optimally.

[14] The kernel corresponding to identity operator $\hat{\mathrm{I}}$ is ζ-almost-everywhere a delta-Dirac function.

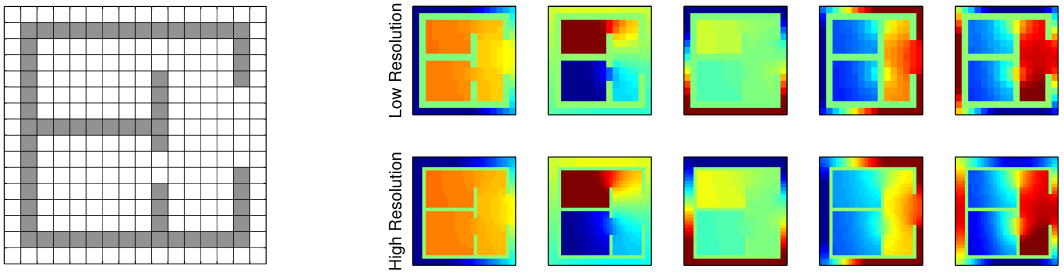


Figure 4.5: A discrete navigation environment with 16×16 states (left plot). The sampling policy randomly chooses stochastic movements in the 4 compass directions, which are blocked by gray walls. On the right are the first 5 theoretical SFA features for environments with 16×16 (upper row) and 32×32 states (lower row).

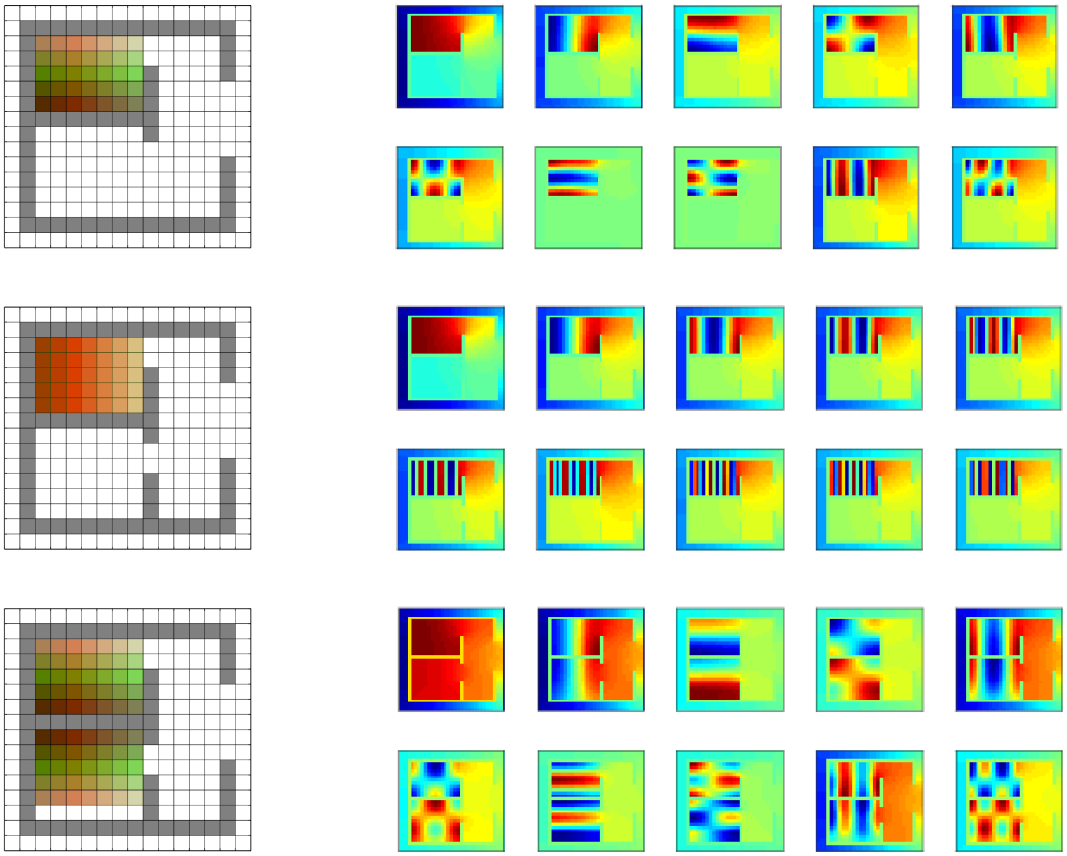


Figure 4.6: The first 10 γ-SFA features with $\gamma = 1$ in the 32×32 states environment of Figure 4.5 (right side). On the left the different covariance kernels K are sketched: independent rewards in one room (upper row), vertical correlation in that room (middle row) and mirrored correlation between two rooms (lower row). White states *can not* yield rewards and states with the same color have *correlated* reward.

4.3 Kernel slow feature analysis

Theoretical analysis in the previous section has demonstrated the outstanding properties of SFA features as basis functions for LSTD. However, existing SFA algorithms were either too restricted (e.g. linear and quadratic SFA, Wiskott and Sejnowski, 2002), had too many hyper-parameters (e.g. deep convolutional quadratic SFA, Franzius et al., 2007) or scaled badly to larger training sets (kernel SFA, Bray and Martinez, 2002).

To learn isomorphic SFA representations of visual tasks, like those described in Section 4.1.3, we developed a *regularized sparse kernel SFA* algorithm (RSK-SFA, Böhmer et al., 2012), which is summarized in Section 4.3.1. The algorithm requires a sparse set of samples, used as "support vectors". Choosing such a set is not trivial and in Section 4.3.2 I derive a suitable support vector selection algorithm (MP-MAH, Böhmer et al., 2012). Both above algorithms are empirically evaluated at the example of linear classification algorithms in Section 4.3.3.

4.3.1 Regularized sparse kernel SFA

Our goal was to learn SFA features from a video recorded by the robot depicted in Figure 4.1 during a random-walk through the environment depicted in Figure 4.2. We chose a kernelized approach, which had been attempted before by Bray and Martinez (2002). However, their derivation of kernel SFA had some formal disadvantages we had to address. First, their objective does not optimize the SFA problem (Wiskott and Sejnowski, 2002, see Equation 4.2 on Page 52). Instead they maximize the quotient over average squared features, centered around means that have been calculated on different time-scales. The solutions appear similar to SFA features, but defy theoretical analysis as in Section 4.2. Secondly, the algorithm estimates features in an online fashion, that can not trivially be extended to batch learning. Thirdly, kernel SFA exhibits numerical instabilities that have to be addressed.

The representer theorem (Proposition 3.2 on Page 26) ensures that the solution of each feature $\phi_i \in \mathcal{H}_\kappa$ can be expressed as linear combination of parameterized kernel functions, that is, $\phi_i(\cdot) = \sum_{t=1}^{n+1} A_{ti}\, \kappa(\cdot, z_t)$. In practice, this is not feasible for large training sets. Although the true solution in $\mathcal{H}_\kappa$ can no longer be guaranteed, a sparse kernel approach aims to represent the feature with a sparse subset (often called SUPPORT VECTORS) $\{s_k\}_{k=1}^m \subset \{z_t\}_{t=1}^{n+1}$, $m \ll n$. The feature is therefore expressed as $\phi_i(\cdot) = \sum_{k=1}^m A_{ki}\, \kappa(\cdot, s_k)$. When the subset is well chosen, the sparse solutions resembles the full kernel solutions closely. Using the reproducing property of Lemma 3.1 on Page 25, the SFA problem w.r.t. parameter matrix $\mathbf{A} \in \mathbb{R}^{m \times p}$ is

SUPPORT VECTORS

$$\inf_{\mathbf{A}} \ \mathbf{tr}\big(\mathbf{A}^\top \dot{\mathbf{K}} \dot{\mathbf{K}}^\top \mathbf{A}\big) \quad \text{s.t.} \quad \mathbf{A}^\top \mathbf{K}\mathbf{K}^\top \mathbf{A} = \mathbf{I}\,, \quad \text{and} \quad \mathbf{A}^\top \mathbf{K}\mathbf{1} = \mathbf{0}\,, \tag{4.7}$$

where $\mathbf{K} \in \mathbb{R}^{m \times (n+1)}$ denotes the kernel matrix with $K_{kt} = \kappa(s_k, z_t)$ and $\dot{\mathbf{K}} \in \mathbb{R}^{m \times n}$ denotes the temporal derivative of $\mathbf{K}$, that is, $\dot{K}_{kt} = K_{k(t+1)} - K_{kt}$.

Fukumizu et al. (2007) have proven for the similar case of *kernel canonical correlation analysis* that the kernel solution can systematically violate the constrains. To stabilize the solution, I introduced the regularization term $\sum_{i=1}^p \|\phi_i\|^2_{\mathcal{H}_\kappa}$, weighted by regularization constant $\lambda \geq 0$. As shown in Section 3.2.4, this is equivalent to virtual sampling in the eigenspace of kernel κ, which is closely related to Gaussian sampling w.r.t. the kernel distance $d_\kappa(x, y) = \sqrt{\kappa(x, x) - 2\kappa(x, y) + \kappa(y, y)}$ of

Algorithm 3 Regularized sparse kernel SFA (RSK-SFA, Böhmer et al., 2012)

`Input:` $\mathbf{K} \in \mathbb{R}^{m\times n}$, $\bar{\mathbf{K}} \in \mathbb{R}^{m\times m}$, $p \in \mathbb{N}$, $\lambda \in \mathbb{R}^+$, $\mathbf{D} \in \mathbb{R}^{n\times n-1}$

$\mathbf{K}' = (\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top)\mathbf{K}(\mathbf{I} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top)$ `// centered` $\mathbf{K}$ `(Eq. 4.8)`

$\bar{\mathbf{K}}' = (\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top)\bar{\mathbf{K}}(\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top)$ `// centered` $\bar{\mathbf{K}}$ `(Eq. 4.8)`

$\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top = \mathtt{eig}(\frac{1}{n}\mathbf{K}'\mathbf{K}'^\top)$ `//` $\mathbf{U}\mathbf{\Lambda}^{-1/2}$ `fulfills constraints`

$(\mathbf{U}_r, \mathbf{\Lambda}_r)$ `= remove_zero_eigenvalues(`$\mathbf{U}, \mathbf{\Lambda}$`)`

$\mathbf{B} = \frac{1}{n-1}\mathbf{K}'\mathbf{D}\mathbf{D}^\top\mathbf{K}'^\top + \lambda\bar{\mathbf{K}}'$ `// original objective (Eq. 4.9)`

$\mathbf{R}\mathbf{\Sigma}\mathbf{R}^\top = \mathtt{eig}\left(\mathbf{\Lambda}_r^{-1/2}\mathbf{U}_r^\top\mathbf{B}\mathbf{U}_r\mathbf{\Lambda}_r^{-1/2}\right)$ `// sphered solution (Eq. 4.9)`

$(\mathbf{R}_p, \mathbf{\Sigma}_p)$ `= keep_lowest_p_eigenvalues(`$\mathbf{R}, \mathbf{\Sigma}, p$`)`

$\mathbf{A} = \left(\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top\right)\mathbf{U}_r\mathbf{\Lambda}_r^{-1/2}\mathbf{R}_p$ `// final solution` $\mathbf{A}$

$\boldsymbol{c} = \frac{1}{n}\mathbf{A}^\top\mathbf{K}\mathbf{1}_n$ `// solution bias` $\boldsymbol{c}$

`Output:` $\mathbf{A} \in \mathbb{R}^{m\times p}$, $\boldsymbol{c} \in \mathbb{R}^p$

Lemma 3.5. The regularization augments the objective in Equation 4.7 by adding the term $\lambda\,\mathbf{tr}(\mathbf{A}^\top\bar{\mathbf{K}}\mathbf{A})$, with $\bar{\mathbf{K}} \in \mathbb{R}^{m\times m}$ being the kernel matrix of the support vectors $\bar{K}_{ij} := \kappa(s_i, s_j)$.

To solve REGULARIZED SPARSE KERNEL SFA (RSK-SFA, Algorithm 3), one first fulfills the zero mean constraint $\mathbf{A}^\top\mathbf{K}\mathbf{1} = \mathbf{0}$ by centering the kernel matrix. To guarantee a symmetric kernel matrix $\bar{\mathbf{K}}$ of the support vectors, it is necessary to also center the kernel output of the support vectors w.r.t. each other (Schölkopf et al., 1998): RSK-SFA

$$\mathbf{K}' := \left(\mathbf{I} - \tfrac{1}{m}\mathbf{1}\,\mathbf{1}^\top\right)\mathbf{K}\left(\mathbf{I} - \tfrac{1}{n}\mathbf{1}\,\mathbf{1}^\top\right). \tag{4.8}$$

The centered matrices $\bar{\mathbf{K}}'$ and $\dot{\mathbf{K}}'$ can be directly derived from $\mathbf{K}'$. With centered kernel matrices RSK-SFA is a generalized eigenvalue decomposition problem. The coefficients $\mathbf{A} \in \mathbb{R}^{m\times p}$ to the features $\{\phi_i \in \mathcal{H}_\kappa\}_{i=1}^p$ are the eigenvectors to the p smallest generalized eigenvalues Λ_{ii} in the diagonal matrix $\mathbf{\Lambda} \in \mathbb{R}^{m\times m}$ of

$$(\tfrac{1}{n}\dot{\mathbf{K}}'\dot{\mathbf{K}}'^\top + \lambda\bar{\mathbf{K}}')\mathbf{A} \;\overset{!}{=}\; (\tfrac{1}{n+1}\mathbf{K}'\mathbf{K}'^\top)\mathbf{A}\mathbf{\Lambda}\,. \tag{4.9}$$

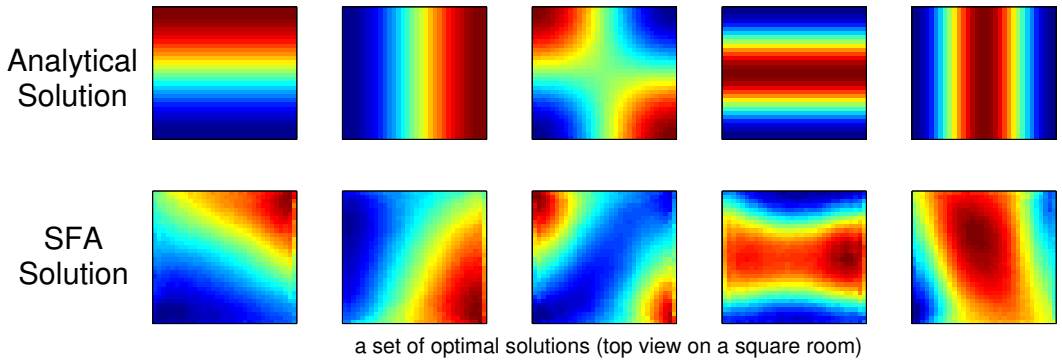


Figure 4.7: RSK-SFA features learned from the two dimensional coordinates of a random-walk in a square room with a Gaussian kernel (lower row). Note the similarity to the analytical solution in the top row (see Franzius et al., 2007).

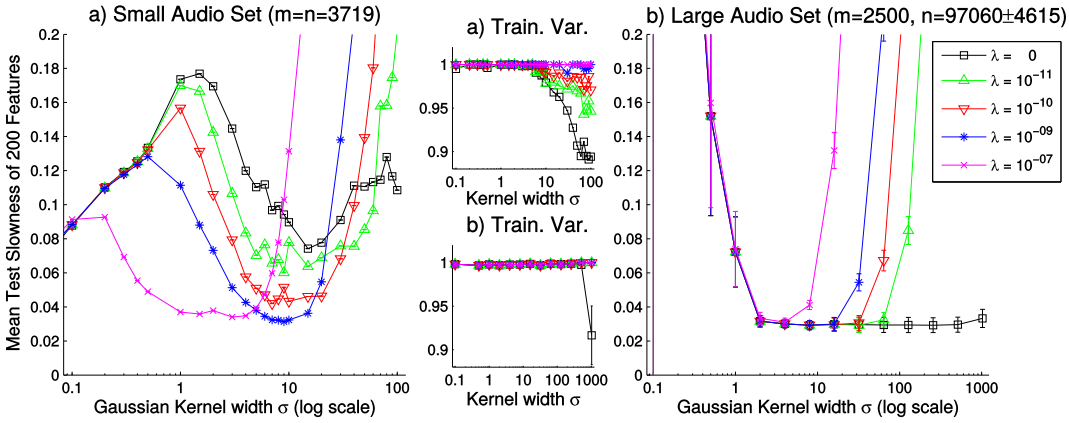


Figure 4.8: Mean Test slowness of 200 RSK-SFA features, based on a Gaussian kernel with varying kernel parameters σ, on two training sets of *audio data* from Böhmer et al. (2012), for details consult Appendix A.1 on Page 124.

Figure 4.7 shows how close learned RSK-SFA features are to analytically predicted features in $L^2(\mathcal{Z}, \zeta)$ (Wiskott, 2003; Franzius et al., 2007). Figure 4.8 evaluates the test slowness on a realistic audio data set from Böhmer et al. (2012). Note that the non-sparse kernel SFA on the left ($m = n$) requires regularization to reach the minimum slowness and to fulfill the constraints (upper center plot). Sparse kernel SFA on the right ($m \ll n$), on the other hand, does not benefit from regularization. Apparently, the restricted function class already constrains the solution sufficiently.

4.3.2 Support vector selection

RSK-SFA requires a set of support vectors for training. Therefore a sparse subset $\{s_j\}_{j=1}^{p} \subset \{z_t\}_{t=1}^{n+1}$ of the training data is needed, which does not constrain the representable functions too much. In other words, the best support vectors span a similar subspace of $\mathcal{H}_\kappa$ as the entire training set. One should therefore select the m-dimensional *index vector* $\boldsymbol{i} \in \mathbb{N}^m$, $s_j := z_{i_j}$, $\forall j \in \{1, \ldots, m\}$, which minimizes the approximation error for all training samples z_t:

$$\epsilon_t^{\boldsymbol{i}} \quad := \quad \inf_{\boldsymbol{a} \in \mathbb{R}^m} \left\| \kappa(\cdot, z_t) - \sum_{j=1}^{m} a_j \, \kappa(\cdot, z_{i_j}) \right\|_{\mathcal{H}_\kappa}^2 \quad = \quad K_{tt} - \boldsymbol{K}_{t\boldsymbol{i}} (\boldsymbol{K}_{\boldsymbol{i}\boldsymbol{i}})^{-1} \boldsymbol{K}_{\boldsymbol{i}t} \,. \tag{4.10}$$

Finding an optimal subset is a NP hard combinatorial problem, and one must therefore find an approximate solution. In Böhmer et al. (2012, for details see Pages 133 and 134 of this thesis) I review two greedy approximation schemes, *online maximization of the affine hull* (online MAH Csató and Opper, 2002) and *matching pursuit for sparse kernel PCA* (MP-KPCA Smola and Schölkopf, 2000). The first is a fast algorithm that selects sparse sets of unknown size m. However, empirical results from RSK-SFA (e.g. in Figure 4.8) show that m is an important parameter for indirect regularization. Selecting sets with predetermined size of m support vectors with online MAH requires a costly binary search of the responsible hyper-parameter. The second algorithm avoids this search by returning a list of up to m samples in order of their importance. This allows to flexibly vary m as a regularization parameter.

Algorithm 4 Support-vector selection with MP-MAH (Böhmer et al., 2012)

```
Input:  {x_t}_{t=1}^n ⊂ X, κ : X × X → ℝ, m ∈ ℕ
K = ∅;   K_1^{-1} = ∅;
∀t ∈ {1,...,n} : ε_t^1 = κ(x_t, x_t)            // initialize errors (Eq. 4.12)
i_1 = arg max_t {ε_t^1}_{t=1}^n                  // select first SV (Eq. 4.11)
for j ∈ {1,...,m−1} do
  a^j = [K_(j,:) K_j^{-1}, −1]^⊤                 // matrix inversion lemma
  K_{j+1}^{-1} = [ K_j^{-1}  0 ; 0^⊤  0 ] + a^j a^{j⊤} / ε_{i_j}^j   // matrix inversion lemma
  for t ∈ {1,...,n} do
    K_(t,j) = κ(x_t, x_{i_j})
    ε_t^{j+1} = ε_t^j − (1/ε_{i_j}^j) (K_(t,:) a^j)^2   // update errors (Eq. 4.12)
  end for
  i_{j+1} = arg max_t {ε_t^{j+1}}_{t=1}^n        // select next SV (Eq. 4.11)
end for
Output:  i ∈ ℕ^m
```

MP-PCA is very costly[15], though, both in time and in memory. To marry these two advantages into a fast algorithm that returns an ordered list of support vectors, I designed the *matching pursuit for online MAH* algorithm (MP-MAH, Algorithm 4, Böhmer et al., 2012).

Online MAH includes a training sample z_t into the set of support vectors $\{z_{i_k}\}_{k=1}^m$ when the error $\epsilon_t^{\boldsymbol{i}}$ surpassed a threshold hyper-parameter η. Increasing η decreases the final size m, and vice versa, but the relationship is not trivial and depends on kernel and data. Note that this selection scheme implicitly guarantees a bound on the approximation error's supremum norm, i.e. $\|\boldsymbol{\epsilon}^{\boldsymbol{i}}\|_\infty = \max_t |\epsilon_t^{\boldsymbol{i}}| \leq \eta$. The idea of MATCHING PURSUIT MAXIMIZATION OF THE AFFINE HULL (MP-MAH) is to select the sample that would minimize this bound given the current index vector $\boldsymbol{i} \in \mathbb{N}^j$: MP-MAH

$$\boldsymbol{i}_{j+1} \quad := \quad \underset{t\in\{1,\dots,n+1\}}{\arg\min} \left\| \left[\epsilon_1^{\boldsymbol{i}\cup t}, \dots, \epsilon_{n+1}^{\boldsymbol{i}\cup t}\right] \right\|_\infty \quad \approx \quad \underset{t\in\{1,\dots,n+1\}}{\arg\max} \epsilon_t^{\boldsymbol{i}} \,. \tag{4.11}$$

As the approximation error of the new support vector vanishes, i.e. $\epsilon_t^{\boldsymbol{i}\cup t} = 0$, one can guarantee that $\min_t \|\boldsymbol{\epsilon}^{\boldsymbol{i}\cup t}\|_\infty \leq \max_t \epsilon_t^{\boldsymbol{i}}$ and the above approximation holds in most cases. Given the current vector of errors $\boldsymbol{\epsilon}^{\boldsymbol{i}} \in \mathbb{R}^{n+1}$, it is therefore easy to select the next support vector. After selection of sample z_k, the error vector can be updated:

$$\epsilon_t^{\boldsymbol{i}\cup k} \quad = \quad \epsilon_t^{\boldsymbol{i}} \quad - \quad \tfrac{1}{\epsilon_k^{\boldsymbol{i}}} \left(K_{kt} - \boldsymbol{K}_{t\boldsymbol{i}} (\mathbf{K}_{\boldsymbol{i}\boldsymbol{i}})^{-1} \boldsymbol{K}_{\boldsymbol{i}k}\right)^2 . \tag{4.12}$$

The inverse of the kernel matrix $\mathbf{K}_{\boldsymbol{i}\boldsymbol{i}}$ can be updated efficiently using the *matrix inversion lemma*. The computational complexity of MP-MAH is with $\mathcal{O}(m^2 n)$ the same as of online MAH, and the memory complexity is with $\mathcal{O}(mn)$ a little higher,

[15] MP-PCA has a computation complexity of $\mathcal{O}(mn^2)$ and a memory complexity of $\mathcal{O}(n^2)$, compared with the corresponding complexities $\mathcal{O}(m^2 n)$ and $\mathcal{O}(m^2)$ of online MAH. Note that $m \ll n$.

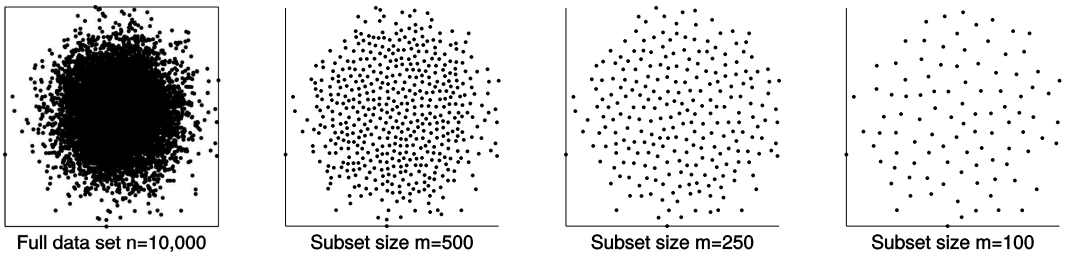


Figure 4.9: Sparse subsets selected by MP-MAH from a Gaussian distribution.

due to caching of some kernel outputs. The algorithm returns an index vector of support vectors, in the order they reduce the supremum norm of the approximation error. In difference to SUPPORT VECTOR MACHINES (SVM, Vapnik, 1995; Boser et al., 1992; Schölkopf and Smola, 2001), these support vectors will be *uniformly distributed* in $\mathcal{Z}$. Figure 4.9 demonstrates this for subsets of different size.

Figure 4.10 shows the effect of support vector selection with Algorithm 4 on the test slowness of RSK-SFA features learned from audio data, that is, the same training set that yielded the left plot of Figure 4.8. All discussed algorithms have comparable performance and clearly outperform random subset selection. Note in particular the close similarity between online MAH and MP-MAH. MP-MAH seems therefore well suited to select support vectors for RSK-SFA.

A different picture emerges when RSK-SFA features are used as state representations in visual tasks in Section 4.1.3. Figure 4.11 shows the fraction of successful test trajectories in the simulated visual navigation experiment in Böhmer et al. (2013, see Appendix A.2). The policies were learned by LSPI on RSK-SFA features with Gaussian kernels of the observed images $z_t \in \mathcal{Z}$. Support vectors were selected by MP-MAH based on Gaussian kernels on the training images (triangles) or the corresponding positions and orientations $x_t \in \mathcal{X}$ (circles). The best test slowness of RSK-SFA features were achieved with support vectors selected in $\mathcal{Z}$ (not shown). The depicted LSPI performance, on the other hand, is much *worse* than with randomly selected subsets[16] (red crosses). It is surprising, however, that the opposite

[16] For small kernel-parameters σ, MP-MAH approaches the performance of random subsets. This is rooted in the selection of samples with the largest approximation errors. In the limit $\sigma \to 0$, all errors are equal and MP-MAH selects therefore samples randomly.

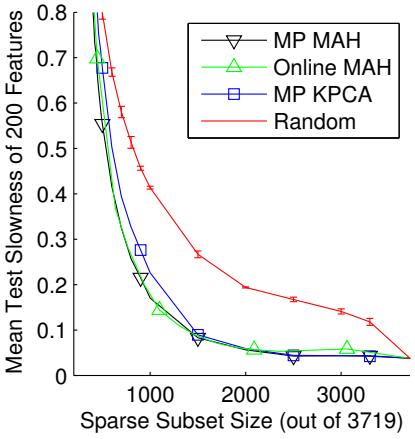


Figure 4.10: The effect of support vector selection on RSK-SFA performance for all discussed algorithms and random subset selection (from Böhmer et al., 2012, for details see Appendix A.1 on Page 124). Note that all algorithms clearly outperform random selection, and how closely MP-MAH follows the performance of Online MAH.

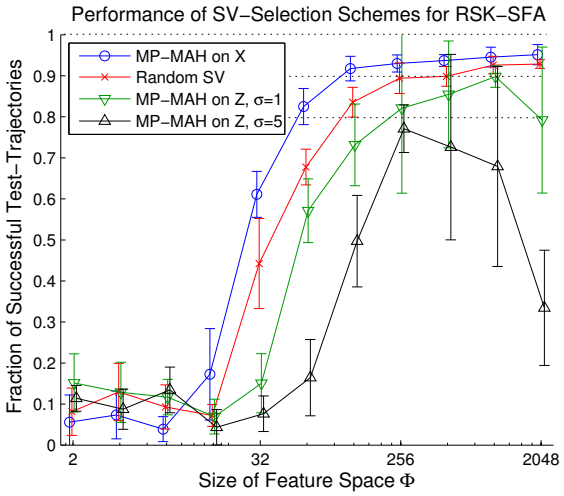


Figure 4.11: The influence of support-vector selection on policies learned by LSPI on RSK-SFA features (from Böhmer et al., 2013, for details see Appendix A.2 on Page 145). Note that the support vectors selected by MP-MAH with Gaussian kernels on images in $\mathcal{Z}$ (green and black triangles) perform *worse* than random selection (red crosses). MP-MAH with Gaussian kernels on the robots coordinates in $\mathcal{X}$ (blue circles), on the other hand, performs significnatly better.

is true for subsets selected in $\mathcal{X}$. Here the learned features were less slow, but LSPI performance excelled over random support vector selection. I can not fully explain this effect[17], but it is worth noting the different Euclidean metrics in $\mathcal{Z}$ and $\mathcal{X}$. MP-MAH selects samples that are uniformly distributed w.r.t. the Euclidean metric used by the Gaussian kernel. A support vector set that is uniformly distributed in $\mathcal{Z}$ can be arbitrarily distributed in $\mathcal{X}$ and vice versa. Although RSK-SFA benefits from uniform distribution in $\mathcal{Z}$, the LSPI algorithm appears to prefer uniform distribution in $\mathcal{X}$. The source of this preference is unclear. Figure 4.12 shows that for linear classification, which estimates a continuous discrimination function, MP-MAH support vector selection in $\mathcal{Z}$ can yield significant improvements. I interpret this as evidence that the above preference is probably not rooted in LSTD, which also estimates a continuous Q-value function. As in the empirical analysis of LSPI's stability in Section 3.3.3, I expect the point-wise policy improvement step to be the most likely culprit. However, future works must find a mathematically sound explanation and derive a way to select support vectors uniform w.r.t a diffusion metric rather than the observation space's Euclidean metric.

4.3.3 SFA for classification

Before applying RSK-SFA to learn representations for autonomous control, I decided to test the learned features as preprocessing of a vowel classification task. I compared the classification of the spoken words "head" and "heed" with linear classification algorithms. To highlight the usefulness of SFA features for linear algorithms, I contrasted the classification results with preprocessing learned by a sparse kernel version of PCA (SK-PCA Schölkopf et al., 1997, 1998). Figure 4.12 shows the accuracy for LINEAR DISCRIMINANT ANALYSIS[18] (LDA, see e.g. Bishop, 2006), regularized[19] LO-

[17] I discuss the influence of support vectors on the RSK-SFA solution more formally in Section 6.3 of Böhmer et al. (2013), which can be found on Page 184 of this thesis.

[18] LDA is almost identical to least-squares classification (Bishop, 2006) and with a small modification to the target weights (Duda and Hart, 1973) to Fisher discriminant analysis (FDA, Fisher, 1936).

[19] Unregularized logistic regression showed clear signs of over-fitting, which vanished with regularization of the Hessian matrix. See Böhmer et al. (2012) in Appendix A.1 for details.

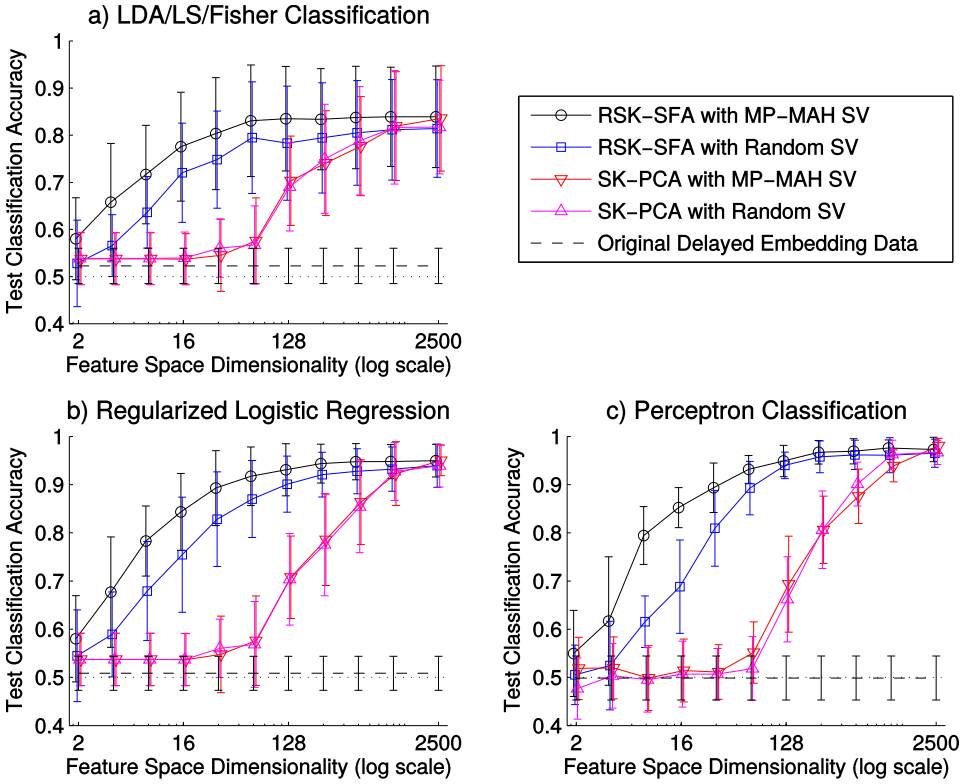


Figure 4.12: Accuracy of vowel detection from Böhmer et al. (2012, for details see Appendix A.1 on Page 124). Delayed embedded sound amplitudes were classified by 3 linear algorithms (a-c) based on RSK-SFA (circles and squares) and SK-PCA (triangles) features, with Gaussian kernels and support vectors selected by MP-MAH or at random. Note that (i) RSK-SFA requires in all cases *much* less features than SK-PCA, which are varied on the horizontal axis (in a log scale), and (ii) RSK-SFA clearly benefits from support vectors selected by MP-MAH.

GISTIC REGRESSION (solved by ITERATIVE REWEIGHTED LEAST SQUARES Rubin, 1983) and the PERCEPTRON ALGORITHM (Rosenblatt, 1962, with a decaying learning rate). All classification algorithms are applied either to the original space of delayed embedded audio data (dashed line), or on features learned by RSK-SFA (circles and squares) or SK-PCA (triangles).

It is worth noticing that the first 64 PCA features do not seem to encode *any* relevant information to classify the spoken vowel. SFA features, on the other hand, become useful early on (between 4 and 8 features) and reach close to perfect accuracy (0.97 for perceptrons and 0.95 for logistic regression) with 256 features. I interpret this as evidence that SFA does not simply encode data according to their reconstruction error, like PCA and other auto encoders (see *deep neural networks* in Section 3.3.4). The theoretical analysis of Section 4.2 suggests instead that SFA extracts latent variables that exhibit the Markov property, that is, which are governed by the underlying transition model. A sequence of the words "head" and "heed", which have the same beginning and end, can be modeled by three latent "states": the vowel "ea", the vowel "ee" and the rest. The fact that 256 features were required for optimal accuracy indicates the problem might not be *linearly separable*. In this case a suitable set of non-linear basis functions are required for good performance.

4.4 Empirical comparison of representations

To evaluate SFA features spaces as representations for control, I compared their LSPI performance with other state-of-the-art representation learning algorithms on benchmark tasks (Böhmer et al., 2013). In Section 4.4.1 I summarize my results for the discrete *puddle-world* task, and in Section 4.4.2 in a visual robot navigation task, both simulated and with a real robot.

4.4.1 Representations for discrete RL

To compare SFA with representation learning methods developed for discrete state spaces (see Section 4.1.2 for an overview), I evaluated the PUDDLE-WORLD task. PUDDLE WORLD
The task was first introduced by Boyan and Moore (1995), but the presented details here were adapted from Sutton (1996). Puddle-world is a continuous stochastic navigation task, in which the agent has to avoid punishing "puddles", that can be discretized in several resolutions. To get rid of discretization effects, all results are

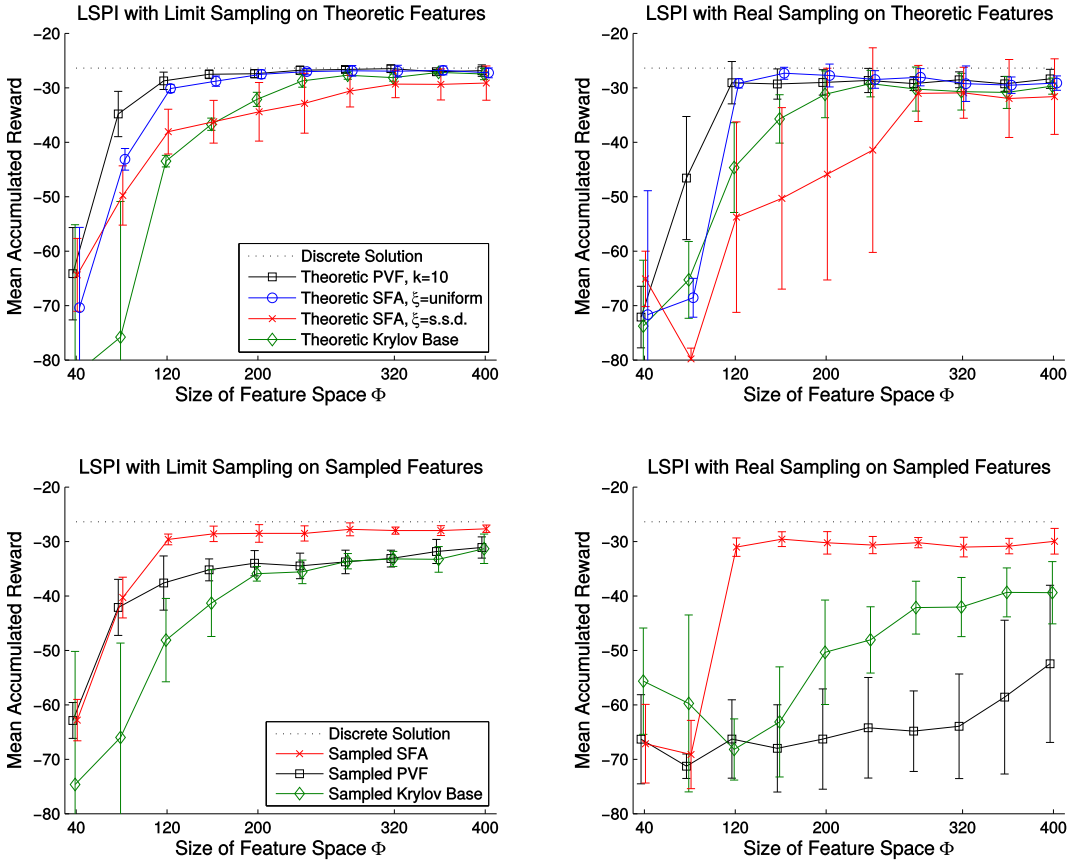


Figure 4.13: Mean accumulated reward as performance of LSPI in the discrete puddle-world task, using theoretically computed (theoretic features, top row) and empirically estimated (sampled features, bottom row) features. LSPI trained either with an ideal set of all state-actions pairs (limit sampling, left column) or a sampled random-walk (real sampling, right column). For details see Böhmer et al. (2013). Note that all algorithms are roughly equivalent with ideal sampling (upper left plot), but SFA features are superior in practice (lower right plot).

presented as mean and standard deviation over state space sizes between 20×20 and 50×50. Details can be found on Pages 172 to 176 of this thesis.

Figure 4.13 plots the LSPI performance (mean accumulated reward) against the number of learned features. Discrete state spaces allow to exploit the true transition and reward model to solve the optimization problems of both representation learning and LSPI exactly. To investigate the influence of sampling on the different representations, I evaluated all combinations of feature learning (computed *theoretically* or *sampled* realistically) and LSPI (based on *limit sampling* of all possible state-action pairs or *real samples* drawn by a Markov chain).

PERFECT INFORMATION

Note that when all algorithms have access to PERFECT INFORMATION (upper left plot), all algorithms perform very good. SFA trained with a uniform "steady-state distribution" (here denoted ξ instead of ζ, blue circles) is almost as good as the front-runner, proto-value functions (PVF, Mahadevan and Maggioni, 2007, black squares). Using the true steady-state distribution ξ of the random sampling policy worsens the performance. Although I can not conclusively prove it, I observed the same behavior time and time again during the course of this thesis and it always seemed to be connected to the policy improvement step in LSPI.

RANDOM-WALK

When the representation is learned by a SAMPLED RANDOM-WALK (lower row), on the other hand, SFA emerges as a clear winner over both PVF and reward-based Krylov bases (Petrik, 2007). In particular when LSPI is also trained on a random-walk (lower right plot), SFA remains the only viable alternative. It is not entirely clear why the LSPI performance of PVF features caves in here, though. I observed that theoretical PVF extracts some high-frequency features, which are probably hard to estimate from realistically sampled training data. This explains the lower left plot, but the lower right scenario remains a mystery and calls for more research.

4.4.2 Representations for autonomous robotics

As the climax of this chapter, I show in the following that features that encode the transition structure yield better continuous representations for LSPI. This is particularly true when the observation space $\mathcal{Z}$ does not provide a metric that generalizes values well. I use the example of visual robot navigation depicted in Figures 4.1 and 4.2 on Page 47. The robot was able to execute three actions: move ca. 30cm forward or turn 45° left/right. Observations are first-person perspective images, scaled down to 32×16 pixels RGB-images, that is, $z_t \in \mathcal{Z} \subset \mathbb{R}^{1536}$. As explained in Section 4.1.3, first person perspective images induce a particularly challenging Euclidean norm. Long movements forward change only the pixel-values depicting nearby objects, whereas even small rotations change every pixel radically. Using a Gaussian kernel, these observations do not generalize to "neighboring" states at all. This provides an ideal testbed to evaluate generalization by the learned representations.

QUALITATIVE EVALUATION

To show that SFA features work in practice, I evaluated the control learned by LSPI QUALITATIVELY on a real robot in a 3m $\times$ 1.8m rectangular space[20]. RSK-SFA was trained with 8000 support vectors out of 35000 observations, which were down-sampled from a continuous video recoded during a 10 hours random-walk. The video contained 11,656 transitions, which were used for LSPI. Figure 4.15 depicts this training set from a birds-eye perspective, colored by the received reward in two

[20] The space was encompassed by tilted tables. The tables were covered in various wallpaper, to yield observations roughly isomorphic to the underlying state of robot positions and orientations.

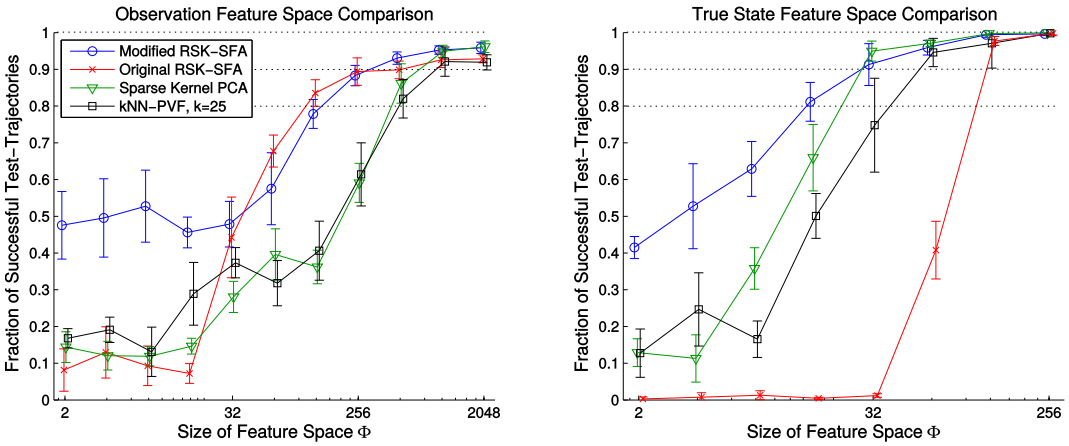


Figure 4.14: Comparison of LSPI performance for different learned representations in a simulated robot navigation experiment (from Böhmer et al., 2013).

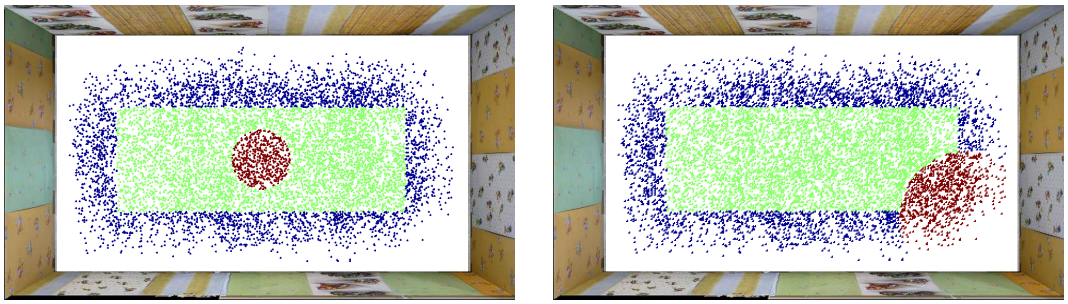


Figure 4.15: Training sets and rewards (blue -1, green 0, red +1) for LSPI in two real-world robotic navigation experiments in Böhmer et al. (2013).

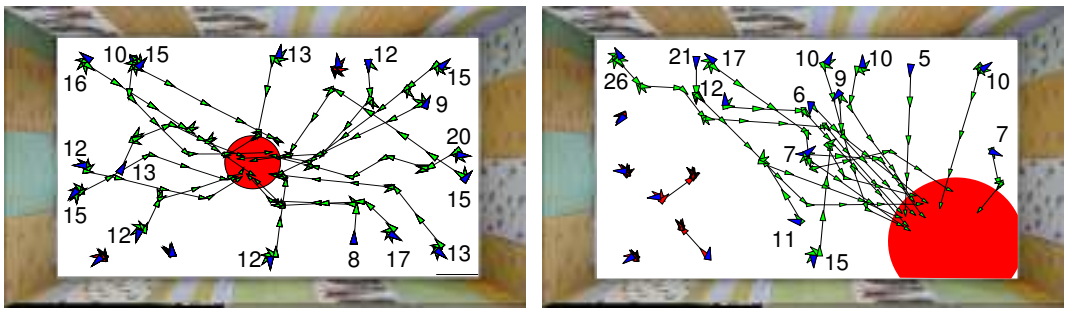


Figure 4.16: Test trajectories of two real-world robotic navigation experiments (from Böhmer et al., 2013, using the training sets depicted in Figure 4.15)

navigation tasks. Figure 4.16 shows 20 test trajectories executed by the control learned with 128 RSK-SFA features anf discount factor $\gamma = 0.9$.

QUANTITATIVE EVALUATION

To compare continuous SFA representations to auto-encoders and PVF, I performed a QUANTITATIVE EVALUATION on a realistic (but not photo-realistic) rendered simulation of the above visual navigation experiment. Comparing the methods rather than the implementations requires to optimize the same class of functions. The continuous extension of PVF constructs a *nearest neighbor* graph based on the output of a Gaussian kernel (Mahadevan and Maggioni, 2007), which is formally equivalent to a sparse kernel method (Mahadevan and Maggioni, 2007). PCA is a linear auto-encoder (see Footnote 7 on Page 49), and one can derive a sparse kernel PCA (SK-PCA, similar to Schölkopf et al., 1998; Smola and Schölkopf, 2000). All algorithms were based on the same set of 4000 support vectors with the same kernel parameters. The reward distribution was similar to the right plot of Figure 4.15.

LSPI performance is plotted in Figure 4.14 against the number of features. Features were learned based on observations $z_t \in \mathcal{Z}$ in left plot and the robot's position and orientation $x_t \in \mathcal{X}$ in the right plot. Note that the first 16-32 RSK-SFA features (red crosses) apparently do not represent the state sufficiently to allow any control. This is due to a to-date little understood imbalance between movement and rotation actions. Results can be vastly improved by *importance sampling*, which weights the actions differently. The MODIFIED RSK-SFA[21] features (blue circles) weight movements 5 times as strongly as rotations and show a dramatic improvement for the first 32 or 64 features. Note also, that the advantage of modified RSK-SFA over PVF and PCA is much more pronounced when based on first person observations than on robot coordinates. I interpret this as evidence that all representation learning methods work equally well in input spaces $\mathcal{X}$, where the metric already resembles a diffusion metric. On the other hand, in observation spaces $\mathcal{Z}$ with vastly different metrics, compression by PCA and continuous PVF does not necessarily generalize values well, whereas SFA features are guaranteed resemble a well generalizing diffusion metric, as proven in Section 4.2.2.

MODIFIED RSK-SFA

4.4.3 Conclusion

In Section 4.4.2, I have presented evidence that non-linear SFA can learn better representations for linear RL algorithms like LSPI. I also traced this advantage analytically to SFA's *diffusion metric* (Section 4.2), which optimally generalizes values for the sampling policy. However, this analysis only holds for the sampling policy and does not account for the policy improvement step of LSPI. Experiments have shown how fickle RSK-SFA features react to the sampling distribution[22]. In conclusion, I have shown that SFA (and the extension γ-SFA) is able to learn near optimal representations for RL, but also their weaknesses in practical applications.

[21] Modified RSK-SFA can be found in Böhmer et al. (2013) and on Pages 160 and 161 of this thesis.

[22] The same effects show up in all non-linear SFA algorithms. An example with deep neural networks can be found in Franzius et al. (2007), where sampling is used to reproduce the behavior of specialized neurons in the visual system.

4.5 Challenges to representation learning

Representation learning for control is an active field of research and many problems remain to be solved in the future. In the following I want to discuss three challenges, that, in my opinion, hold back the field and deserve more attention.

4.5.1 The curse of state dimensionality

I have shown that SFA *embeds* all observed states in a feature space $\{\phi_i\}_{i=1}^{p}$, which approximates a diffusion metric $d_\phi \approx d_\gamma^\pi$. This is a generalization of the FOURIER BASES predicted by Wiskott (2003) and Franzius et al. (2007) for specific transition models. Fourier bases are known as UNIVERSAL FUNCTION APPROXIMATORS and can (in the limit $p \to \infty$) approximate all continuous functions. As values are continuous w.r.t. metric d_γ^π (see Lemma 4.4), features that resemble Fourier bases in this metric can (in the limit) approximate every value function in the same environment arbitrarily well (see Proposition 4.11). In summary, SFA features can be interpreted as Fourier bases, warped by the symmetrized transition structure of $P_s^\pi = \frac{1}{2}\hat{\mathrm{P}}^\pi + \frac{1}{2}(\hat{\mathrm{P}}^\pi)^*$ (see Lemma 4.5 and Proposition 4.6) to generalize values in observation space $\mathcal{Z} \subset \mathbb{R}^d$. FOURIER BASES

On the one hand, my experiments demonstrate that, as long as the observation space is an isomorphism to the "true" Markovian state space $\mathcal{X}$, the dimensionality of the observations d is *not* a limiting factor. On the other hand, the approximation quality of Fourier bases in a "true" state space $\mathcal{X} := [0,1]^b \subset \mathbb{R}^b$ depends *exponentially* on the dimensionality b: doubling the number of sine/cosine waves in each dimension produces 2^b times as many Fourier bases. Adding an independent state variable requires easily a magnitude more features for same approximation quality. SFA suffers therefore the CURSE OF DIMENSIONALITY w.r.t. the underlying state space. In practical applications this is amplified by two effects: (i) every independent variable that reliably changes the observation increases state dimensionality b, and (ii) most SFA features are mixtures of *all* state variables. These problems can be illustrated by including DISTRACTOR variables, like illumination and independently moving objects, which do not affect the controllable state transitions or rewards. Jonschkowski and Brock (2014) propose heuristic regularization terms for "causality" and "repeatability" to reduce the influence of these distractors. They demonstrate their approach in a simplified simulation of the slot-car racer task of Lange et al. (2012), which included an uncontrollable second car. The results are very encouraging: the cars positions appear to be encoded in separate variables. Learning or planning a control for the controllable car can therefore be greatly accelerated by excluding the second car's representation entirely. Note that this kind of regularization could also help to determine the task-dependent subspace (see Section 4.1.1). However, their architecture resembles the auto-encoder setup, and it is not clear how to introduce these regularizations into an approach based on SFA. DISTRACTORS

In Section 4.2.3 I suggest another possibility to address the curse of dimensionality: reducing the number of anticipated reward functions to reduce the number of required features. This can exploit many REGULARITIES IN REWARDS. Consider the case of a household robot that must learn to fetch something, for example, a newspaper or a pair of slippers. If fetching one newspaper is rewarded, every other newspaper will be as well. The corresponding covariance kernel K would therefore correlate all states which encounter an object of the same "class", whereas states REWARD REGULARITIES

without encounters would not anticipate any reward. The resulting representation should reduce the number of required features drastically, while retaining the approximation abilities of SFA features. As an interesting side-effect, the representation would also allow to generalize experiences from one object to another in an *object-oriented* way. However, I have not performed any experiments with these features and future research must show how stable and efficient they are in practice.

4.5.2 Representations for policy iteration

SFA learns features that generalize values of the sampling policy π. I confirmed empirically that other policies appear to also generalize, but there is ultimately no guarantee. In Böhmer et al. (2013, see also Page 183 of this thesis) I discuss alternative definitions of optimality, for example, "all possible policies for the same task" and a "worst case bound". However, optimizing over a distribution of possible policies involves extremely challenging math and I have not come up with an objective that can be estimated from data yet. As an alternative to theoretical derivations, Watter et al. (2015) train an auto-encoder and the corresponding transition model at the same time. This resembles a KALMAN FILTER (Kalman, 1960) with deep convolutional neural networks and is reported to be extraordinary stable when *planning* trajectories with the learned model. Unlike SFA, the transition model is conditioned on the executed action, and the method appears fairly robust to the choice of sampling policies. Other authors used instead HEURISTICS to ensure Q-values can encode different policies (e.g. "repeatability loss" in Jonschkowski and Brock, 2014). As an example for this type of optimization, I propose here the heuristic of "maximal action variance". Q-value functions can express different policies when actions in the same state *can* have strongly varying values. Function outputs can vary the most when for each state the variance of each basis function w.r.t. actions is as high as possible. With a sufficiently random[23] sampling policy π, and under the constraints $\langle \phi_i, \phi_j \rangle_{\xi\pi} = \delta_{ij}$, the average variance of actions can be simplified:

HEURISTIC OPTIMALITY CRITERIA

$$\sup_{\{\phi_i\}} \sum_{i=1}^{p} \iint \xi(dx)\, \pi(da|x) \Big(\phi_i(x,a) - \int \pi(da'|x)\, \phi_i(x,a') \Big)^2 \equiv \inf_{\{\phi_i\}} \sum_{i=1}^{p} \Big\| \hat{\Gamma}_\pi[\phi_i] \Big\|_\xi^2, \quad (4.13)$$

which is a feasible regularization term when $\pi(\cdot|x_t)$ can be sampled[24]. Note that SFA with this regularization does no longer provide any guarantees. Evaluation whether or not this representation improves LSPI performance is left for future works.

GENERALIZATION ERROR

Autonomous learning may also require a different notion of GENERALIZATION ERROR. Both policy π and steady-state distribution ξ change over time. The discussed algorithms optimize functions in $L^2(\mathcal{Z}, \zeta)$, which depends on probability measure ζ, either as $\zeta(dx) = \xi(dx)$ or $\zeta(dx, da) = \xi(dx)\, \pi(da|x)$. Without prior knowledge about these distributions, it may be prudent to optimize γ-SFA w.r.t. a neutral probability measure, for example the uniform measure ϑ. Optimization may still rely on training sequences by *importance sampling*[25] of observed samples $z_t \in \mathcal{Z}$

[23] For deterministic sampling policies π (i.e. point-distributions) holds $\|\hat{\Gamma}_\pi[\phi_i]\|_\xi^2 = \|\phi_i\|_{\xi\pi}^2 \overset{!}{=} 1$.

[24] Variance could otherwise be defined w.r.t. uniform policy τ, which is independent of state $x_t \in \mathcal{X}$ and thus easy to sample. The corresponding regularization term is $\sum_{i=1}^{p} (\|\hat{\Gamma}_\tau[\phi_i]\|_\xi^2 - \|\phi_i\|_{\xi\tau}^2)$.

[25] Using importance sampling in γ-SFA to change the observed steady state distribution ξ of policy π into the state distribution ϑ and policy τ effectively changes the transition operator $\hat{\mathrm{P}}'[f] = \sqrt{\frac{d\vartheta}{d\xi}} \cdot \hat{\mathrm{P}}^\pi \big[\sqrt{\frac{d\xi}{d\vartheta}} \cdot \frac{d\tau}{d\pi} \cdot f \big]$ and the correlation operator $\hat{\mathrm{K}}'[f] = \sqrt{\frac{d\vartheta}{d\xi}} \cdot \hat{\mathrm{K}} \big[\sqrt{\frac{d\vartheta}{d\xi}} \cdot f \big], \forall f \in L^2(\mathcal{X} \times \mathcal{A}, \xi\pi)$.

with Radon-Nikodym derivative $\frac{d\vartheta}{d\zeta}(z_t)$, as demonstrated by modified SFA on Page 68 (see details in Appendix A.2 on Pages 160 and 161). However, transitions that are not observed can not influence optimization. In regions of $\mathcal{Z}$ not visited by the training sequence, the estimate of features ϕ_i must remain uncertain, even for Markov chains that are technically ergodic in the limit of infinite samples. This problem is rooted in the lack of assumptions in inductive machine learning and can fundamentally not be overcome by inductive representation learning. Any solution must therefore, at least in parts, contain a deductive element, which I introduce in the next section. As of now, however, there is no alternative to sampling the entire observation space with some suitable sampling policy, while simultaneously estimating the probability density function $\frac{d\zeta}{d\vartheta} \in L^2(\mathcal{Z}, \zeta)$. This PDF is the inverse of the desired importance weights $\frac{d\vartheta}{d\zeta} = \frac{d\zeta}{d\vartheta}^{-1}$ and can thus be directly used to estimate γ-SFA features w.r.t. the uniform state(-action) distribution ϑ. I have not performed any experiments, though. Future works must evaluate how applicable this approach is in practice.

4.5.3 The curse of insufficient samples

I have shown above how the curse of state dimensionality prevents to approximate diffusion distances accurately with SFA-like feature spaces. I also reviewed alternative approaches, often (but not always) resembling auto-encoders, that aim for compact representations which scale linearly with the state dimensionality (e.g. Jonschkowski and Brock, 2014; Watter et al., 2015). These approaches require to plan or learn control with non-linear algorithms, as no functional basis on the "true" state space $\mathcal{X}$ is constructed. Nonetheless, recent results by Watter et al. (2015) report compact representations that allow to *deductively* plan in low-dimensional problem domains. However, learning *any* isomorphic state representation with inductive learning *must* face the CURSE OF INSUFFICIENT SAMPLING. This is another consequence of the curse of state dimensionality: to generalize the representation over $\mathcal{X}$ (or the isomorphic observation space $\mathcal{Z}$), inductive learning requires samples from all regions of $\mathcal{X}$. The number of necessary training samples for purely inductive learning grows therefore exponential in the state dimensionality d. Control in simulated domains (like ATARI games, Mnih et al., 2015) can provide sufficient samples, but physical systems like robots may require years of training and will most likely break before the representation converges. Learning representations with less samples is thus imperative.

CURSE OF SAMPLING

Machine learning generalizes to unseen samples by averaging the predictions of "similar" training samples through regularization or constraints (see Section 3.2). Similarity is usually determined by the Euclidean metric in observation space $\mathcal{Z}$, but could be defined by any similarity kernel as well. There are in principle three approaches to exploit similarity, if it exists: (i) by constraining the function class like in convolutional deep neural networks, (ii) by regularizing w.r.t. some predefined similarity kernel like in RKHS (see Section 3.1.2) or (iii) by generating *virtual samples* from a generative model. The latter can be seen as a special case of deductive learning. There are *many* possible structures in state spaces, which limits the applicability of the first two approaches for autonomous control. But how can one autonomously *learn* a generative model that exploits STRUCTURE in the state space? I want to explore this question at the example of COMBINATORIAL STATE SPACES. Imagine cars traveling on a highway. For tasks like traffic control, each car

COMBINATORIAL STATE SPACES

can be described by few variables: it's position, velocity, lane and, if a navigation computer is used, intended target. The state of the system consists of the combined descriptions of all cars on the road. As (almost) all combinations of state variables are allowed, this state space grows exponential in the number of cars. The transition model of this system, however, is highly structured by two effects: movement of cars depend almost exclusively on their own state (*conditional independence*) and the behavior of cars is only influenced by nearby cars (*locality*). I analyze these structures, and how they can be exploited for autonomous control, in more detail in Chapter 5. In the following I want to discuss instead the implications such structures have for representation learning.

DBN

Let us assume that the "true" transition model P is a DYNAMIC BAYESIAN NETWORK (DBN, Boutilier et al., 1999). DBN can be described as a product of transition models P^α which depend each only on a small subset $\boldsymbol{s}_\alpha$ of all variables:

$$P(d\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{a}) = \prod_{\alpha=1}^{d} P^\alpha(dx'_\alpha|\boldsymbol{s}_\alpha)\,, \quad \boldsymbol{s}_\alpha \subset \boldsymbol{x}\cup\boldsymbol{a}\,,\ \forall d\boldsymbol{x}' \in \mathcal{B}(\mathcal{X})\,,\ \forall \boldsymbol{x}\in\mathcal{X},\ \forall \boldsymbol{a}\in\mathcal{A}\,. \tag{4.14}$$

Each transition model P^α must only generalize over the domain of $\boldsymbol{s}_\alpha$, which in the above traffic control example is only the low-dimensional state of one car. A structured representation of $\mathcal{X}$ must find a mapping $\boldsymbol{\phi}:\mathcal{Z}\to\mathbb{R}^p$ from high-dimensional observations $\boldsymbol{z}\in\mathcal{Z}$, for example a collection of traffic cameras, into a space with similar conditionally independent variables. This could be achieved by using a *sparse connection* regularization (e.g. GROUP LASSO, Yuan and Lin, 2006) on the transition model learned by Watter et al. (2015). The regularization should ensure a suitable mapping $\boldsymbol{\phi}$, as the representation is learned together with the transition model. However, the transition structure in observation spaces like traffic camera images is highly non-linear and certainly not a DBN. Training of $\boldsymbol{\phi}$ by such a regularized auto-encoder therefore still requires to sample the entire observation space $\mathcal{Z}$, which is unrealistic in our scenario. Instead, one could borrow concepts from SPARSE CODING (Olshausen and Field, 1996) to determine *which* pixels influenced the prediction of a particular variable transition. Auto-encoder that ignore the "irrelevant" pixels for the encoding of that particular variable should be able to generalize over (almost) all combinatorics. This could allow to learn a representation without seeing all possible observations $\boldsymbol{z}\in\mathcal{Z}$ during training.

TASK-RELEVANT SUBSPACE

Moreover, such a well structured representation would allow to select a TASK-RELEVANT SUBSPACE *on-the-fly*. Controlling one particular traffic light, for example, requires at each given time only the knowledge of cars near the regulated crossroad. Instead of finding a stationary control for all possible *global* situations, the agent could compute a temporary policy that involves only the cars in the immediate vicinity. The same holds true for navigation in variable environments. Once a representation of walls, rooms and corridors has been learned, the agent could plan its navigation in the task-dependent subspace induced by the current floor-plan. In Chapter 5 I discuss these possibilities further and propose in Section 5.4.2 a mixture-of-expert DBN that can be parameterized by relational rules to generate task-relevant transition models.

Chapter 5

Generalization by Factorization

In order for people to rely on autonomous control, this control must *generalize* well to all anticipated scenarios. Moreover, the lack of supervision and the fact that users do not understand *how* a decision was made raises the bar for acceptable behavior. Expectations to perform as intended, even in *unexpected* situations, are much higher for autonomous agents that for traditional machines. Autonomous cars, for example, are held to much higher standards than traditional driver-assistance systems like adaptive cruise control. Chapter 2 introduces how control can be planned from rules or learned from experience, and Chapter 3 lays out how inductive machine learning generalizes to unseen situations using constrains and regularization. However, I have also shown in Chapter 4 that any inductive learning approach suffers the *curse of insufficient samples* in large state spaces $\mathcal{X}$ (Section 4.5.3). Almost any practical application has state and action spaces that are considered "large" by the current state of the art. To my knowledge, all successful applications of autonomous control relied on hand-crafted feature spaces (e.g. autonomous helicopters, Abbeel et al., 2007) or tightly controlled environments (e.g. autonomous robot navigation, Böhmer et al., 2013). At this point in time, inductive reasoning appears to be the limiting factor for generalization in autonomous control.

STRUCTURAL ASSUMPTIONS

In general, only STRUCTURAL ASSUMPTIONS can overcome the curse by carrying experiences over to "similar" states. These come in two flavors: states with similar reward or similar transitions. The former can be used in inductive learning[1] to either constrain the function space (Lemmas 3.4 and 3.5) or regularize the solution (Section 3.2.4). The latter, on the other hand, can be exploited for efficient deductive planning in large state-action spaces and will be the primary focus of this chapter. In the interest of more autonomy, Boutilier et al. (1999) identify three major areas to learn and/or exploit structural assumptions for control:

AGGREGATION

AGGREGATION aims to cluster states with similar "behavior" together. If the aggregated state space has the Markov property, a control policy can be learned in this much smaller space. This idea can be extended to *hierarchical RL* (see Section 2.3.1), where large parts of the state space can be traversed by executing one *option*. The traversed states are here aggregated due to the similar behavior of the options rather than the actions of the original transition model. Although planning is more efficient, options have to be learned for all states in $\mathcal{X}$ and the curse of insufficient samples plagues hierarchical RL as well.

[1] In Section 4.5.1 I gave an example for exploiting reward similarities by specifying a correlation kernel in γ-SFA (see Section 4.2.3). These kernels must be defined before training, though.

ABSTRACTION

ABSTRACTION uses *external knowledge* to express the state as relationships between symbols. Recent attempts to allow nondeterministic transitions in abstract, relational state spaces are known as RELATIONAL REINFORCEMENT LEARNING (RRL, Džeroski et al., 2001; Kersting et al., 2004; Sanner and Boutilier, 2009; Lang and Toussaint, 2010; Lang et al., 2012). For example, a robot arm may conceptualize the environment as being composed of regular objects that can be manipulated. Spatial relationships between objects and properties like weight constitute the abstract state of the system. These relationships give rise to transition rules that predict the change of relational predicates rather than states. For example, Pasula et al. (2007) show how nondeterministic relational rules can be learned from interaction with the environment. Given some suitable symbolic representation and rules, classical AI algorithms (like STRIPS, Fikes and Nilsson, 1971) or Bayesian inference algorithms (like PRADA, Lange and Riedmiller, 2010) can plan optimal control policies in large state spaces. Needless to say that learning such a symbolic representation of the environment is extremely difficult. In this thesis I do not consider how to learn symbolic abstractions, but use them for mixture models in Section 5.4.2.

RRL

FACTORIZATION

FACTORIZATION assumes that state variables are CONDITIONALLY INDEPENDENT of each other, akin to OCCHAM'S RAZOR in physics. Newton's first law of motion, for example, states that a body continues to move in the same direction unless acted upon by a force. The transition probability of the corresponding state variable is therefore almost always conditionally independent of all other bodies' states. The only exception[2] are, according to Newton's third law, collisions. When one body exerts a force on another, the latter exerts simultaneously an equal and opposite force on the former. Note that the transitions of the two bodies depend on each other, but remain conditionally independent from all other bodies. In fact, physics *requires* conditional independence, as a HOLISTIC approach could never be verified due to the exorbitant number of experiments necessary. We see here the universality of the curse of insufficient samples. Transition models with sparse conditional independence are called FACTORED MDP and will be discussed in Section 5.3.3. Newton's laws also demonstrate that realistic transition models never factorize *everywhere* the same way. This thesis proposes therefore a mixture of many factorizing transition models, based on conditions like "unless acted upon by a force" or "except for collisions". Conditions can often be expressed as logical predicates, and Section 5.4.2 proposes a relational parameterization of factored transition models to allow further generalization by symbolic abstraction.

In Section 5.1 I introduce how to combine inductive and deductive learning of control policies and how to exploit factorization with a novel function class. I show in Section 5.2 how to use this class for inductive learning like regression and value estimation. Section 5.3 introduces factored models and how they can be exploited to efficiently estimate deductive (Q-)values. As few environments have fully factorized transition models, Section 5.4 shows how to decompose a model into factorizing submodels, and how to parameterize the individual models using a symbolic abstraction of the environment. Section 5.5 takes a critical look at the prospects of generalization in autonomous control and some of the most challenging problems.

[2] Earth's gravitation affects all nearby bodies, but due to the difference in mass earth moves conditionally independent of them. As the gravitational force between two human-scale objects is negligible, only the electro-magnetic force, that bind the bodies' molecules together, yields during collisions a measurable effect (called the CONSERVATION OF MOMENTUM for rigid bodies).

5.1 Exploiting factorization

Using regularization or constrains to exploit structural knowledge in inductive learning requires to know how the target value of states will be correlated, as demonstrated in 4.2.3. Knowledge of the transitions in a state, however, does not allow us to directly forecast the state's value. Exploiting structured transition, factored or otherwise, is therefore more involved that exploiting structure in regression targets. I propose to *inductively* estimate the reward function $r' \in L^2(\mathcal{X} \times \mathcal{A}, \vartheta)$ and the transition model $P' : \mathcal{X} \times \mathcal{A} \times \mathcal{B}(\mathcal{X}) \to [0,1]$ from data. Given these models, one can *deductively* estimate the (Q-)value function. However, as I have discussed in Section 3.4.1, inductively learned models will always exhibit errors which can not be compensated by deductive reasoning alone. In Section 5.1.1 I sketch out how to integrate inductive and deductive learning into an approximate policy iteration framework. Probabilistic deductive learning in large state-action spaces is infeasible for many common function classes, like neural networks and kernel methods, and I derive a class of *linear factored functions* (LFF) that may be able to break the curse of insufficient samples in Section 5.1.2. Optimizing this function class yields non-convex cost functions and I derive a greedy coordinate descend approach for LFF in Sections 5.1.3 and 5.1.4. As in Section 3.2, I will use in the following either $\mathcal{Z} := \mathcal{X} \times \mathcal{A}$ with $\zeta := \xi\pi$, or $\mathcal{Z} := \mathcal{X}$ with $\zeta := \xi$, to avoid cluttered notation.

5.1.1 Combining inductive and deductive learning

To use both predicted and observed samples, Richard Sutton introduced the Dyna-Q framework (Sutton, 1990; Sutton and Barto, 1998). Sutton was mainly concerned with the slow propagation of reward information in classical TD-learning and learned a transition model to generate additional samples. This is very similar to my discussion of batch algorithms in Section 2.2.3 and to model based reinforcement learning for discrete state spaces in Section 3.3.1. However, these methods rely on discrete state spaces and suffer both curses of dimensionality and of insufficient samples. Instead I propose to estimate (Q-)values by splitting the cost function in two terms: states that are sampled often can be estimated inductively, and all other states are estimated deductively. As the sampling steady-state(-action) distribution $\zeta : \mathcal{B}(\mathcal{Z}) \to [0,1]$ is not a very suitable measure for optimization (see Section 4.5.2), I formulate value estimation w.r.t. the uniform distribution $\vartheta : \mathcal{B}(\mathcal{Z}) \to [0,1]$.

Let $\mathrm{td} := r + \gamma \hat{\mathrm{P}}^\pi[f] - f \in L^2(\mathcal{Z}, \vartheta)$ denote the observed temporal difference error of estimated (Q-)value function $f \in L^2(\mathcal{Z}, \vartheta)$, and $\mathrm{td}' := r' + \gamma \hat{\mathrm{P}}'^\pi[f] - f \in L^2(\mathcal{Z}, \vartheta)$ the error estimated using the learned models. Experienced errors $\mathrm{td}(\cdot)$ are more reliable than predicted $\mathrm{td}'(\cdot)$. Ideally, (Q-)value estimation would rely on the experienced errors wherever enough samples are available, and fall back on the predicted error otherwise. One must therefore find a partition function[3] $p \in L^2(\mathcal{Z}, \vartheta)$, with $p(z) = 1$ for all sufficiently known state(-action)s $z \in \mathcal{Z}$, and $0 \leq p(z) < 1$ for all other state(-action)s. I propose that z is sufficiently known when the Radon-Nikodym derivative $\frac{d\zeta}{d\vartheta}(z)$, also known as the PROBABILITY DENSITY FUNCTION (PDF), is larger than some "regularization constant" $\lambda \in [0, \infty) \subset \mathbb{R}$. Here I define[4]

[3] Note the similarity of $p(\cdot)$ with the "set of known states" in E^3 (Kearns and Singh, 2002).

[4] One could also define the partition function as 1, if $\frac{d\zeta}{d\vartheta}(z) \geq \lambda$, or 0 otherwise. The resulting importance sampling function in Equation 5.2 is $\tilde{p}(z) = 1/\frac{d\zeta}{d\vartheta}(z)$, if $\frac{d\zeta}{d\vartheta}(z) \geq \lambda$, or 0 otherwise.

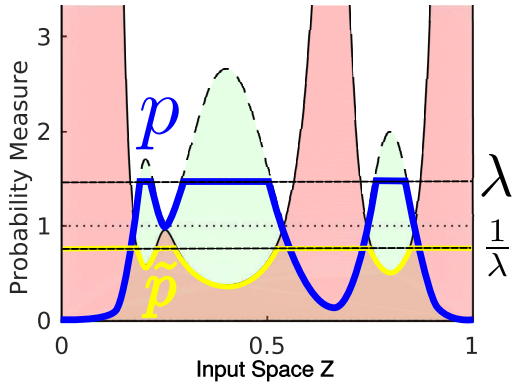


Figure 5.1: The partition function p (scaled by λ, in blue) and the resulting importance sampling function $\tilde{p}$ (in yellow) at the example of the uncertainty measure of Figure 3.3.

the partition function as $p(z) := \frac{1}{\lambda} \min(\frac{d\zeta}{d\vartheta}(z), \lambda), \forall z \in \mathcal{Z}$. At each state $z \in \mathcal{Z}$, the quadratic error $\mathrm{td}^2(z)$ is approximated with $p(z)\,\mathrm{td}^2(z) + (1-p(z))\,\mathrm{td}'^2(z)$, and thus

$$\begin{aligned} \|\mathrm{td}\|^2_\vartheta &\approx \|\mathrm{td}'\|^2_\vartheta + \int \zeta(dz) \overbrace{\tfrac{d\vartheta}{d\zeta}(z)\,\tfrac{1}{\lambda}\,\min(\tfrac{d\zeta}{d\vartheta}(z), \lambda)}^{\tilde{p}(z)} \left(\mathrm{td}^2(z) - \mathrm{td}'^2(z)\right) \qquad (5.1) \\ &\equiv \|\mathrm{td}'\|^2_\vartheta + \|\sqrt{\tilde{p}} \cdot \mathrm{td}\|^2_\zeta - \|\sqrt{\tilde{p}} \cdot \mathrm{td}'\|^2_\zeta . \end{aligned}$$

As the derivative $\frac{d\vartheta}{d\zeta}$ is the inverse function[5] of $\frac{d\zeta}{d\vartheta}$, one can rewrite the *effective* IMPORTANCE SAMPLING function $\tilde{p} \in L^2(\mathcal{Z}, \vartheta)$ as

IMPORTANCE SAMPLING

$$\tilde{p}(z) := \begin{cases} 1/\lambda & , \text{ if } \frac{d\zeta}{d\vartheta}(z) < \lambda \\ 1/\frac{d\zeta}{d\vartheta}(z) & , \text{ if } \frac{d\zeta}{d\vartheta}(z) \geq \lambda \end{cases} . \qquad (5.2)$$

Figure 5.1 illustrates both the partition function p and the importance sampling function $\tilde{p}$. The approximated cost function in Equation 5.1 can be interpreted as the deductive cost that is *corrected* by the difference between two inductive costs, each using important weights $\tilde{p}$. For $\lim \lambda \to 0$ the cost function contains only the inductive $\|\sqrt{\frac{d\zeta}{d\vartheta}} \cdot \mathrm{td}\|^2_\zeta$. For $\lim \lambda \to \infty$, on the other hand, the cost function relies purely on deductive predictions $\|\mathrm{td}'\|^2_\vartheta$.

There are two obstacles to estimate the above cost function. First, the importance weights rely on the PDF $\frac{d\zeta}{d\vartheta}$ of the sampling distribution ζ. While many other partition functions are imaginable, each must in some respect rely on the empirical sampling density ζ of the training data. Secondly, the deductive term $\|\mathrm{td}'\|^2_\vartheta$ requires to compute or estimate a uniform integral over all state(-action)s in $\mathcal{Z}$. All function classes discussed in Chapter 3 suffer the curse of dimensionality for uniform integrals. To minimize the cost of Equation 5.1 *efficiently*, one needs a class of parameterizable functions that (i) solve integrals w.r.t. uniform distribution ϑ in sub-exponential complexity, (ii) allow to solve integrals over factored models P' efficiently and (iii) allows to estimate the PDF $\frac{d\zeta}{d\vartheta}$ from data. In the next subsection I define a function class that has all of these properties.

5.1.2 Linear factored functions (LFF)

Let $\mathcal{Z} := \mathcal{Z}_1 \times \ldots \times \mathcal{Z}_d$ denote a d-dimensional multivariate state(-action) space. Let furthermore ϑ be a FACTORIZING DISTRIBUTION over $\mathcal{Z}$, i.e. $\vartheta := \prod_{\alpha=1}^d \vartheta^\alpha$. In

[5] In this chapter I assume that the inverse technically exists, that is, $\frac{d\zeta}{d\vartheta}(z) > 0, \forall z \in \mathcal{Z}$. This is true for empirical estimates w.r.t. training samples $\{z_t\}_{t=1}^n \sim \zeta$, which indicates $\frac{d\zeta}{d\vartheta}(z_t) > 0, \forall t$.

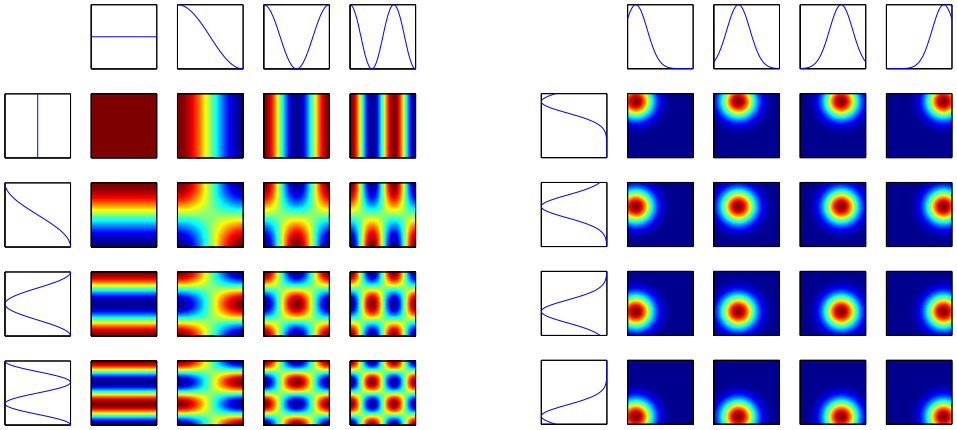


Figure 5.2: Two examples for two-dimensional factored functions ψ_i based on one-dimensional basis functions ϕ_j, that in the limit $m \to \infty$ can express almost any continuous function. Fourier-cosine functions on the left are a basis of $L^2(\mathcal{Z}, \vartheta)$ and Gaussian kernels on the right are a basis of the RKHS $\mathcal{H}_\kappa$ of the kernel κ. Note that the number m of d-dimensional basis functions ψ_i is in both cases exponential in the number of dimensions d, in this example $m = \prod_{\alpha=1}^{d} m_\alpha = 4^2$.

this thesis I will exclusively use the example of ϑ being the uniform distribution over $\mathcal{Z}$, which is a factorizing distribution, but other definitions like diagonal Gaussian distributions are possible, too. A FACTORED FUNCTION $g \in \mathcal{F} \subset L^2(\mathcal{Z}, \vartheta)$ is defined as the product[6] of d one-dimensional FACTOR FUNCTIONS $g^\alpha \in L^2(\mathcal{Z}_\alpha, \vartheta^\alpha)$. Due to FUBINI'S THEOREM, inner products of factored functions g and h in $L^2(\mathcal{Z}, \vartheta)$ can be written as the product of inner products in $L^2(\mathcal{Z}_\alpha, \vartheta^\alpha), \forall \alpha \in \{1, \ldots, d\}$:

FACTORED FUNCTIONS

$$\langle g, h \rangle_\vartheta \quad = \quad \prod_{\alpha=1}^{d} \langle g^\alpha, h^\alpha \rangle_{\vartheta^\alpha} \,, \qquad \forall f, g \in \mathcal{F} \subset L^2(\mathcal{Z}, \vartheta) \,. \tag{5.3}$$

The complexity of estimating this d-dimensional integral by sampling drops from exponential to linear in d. However, only a small subset of functions $f \in L^2(\mathcal{Z}, \vartheta)$ can be expressed as factored functions. I address this shortcoming in the following by using linear combinations of factored functions.

I define LINEAR FACTORED FUNCTIONS (LFF) $f \in \mathcal{F}^m \subset L^2(\mathcal{Z}, \vartheta)$ as a linear combination of m factored functions $\psi_i \in \mathcal{F}$. Each factored function ψ_i is the product of d factor functions $\psi_i^\alpha \in L^2(\mathcal{Z}_\alpha, \vartheta^\alpha), \forall \alpha \in \{1, \ldots, d\}$, which are linear combinations of m_α one-dimensional basis functions $\phi_j^\alpha \in L^2(\mathcal{Z}_\alpha, \vartheta^\alpha)$, that is, $\forall \boldsymbol{z} \in \mathcal{Z}$:

LFF

$$f(\boldsymbol{z}) \;:=\; \sum_{i=1}^{m} a_i \, \psi_i(\boldsymbol{z}) \;:=\; \sum_{i=1}^{m} a_i \prod_{\alpha=1}^{d} \psi_i^\alpha(z_\alpha) \;:=\; \sum_{i=1}^{m} a_i \prod_{\alpha=1}^{d} \sum_{j=1}^{m_\alpha} B_{ji}^\alpha \, \phi_j^\alpha(z_\alpha) \,. \tag{5.4}$$

Each LFF $f \in \mathcal{F}^m$ has the parameters $\boldsymbol{a} \in \mathbb{R}^m$ and $\{\mathbf{B}^\alpha \in \mathbb{R}^{m_\alpha \times m}\}_{\alpha=1}^{d}$.

[6] In this thesis I indicate the dimension of factored products by raised Greek indices, e.g. g^α.

Corollary 5.1 (from Böhmer and Obermayer, 2015) *Let all $\mathcal{Z}_k$ be bounded continuous sets and ϕ_j^k the j'th Fourier base over $\mathcal{Z}_k$. In the limit of $m_\alpha \to \infty, \forall \alpha \in \{1,\dots,d\}$, holds $\mathcal{F}^\infty = L^2(\mathcal{Z},\vartheta)$.*

Corollary 5.1 shows, at the example of Fourier bases, that in the limit of infinitely many basis functions, LFF can represent any function in $L^2(\mathcal{Z},\vartheta)$. Similar statements can be derived for Gaussian kernels and all functions from the corresponding RKHS. Figure 5.2 demonstrates how one-dimensional basis functions ϕ_j^α can construct a full basis $\{\psi_i\}$ of a two dimensional state(-action) space. Although LFF are thus powerful function classes, Fubini's theorem still applies to inner products:

$$\langle f, f' \rangle_\vartheta \quad = \quad \boldsymbol{a}^\top \Big[\prod_{\alpha=1}^{d} \mathbf{B}^{\alpha\top} \underbrace{\langle \boldsymbol{\phi}^\alpha, \boldsymbol{\phi}^{\alpha\top} \rangle_{\vartheta^\alpha}}_{\mathbf{C}^\alpha \in \mathbb{R}^{m_\alpha \times m_\alpha}} \mathbf{B}'^\alpha \Big] \boldsymbol{a}' \,, \qquad \forall f \in \mathcal{F}^m \,, \quad \forall f' \in \mathcal{F}^{m'} \,. \tag{5.5}$$

FOURIER BASES

Inner products $\langle \phi_i^\alpha, \phi_j^\alpha \rangle_{\vartheta^\alpha}$ can be solved analytical for many bases ϕ_j^α. In the rest of this thesis I use in particular FOURIER COSINE BASES over $\mathcal{Z}_\alpha := [a,b] \subset \mathbb{R}$:

$$\phi_0^\alpha(z_\alpha) = 1 \,, \qquad \phi_j^\alpha(z_\alpha) = \sqrt{2}\, \cos(j\pi\, \tfrac{z_\alpha - a}{b-a}) \,, \qquad \forall j \in \mathbb{N} \,, \qquad \forall z_\alpha \in \mathcal{Z}_\alpha \,, \tag{5.6}$$

to represent continuous variables that have a minimum and a maximum. Continuous variables that represent a circular value, like the orientation of a robot, are instead often represented by FULL FOURIER BASES over $\mathcal{Z}_\alpha := [a,b) \subset \mathbb{R}$:

$$\phi_0^\alpha(z_\alpha) = 1 \,, \quad \begin{array}{ll} \phi_j^\alpha(z_\alpha) = \sqrt{2}\, \cos(2j\pi\, \frac{z_\alpha - a}{b-a}) & \text{for even } j \\ \phi_j^\alpha(z_\alpha) = \sqrt{2}\, \sin((2j{+}1)\pi\, \frac{z_\alpha - a}{b-a}) & \text{for odd } j \end{array} \,, \quad \begin{array}{l} \forall j \in \mathbb{N} \\ \forall z_\alpha \in \mathcal{Z}_\alpha \end{array} \,. \tag{5.7}$$

Both sets of Fourier bases have an analytic mean and covariance matrix:

$$C_{ij}^\alpha := \langle \phi_i^\alpha, \phi_j^\alpha \rangle_{\vartheta^\alpha} = \delta_{ij} \,, \qquad \mu_j^\alpha := \langle \phi_j^\alpha, 1 \rangle_{\vartheta^\alpha} = \delta_{j0} \,, \qquad \forall i,j \in \mathbb{N} \,. \tag{5.8}$$

DISCRETE BASES

Variables with only a finite set of possible states can be encoded by DISCRETE BASES

$$\phi_i^\alpha(z) := \sqrt{m_\alpha}\, \delta_{iz} \,, \quad \text{with} \quad C_{ij}^\alpha = \delta_{ij} \quad \text{and} \quad \mu_j^\alpha = 1/\sqrt{m_\alpha} \,, \quad \forall i,j,z \in \mathcal{Z}_\alpha \,. \tag{5.9}$$

MARGINALS

LFF inherit many desirable properties from linear functions, like analytical SCALAR MULTIPLICATION and POINT-WISE ADDITION. Moreover, integrals over individual dimensions, called MARGINALIZATION, can be computed analytically, i.e. $\forall f \in \mathcal{F}^m$:

$$\begin{aligned} \int \vartheta^\beta(dz_\beta)\, f(\boldsymbol{z}) \quad &= \quad \sum_{i=1}^{m} \Big(a_i \sum_{j=1}^{m_\beta} B_{ji}^\beta \overbrace{\langle \phi_j^\beta, 1 \rangle_{\vartheta^\beta}}^{\text{mean of } \phi_j^\beta} \Big) \Big[\prod_{\alpha \neq \beta}^{d} \psi_i^\alpha(z_\alpha) \Big] \\ &\overset{\text{Fourier}}{=} \quad \sum_{i=1}^{m} \underbrace{a_i B_{0i}^\beta}_{\text{new } a_i} \Big[\prod_{\alpha \neq \beta}^{d} \psi_i^\alpha(z_\alpha) \Big] \,. \end{aligned} \tag{5.10}$$

POINT-WISE MULTIPLICATION

Last, but not least, LFF based on the above bases can be POINT-WISE MULTIPLIED. For example, Fourier-cosine functions can use the trigonometric *product-to-sum* identity[7] $\cos(x)\cdot\cos(y) = \frac{1}{2}\big(\cos(x-y)+\cos(x+y)\big)$ to solve the point wise product $\phi_j^\alpha \cdot \phi_q^\alpha$ (solutions to other Fourier bases are possible but less elegant), $\forall f \in \mathcal{F}^m, \forall \bar{f} \in \mathcal{F}^{\bar{m}}$:

[7] Note the different scaling: $\phi_i^\alpha(x) \cdot \phi_j^\alpha(x) = \begin{cases} \phi_{i+j}^\alpha(x) & , \text{ if } i = 0 \vee j = 0 \\ \phi_{i-j}^\alpha(x) + \frac{1}{\sqrt{2}} \phi_{i+j}^\alpha(x) & , \text{ if } i > 0 \wedge i = j \\ \frac{1}{\sqrt{2}} \phi_{i-j}^\alpha(x) + \frac{1}{\sqrt{2}} \phi_{i+j}^\alpha(x) & , \text{ otherwise} \,. \end{cases}$

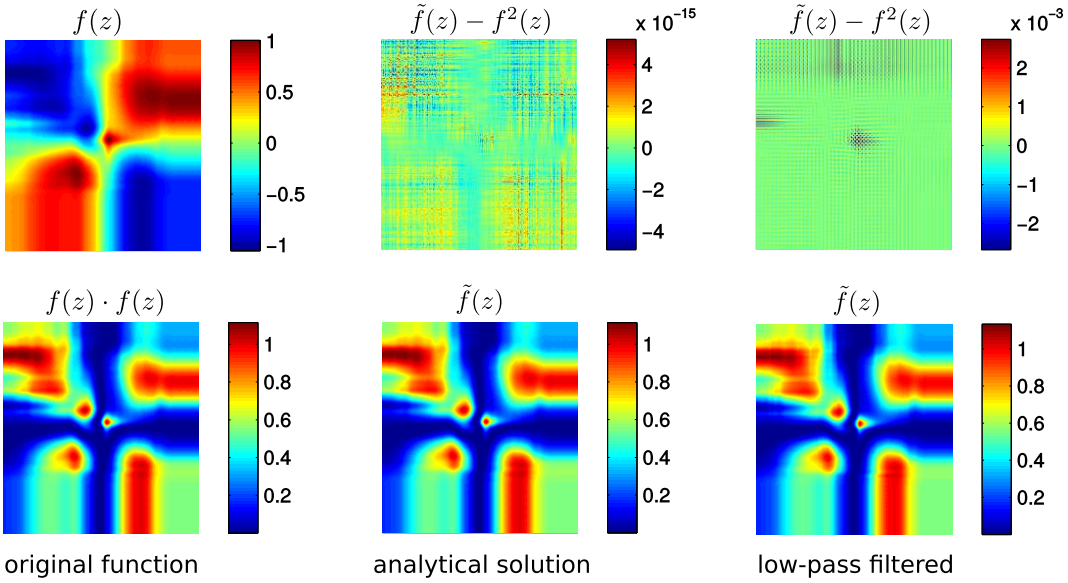


Figure 5.3: An example of point-wise multiplication of LFF $f \in \mathcal{F}^{12}$, $m_\alpha = 50$, (upper left) with itself (lower left). The middle column shows the analytical solution $\tilde{f} \in \mathcal{F}^{144}$, $m_\alpha = 100$, according to Equation 5.11 (below) and the corresponding error (above). The right column shows $\tilde{f}$ and the corresponding error after a low-pass filter reduced the Fourier bases in each dimension back to the original $m_\alpha = 50$.

$$\tilde{f}(\boldsymbol{z}) \quad := \quad f(\boldsymbol{z}) \cdot \bar{f}(\boldsymbol{z}) \quad = \quad \sum_{k=1}^{m}\sum_{l=1}^{\bar{m}} a_k\, \bar{a}_l \prod_{\alpha=1}^{d} \sum_{j,q=1}^{m_\alpha} B^\alpha_{jk}\, \bar{B}^\alpha_{ql}\, [\phi^\alpha_j \cdot \phi^\alpha_q](z_\alpha) \tag{5.11}$$

$$\overset{\text{Fourier}}{=} \sum_{k=1}^{m}\sum_{l=1}^{\bar{m}} \underbrace{a_k\, \bar{a}_l}_{\text{new } \tilde{a}_i} \prod_{\alpha=1}^{d} \sum_{j=1}^{2m_\alpha} \underbrace{\Big(\tfrac{1}{2}\sum_{q=1}^{j-1} B^\alpha_{qk}\, \bar{B}^\alpha_{(j-q)l} + \tfrac{1}{2}\sum_{q=j+1}^{m_\alpha} B^\alpha_{qk}\, \bar{B}^\alpha_{(q-j)l} \Big)}_{\text{new } \tilde{B}^\alpha_{ji}} \phi^\alpha_j(z_\alpha)\,.$$

Note that this multiplication increases the number of factored basis functions $\tilde{m} = m\,\bar{m}$ and the number of Fourier bases in each dimension $\tilde{m}_\alpha = 2m_\alpha$. To project the resulting function $\tilde{f}$ back in the original function space of f and $\bar{f}$, $\tilde{f}$ can be LOW-PASS FILTERED by setting $B^\alpha_{ji} := 0, \forall j > m_\alpha, \forall \alpha \in \{1, \ldots, d\}$. The resulting function has $\tilde{m}_\alpha = m_\alpha$. Figure 5.3 shows the example of a LFF that is multiplied with itself. Note that the analytical solution for $\tilde{f}$ in the middle column has approximation errors close to the system's accuracy ($\approx 10^{-16}$). Although the low-pass filtered function in the right column has much higher approximation errors (upper row), the two functions look almost identical to the eye (lower row). Reducing the number of factored bases $\tilde{m}$ is less trivial and requires a *compression* algorithm, which I develop in the next subsection. All above analytical operations with and between LFF are summarized in Table 5.1 on the next page.

LOW-PASS FILTER

LFF can also estimate factored transition operators (see Equation 4.14 on Page 72)

$$\hat{\mathrm{P}}[f](\boldsymbol{z}) \quad = \quad \sum_{i=1}^{m} a_i \prod_{\alpha=1}^{d} \sum_{j=1}^{m_\alpha} B^\alpha_{ji}\, \hat{\mathrm{P}}^\alpha[\phi^\alpha_j](\boldsymbol{s}^\alpha)\,, \qquad \forall f \in \mathcal{F}^m\,. \tag{5.12}$$

Here each $\hat{\mathrm{P}}^\alpha[\phi^\alpha_j]$ can be estimated from training data by *regression* (see Section 3.2) over the parent-variables of state(-action) dimension α. For a given sparse DBN,

Mathematical operation	$\tilde{a}_i$	$\tilde{B}^\alpha_{ji}$	Notes
scalar multiplication $\tilde{f}(z) := s\ f(\boldsymbol{z})$	$s\ a_i$	B^α_{ji}	$s \in \mathbb{R}$
point-wise addition $\tilde{f}(z) := f(\boldsymbol{z}) + \bar{f}(\boldsymbol{z})$	$\begin{matrix} a_i & , i \leq m \\ \bar{a}_{i-m} & , i > m \end{matrix}$	$\begin{matrix} B^\alpha_{ji} & , i \leq m \\ \bar{B}^\alpha_{j(i-m)} & , i > m \end{matrix}$	$i \in \{1,\dots,m+\bar{m}\}$
marginalization (Fourier cosine bases) $\tilde{f}(z) := \int \vartheta^\beta(dz_\beta)\, f(\boldsymbol{z})$	$a_i\, B^\beta_{1i}$	B^α_{ji}	$\forall \alpha \neq \beta$
point-wise multiplication (Fourier cosine bases) $\tilde{f}(z) := f(\boldsymbol{z}) \cdot \bar{f}(\boldsymbol{z})$	$a_k\, \bar{a}_l$	$\frac{1}{2}\sum_{q=1}^{j-1} B^\alpha_{ql}\, \bar{B}^\alpha_{(j-q)k} + \frac{1}{2}\sum_{q=j+1}^{m_k} B^\alpha_{ql}\, \bar{B}^\alpha_{(q-j)k}$	$\begin{matrix} i = (m-1)l+k \\ i \in \{1,\dots,m\bar{m}\} \\ j \in \{1,\dots,2m_k\} \end{matrix}$
marginalization (discrete bases) $\tilde{f}(z) := \int \vartheta^\beta(dz_\beta)\, f(\boldsymbol{z})$	$a_i \frac{(\mathbf{1}^\top \mathbf{B}^\beta)_i}{\sqrt{m_\alpha}}$	B^α_{ji}	$\forall \alpha \neq \beta$
point-wise multiplication (discrete bases) $\tilde{f}(z) := f(\boldsymbol{z}) \cdot \bar{f}(\boldsymbol{z})$	$a_k\, \bar{a}_l$	$\sqrt{m_\alpha}\, B^\alpha_{jk}\, \bar{B}^\alpha_{jl}$	$\begin{matrix} i = (m-1)l+k \\ i \in \{1,\dots,m\bar{m}\} \\ j \in \{1,\dots,m_k\} \end{matrix}$

Table 5.1: Analytical operations on and between LFF with Fourier/discrete bases.

the factored transition model can thus be learned efficiently as a set of LFF (the algorithm is developed in Section 5.2.2). The multiple applications of the Bellman operator (see Section 2.2.2) yield therefore a deep network of product- and sum-nodes, similar to *sum-product networks* (Poon and Domingos, 2011; Peharz et al., 2015). Using the analytical point-wise product above together with the LFF compression in the next section, however, the deep network can be flattened to a single LFF $f' \in \mathcal{F}^{m'}$. This allows to minimize the deductive cost $\|\mathrm{td}'\|^2_\vartheta \approx \|f - r - \gamma f'\|^2_\vartheta$ in Equation 5.1 in sub-exponential time without the need to sample the entire state (-action) space $\mathcal{Z}$. See Section 5.3.4 for details.

5.1.3 Compression of LFF

Analytical operations like point-wise addition and multiplication can increase the number of factored basis functions m enormously. Many of these bases are very similar though, which makes them redundant. In this section I develop an algorithm to COMPRESS $f \in \mathcal{F}^m$, that is, to find a LFF $\tilde{f} \in \mathcal{F}^{\tilde{m}}$ with $\|f - \tilde{f}\|^2_\vartheta < \epsilon$ and $\tilde{m} \ll m$. Note that such a function $\tilde{f}$ is invariant to the norm of its factored basis functions $\|\tilde{\psi}_i\|_\vartheta = \prod_{\alpha=1}^d \|\tilde{\psi}^\alpha_i\|_{\vartheta^\alpha}$, which can always be compensated by the linear parameter a_i. Furthermore, the magnitude of each factor function $\|\psi^\alpha_i\|_{\vartheta^\alpha}$ is also arbitrary, as long as the product is constant. I remove these degrees of freedom by introducing the constraints $\|\psi^\alpha_i\|_{\vartheta^\alpha} \stackrel{!}{=} 1, \forall i \in \{1,\dots,m\}, \forall \alpha \in \{1,\dots,d\}$.

The last degree of freedom, the order of the factored bases, can be removed by a GREEDY approximation procedure. Algorithm 5 sketches the approach. Here the set of basis function is build incrementally by minimizing the approximation error

Algorithm 5 (abstract) – greedy LFF compression, for details see Algorithm 6.

```
input:  f ∈ F^m                                   // the function to be compressed
while  ‖f − f̃‖²_ϑ > ϵ  do
  g := 1;   g′ := ∞                               // initialize new basis function g
  while  ‖f − f̃ − g′‖²_ϑ − ‖f − f̃ − g‖²_ϑ > ε  do
    g′ := g
    for α in randperm{1, ..., d} do
      g^α := arg min_{g^α} ‖f − f̃ − g^α · ∏_{β≠α} g′^β‖²_ϑ   // update factor function g^α
    end for
  end while                                       // end inner loop: found new basis g
  {ψ̃_i} := {ψ̃_i} ∪ {g}                            // add g to set of bases {ψ̃_i} of f̃
  f̃ := arg min_{ã} ‖f − ã^⊤ ψ̃‖²_ϑ                // recompute lin. parameter ã of f̃
end while                                         // end outer loop: f̃ approximates f
output:  f̃ ∈ F^m̃                                  // the compressed function f̃
```

of factored function $g \in \mathcal{F}$ with the RESIDUAL $f - \tilde{f}$ of current estimate $\tilde{f}$. The optimization problem for the next basis function g is:

$$\inf_{g \in \mathcal{F}} \|f - \tilde{f} - g\|_\vartheta^2 \qquad \text{s.t.} \qquad \|g^\alpha\|_{\vartheta^\alpha} = 1\,, \quad \forall \alpha \in \{1, \dots, d\}\,. \tag{5.13}$$

After a suitable basis has been found, the linear parameters $\tilde{\boldsymbol{a}}$ are optimized with OLS (see Section 3.2.1). This procedure is repeated in an OUTER LOOP until the final convergence criterion $\|f - \tilde{f}\|_\vartheta^2 < \epsilon$ is fulfilled (first `while` loop of Algorithm 5). OUTER LOOP

Due to the non-linear product in $g(\boldsymbol{z}) = \prod_{\alpha=1}^d g^\alpha(z_\alpha) = \prod_{\alpha=1}^d \boldsymbol{b}^{\alpha\top} \boldsymbol{\phi}^\alpha(z_\alpha), \forall \boldsymbol{z} \in \mathcal{Z}$, the cost function in Equation 5.13 is not convex, though. Instead of a costly gradient descend procedure, I use a COORDINATE DESCEND approach (Wright, 2015) to optimize one dimension at a time. Each optimization step is convex and has an analytical solution. When changing only factor function g^α, all other factor functions g^β remain the same as in the old function g', that is, $\boldsymbol{b}^\beta = \boldsymbol{b}'^\beta, \forall \beta \neq \alpha$. Setting the gradient w.r.t. $\boldsymbol{b}^\alpha$ to zero yields the unconstrained solution $\boldsymbol{b}_{\text{uc}}$. This solution can be projected onto the constraint $\|g^\alpha\|_{\vartheta^\alpha} = 1$ by normalization.

$$\frac{\partial}{\partial \boldsymbol{b}^\alpha} \|f - g\|_\vartheta^2 \;=\; 2\,\mathbf{C}^\alpha \boldsymbol{b}^\alpha \Big(\prod_{\beta \neq \alpha} \overbrace{\|g^\beta\|_{\vartheta^\beta}^2}^{1} \Big) + 2\,\mathbf{C}^\alpha \mathbf{B}^\alpha \Big(\boldsymbol{a} \cdot \prod_{\beta \neq \alpha} \overbrace{\mathbf{B}^{\beta\top} \mathbf{C}^\beta \boldsymbol{b}'^\beta}^{\boldsymbol{c}^\beta \in \mathbb{R}^m} \Big)\,, \tag{5.14}$$

$$\Rightarrow \quad \boldsymbol{b}_{\text{uc}}^\alpha \;=\; \mathbf{B}^\alpha \Big(\boldsymbol{a} \cdot \prod_{\beta \neq \alpha} \boldsymbol{c}^\beta \Big) \qquad \text{and} \qquad \boldsymbol{b}^\alpha \;=\; \frac{\boldsymbol{b}_{\text{uc}}^\alpha}{\sqrt{\boldsymbol{b}_{\text{uc}}^{\alpha\top} \mathbf{C}^\alpha \boldsymbol{b}_{\text{uc}}^\alpha}}\,. \tag{5.15}$$

Note that the solution depends on COEFFICIENTS $c_i^\beta := \langle \psi_i^\beta, g'^\beta \rangle_{\vartheta^\beta}$, which represent the correlation of g' with all factored bases ψ_i of f. This optimization step is repeated in an INNER LOOP for random state(-action) dimensions α. Convergence of the inner loop is not trivial. I observed that parameter vectors $\{\boldsymbol{b}^\alpha\}_{\alpha=1}^d$ evolve along chaotic attractors and may never converge to a single solution. As each optimization step minimizes the overall cost function, however, the cost of the sequence of functions g *must* converge. Ignoring the risk of finding saddle-points, the inner loop terminates when the cost improvement for all dimensions falls below some threshold ε. The improvement d_α by an update from g' to g is: INNER LOOP

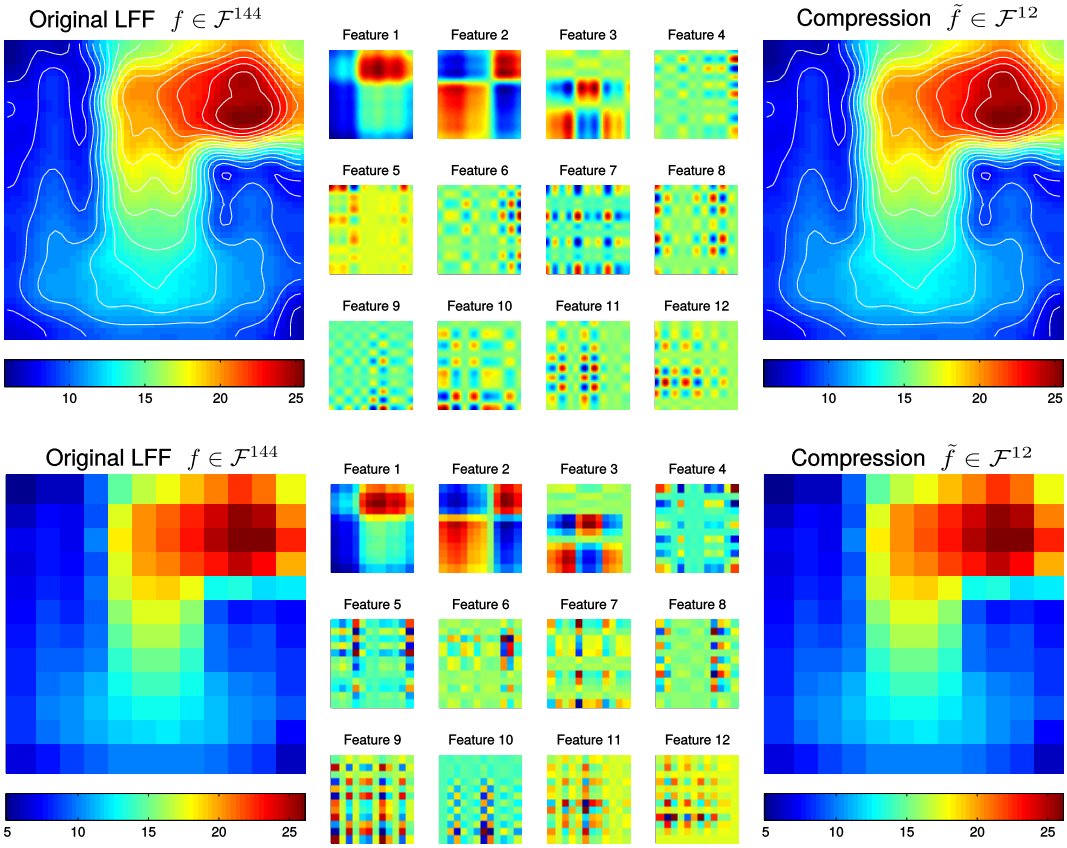


Figure 5.4: Compression of LFF with Algorithm 6. The original function (left) is represented with 12 Fourier (upper row) or discrete (lower row) bases ϕ_j^α in both dimensions α. Factored basis functions ψ_i are shown in the middle, the compressed functions $\tilde{f}$ on the right. Note that the first 3 factored bases ψ_i look almost identical.

$$d_\alpha \quad := \quad \|f - g'\|_\vartheta^2 - \|f - g\|_\vartheta^2 \quad = \quad 2\,(\boldsymbol{b}^\alpha - \boldsymbol{b}'^\alpha)^\top \mathbf{C}^\alpha\, \boldsymbol{b}_{\text{uc}}^\alpha\,. \tag{5.16}$$

To ensure that all "randomly" chosen[8] dimensions α have been updated, the `for` loop iterates over a random permutation of all dimensions. The complete procedure with all details is given in Algorithm 6.

Figure 5.4 shows a typical continuous value function $f \in \mathcal{F}^{144}$ that has been learned with a full basis of 12 Fourier cosine waves (upper row) or discrete identity functions (lower row) in both dimensions. The compressed functions $\tilde{f}$ are virtually indistinguishable from their originals f by the naked eye. Note also that the first 3 factored bases (in the middle) are very similar, irrespective of their underlying basis ϕ_i^α. This indicates that the optimization procedure is very stable w.r.t. the underlying basis and the order of updates.

In summary, the LFF compression of Algorithm 6 appears to be robust, efficient and, in comparison with inductive algorithms in Section 5.2, relatively fast. The error introduced by compression is guaranteed to be below a threshold ϵ, which can be provided as a hyper-parameter.

[8] Random choice is used here for simplicity. Recent experiments indicate that compression can be sped up by choosing the dimension α with the largest last improvement, i.e. $\alpha := \arg\max_\alpha\{d_\alpha\}$. Many algorithms in this chapter use this improved scheme to speed up computation.

Algorithm 6 – greedy LFF compression, e.g. after a point-wise multiplication

```
input:  B^α ∈ ℝ^{m_α × m}, ∀α,   a ∈ ℝ^m                    // input function f
B̃^α := ∅,  ∀α;   ã := ∅;   err := ∞                        // initialize output f̃
while  err > ε  do
  c^α := 1 ∈ ℝ^{m+|ã|};   d_α := ∞,  ∀α                     // initialize coefficients
  while  max_α{d_α} > ε  do
    for α in randperm{1,...,d} do
      b_uc := [B^α, B̃^α]( (∏_{β≠α} c^β) · [ a ; −ã ] )      // unconstrained solution for g'^α
      d_α := 2 b_uc^⊤ C^α b^α                               // d_α = ‖f − g'‖²_ϑ − ‖g'‖²_ϑ − ‖f‖²_ϑ
      b^α := b_uc / sqrt(b_uc^⊤ C^α b_uc)                   // normalize functions to length 1
      d_α := 2 b_uc^⊤ C^α b^α − d_α                         // d_α = ‖f − g'‖²_ϑ − ‖f − g‖²_ϑ
      c^α := [B^α, B̃^α]^⊤ C^α b^α                           // recompute coefficients
    end for                                                 // end for loop:  all g^α updated
  end while                                                 // end inner loop:  g is new basis
  B̃^α := [B̃^α, b^α],  ∀α;   ã := (∏_α B̃^{α⊤} C^α B̃^α)^{−1} (∏_α B̃^{α⊤} C^α B^α) a   // update f̃
  err := [ a ; −ã ]^⊤ ( ∏_α [B^α, B̃^α]^⊤ C^α [B^α, B̃^α] ) [ a ; −ã ]   // error err = ‖f − f̃‖²_ϑ
end while                                                   // end outer loop:  ‖f − f̃‖²_ϑ ≤ ε
output:  B̃^α ∈ ℝ^{m_α × m̃}, ∀α,   ã ∈ ℝ^{m̃}
```

5.1.4 Compression of multiple LFF

In Section 5.3.3 I demonstrate how to represent transition models P^α with a set of m' LFF $f_k \approx \hat{\mathrm{P}}^\alpha[\phi_k^\alpha]$. Later computations can be greatly simplified when all LFF have the same m basis functions $\tilde{\psi}_i \in \mathcal{F}$, that is, when they are *one* MULTIVARIATE LFF $\tilde{\boldsymbol{f}} \in \mathcal{F}^{m \times m'}$ with parameter matrices $\tilde{\mathbf{A}} \in \mathbb{R}^{m \times m'}$ and $\{\tilde{\mathbf{B}}^\alpha \in \mathbb{R}^{m_\alpha \times m}\}_{\alpha=1}^d$: MULTIVARIATE LFF

$$\tilde{\boldsymbol{f}}(\boldsymbol{z}) \;:=\; \tilde{\mathbf{A}}^\top \tilde{\boldsymbol{\psi}}(\boldsymbol{z}) \;=\; \tilde{\mathbf{A}}^\top \Big(\prod_{\alpha=1}^{d} \tilde{\mathbf{B}}^{\alpha\top} \boldsymbol{\phi}^\alpha(z_\alpha) \Big) \;\in\; \mathbb{R}^{m'}, \qquad \forall \boldsymbol{z} \in \mathcal{Z}\,. \tag{5.17}$$

Therefore, I propose a method to compress multiple LFF $\{f_k \in \mathcal{F}^{m_k}\}_{k=1}^{m'}$ into one multivariate LFF. The greedy optimization problem for the next basis function g is

$$\inf_{g \in \mathcal{F}} \; \underbrace{\tfrac{1}{2d} \sum_{k=1}^{m'} \big\| f_k - \tilde{f}_k - g \big\|_\vartheta^2}_{\hat{\mathrm{C}}[g]} \qquad \text{s.t.} \quad \|g^\alpha\|_{\vartheta^\alpha} = 1\,, \qquad \forall \alpha \in \{1,\dots,d\}\,. \tag{5.18}$$

The gradient of cost $\hat{\mathrm{C}}[g]$ w.r.t. parameter vector $\boldsymbol{b}^\alpha$ of factor function g^α is

$$\tfrac{\partial \hat{\mathrm{C}}[g]}{\partial \boldsymbol{b}^\alpha} = \mathbf{C}^\alpha \boldsymbol{b}^\alpha - \tfrac{1}{d} \sum_{k=1}^{m'} \mathbf{C}^\alpha \mathbf{B}_k^\alpha \Big(\boldsymbol{a}^k \cdot \prod_{\beta \neq \alpha} (\mathbf{B}_k^\beta)^\top \mathbf{C}^\beta \boldsymbol{b}^\beta \Big) + \tfrac{1}{d} \mathbf{C}^\alpha \tilde{\mathbf{B}}^\alpha \Big(\tilde{\mathbf{A}}\mathbf{1} \cdot \prod_{\beta \neq \alpha} \tilde{\mathbf{B}}^{\beta\top} \mathbf{C}^\beta \boldsymbol{b}^\beta \Big)$$

and yields the unconstrained solution $\boldsymbol{b}_{\mathrm{uc}} \in \mathbb{R}^{m_\alpha}$, which differs only in the construction of the involved matrices from the LFF compression in Algorithm 6:

$$\boldsymbol{b}_{\mathrm{uc}} := \tfrac{1}{d} [\bar{\mathbf{B}}^\alpha, \tilde{\mathbf{B}}^\alpha] \Big(\big(\prod_{\beta \neq \alpha} \bar{\boldsymbol{c}}^\beta \big) \cdot \begin{bmatrix} \bar{\boldsymbol{a}} \\ -\tilde{\mathbf{A}}\mathbf{1} \end{bmatrix} \Big) \qquad \text{with} \qquad \bar{\boldsymbol{c}}^\alpha := [\bar{\mathbf{B}}^\alpha, \tilde{\mathbf{B}}^\alpha]^\top \mathbf{C}^\alpha \boldsymbol{b}^\alpha\,, \tag{5.19}$$

$$\text{and matrices} \quad \bar{\mathbf{B}}^\alpha := \left[\mathbf{B}^\alpha_1, \dots, \mathbf{B}^\alpha_{m'}\right] \quad \text{and} \quad \bar{\boldsymbol{a}}^\top := \left[\boldsymbol{a}_1^\top, \dots, \boldsymbol{a}_{m'}^\top\right]. \tag{5.20}$$

Algorithm 6 can therefore easily be adapted to compress multiple LFF f_i, each with its own set of factored bases, into a multivariate LFF $\tilde{\boldsymbol{f}}$ with shared bases.

5.2 Inductive factored learning

Linear factored functions (LFF) are very powerful tools for deductive value estimation (see Section 5.3 for more). However, deductive reasoning must always be grounded in knowledge from which new conclusions can be derived. This section investigates inductive learning with LFF. In this thesis, this serves three purposes:

1. The importance sampling in Equation 5.2 requires the PDF $\frac{d\zeta}{d\vartheta}$. In Section 5.2.1 I derive an algorithm that estimates the PDF from training samples.

2. An autonomous agent must be able to learn both the reward function and transition operator from experience. As shown in Equation 5.12, this is regression and is explored in Section 5.2.2 and more detailed in my publication Böhmer and Obermayer (2015, see Appendix B.2 on Page 216).

3. To estimate the (Q-)value function in Equation 5.1, one needs to empirically minimize the cost function $\|\text{td}\|^2_\zeta = \|f - r - \gamma \hat{\mathrm{P}}^\pi[f]\|^2_\zeta$. For this purpose I derive a LFF algorithm for (Q-)value estimation in Section 5.2.3 .

5.2.1 Density estimation with LFF

The importance sampling approach of Section 5.1.1 depends on the PDF of sampling distribution ζ. Instead of estimating the likelihood to draw a sample $\boldsymbol{z} \in \mathcal{Z}$, as for example in KERNEL DENSITY ESTIMATION (see e.g. Bishop, 2006), I propose here *regression* of the Radon-Nikodym derivative $\frac{d\zeta}{d\vartheta}$, which is the PDF for uniform ϑ. Following the methodology for compression in Section 5.1.3, I optimize one factored basis function $g(\boldsymbol{z}) := \prod_{\alpha=1}^d g^\alpha(z_\alpha) := \prod_{\alpha=1}^d \boldsymbol{b}^\alpha \boldsymbol{\phi}^\alpha(z_\alpha), \forall \boldsymbol{z} \in \mathcal{Z}$, at a time by assuming an estimate $f \in \mathcal{F}^m$ already exists. The least-squares costs are:

$$\inf_{g \in \mathcal{F}} \left\| \tfrac{d\zeta}{d\vartheta} - f - g \right\|^2_\vartheta \qquad \text{s.t.} \quad \left\| g^\alpha \right\|_{\vartheta^\alpha} = 1\,, \forall \alpha\,. \tag{5.21}$$

VIRTUAL SAMPLES

However, in difference to compression of LFF, one does not have access to $\frac{d\zeta}{d\vartheta}$ everywhere. To avoid over-fitting on the training set (see Section 3.2.2), one can regularize the cost function with VIRTUAL SAMPLES. Following the description in Section 3.2.4, I draw the virtual samples from the diagonal Gaussian conditional distribution $\chi = \prod_{\alpha=1}^{|\mathcal{Z}|} \chi^\alpha$, defined in Equation 3.18 on Page 31. In the limit $n \to \infty$ of an infinite training set, the cost with virtual samples are:

$$\iint \vartheta(d\boldsymbol{z})\, \chi(d\boldsymbol{y}|\boldsymbol{z}) \Big(\tfrac{d\zeta}{d\vartheta}(\boldsymbol{z}) - f(\boldsymbol{y}) - g(\boldsymbol{y}) \Big)^2 = \|g\|^2_{\vartheta\chi} + 2\langle g, f \rangle_{\vartheta\chi} - 2\langle g, 1 \rangle_\zeta + c\,. \tag{5.22}$$

Here $c \in \mathbb{R}$ denotes a constant that is independent of g and $\langle g, 1 \rangle_\zeta$ can be estimated by the empirical mean of g over the training samples. Using inner products $\langle f, g \rangle_{\vartheta\chi} = \iint \vartheta(d\boldsymbol{z})\, \chi(d\boldsymbol{y}|\boldsymbol{z})\, f(\boldsymbol{y})\, g(\boldsymbol{y})$ in $L^2(\mathcal{Z}, \vartheta\chi)$ allows a first order Taylor approximation of

the *regularized covariance matrices* $\mathbf{C}^\alpha_{\vartheta\chi} := \mathbf{C}^\alpha_\vartheta + \sigma^2_\alpha \dot{\mathbf{C}}^\alpha_\vartheta, \forall \alpha \in \{1,\dots,d\}$:

$$\begin{aligned}\langle \boldsymbol{\phi}^\alpha, \boldsymbol{\phi}^{\alpha\top}\rangle_{\vartheta\chi} &= \iint \vartheta^\alpha(dz_\alpha)\, \chi^\alpha(dy_\alpha|z_\alpha)\, \boldsymbol{\phi}^\alpha_{(y_\alpha)}\, \boldsymbol{\phi}^{\alpha\top}_{(y_\alpha)} \\ &\approx \underbrace{\langle \boldsymbol{\phi}^\alpha, \boldsymbol{\phi}^{\alpha\top}\rangle_{\vartheta^\alpha}}_{\mathbf{C}^\alpha_\vartheta} + \sigma^2_\alpha \underbrace{\langle \tfrac{\partial}{\partial z_\alpha}\boldsymbol{\phi}^\alpha, \tfrac{\partial}{\partial z_\alpha}\boldsymbol{\phi}^{\alpha\top}\rangle_{\vartheta^\alpha}}_{\dot{\mathbf{C}}^\alpha_\vartheta} =: \mathbf{C}^\alpha_{\vartheta\chi}\,. \end{aligned} \tag{5.23}$$

Note that for Fourier cosine functions ϕ^α_i (and many others) the covariance matrices can be determined analytically, that is, $C^\alpha_{ij} = \delta_{ij}$ and $\dot{C}^\alpha_{ij} = i^2\pi^2\delta_{ij}, \forall i,j \in \mathbb{N}$.

I also change the constrains to $\|g^\alpha\|_{\vartheta\chi} = \sqrt{\boldsymbol{b}^{\alpha\top}\mathbf{C}^\alpha_{\vartheta\chi}\boldsymbol{b}^\alpha} \stackrel{!}{=} 1$ for the sake of mathematical convenience. The resulting optimization problem is thus

$$\inf_{g\in\mathcal{F}} \underbrace{\|g\|^2_{\vartheta\chi} + 2\langle g, f\rangle_{\vartheta\chi} - 2\langle g, 1\rangle_\zeta}_{\hat{\mathrm{C}}[g]} \quad \text{s.t.} \quad \|g^\alpha\|_{\vartheta\chi} = 1\,, \;\; \forall\alpha\in\{1,\dots,d\}\,. \tag{5.24}$$

Algorithm 7 – density estimation with LFF

`input:` $\{\boldsymbol{z}_t\}_{t=1}^n \subset \mathbb{R}^d$, $\boldsymbol{\sigma} \in \mathbb{R}^d$ `// training samples` $\boldsymbol{z}_t \sim \zeta$

$\mathbf{B}^\alpha := \emptyset\,, \;\; \forall\alpha; \;\; \boldsymbol{a} := \emptyset\,; \;\; \det := \infty\,; \;\; \boldsymbol{\mu}_\psi := \emptyset$ `// initialize output` f

$\boldsymbol{\Phi}^\alpha := [\boldsymbol{\phi}^\alpha(\boldsymbol{z}_1),\dots,\boldsymbol{\phi}^\alpha(\boldsymbol{z}_n)]\,; \;\; \mathbf{C}^\alpha_{\vartheta\chi} := \langle \boldsymbol{\phi}^\alpha, \boldsymbol{\phi}^{\alpha\top}\rangle_{\vartheta^\alpha} + \sigma^2_\alpha\langle \nabla\boldsymbol{\phi}^\alpha, \nabla\boldsymbol{\phi}^{\alpha\top}\rangle_{\vartheta^\alpha}\,, \;\; \forall\alpha$

while $|\det| > \epsilon$ **do**

  $\boldsymbol{c}^\alpha := \mathbf{1} \in \mathbb{R}^{|\boldsymbol{a}|}\,; \;\; \boldsymbol{g}^\alpha := \mathbf{1} \in \mathbb{R}^n\,; \;\; d_\alpha := \infty\,, \;\; \forall\alpha$ `// initialize new` g

  while $\max_\alpha\{|d_\alpha|\} > \varepsilon$ **do**

    for $\alpha := \arg\max_\alpha\{|d_\alpha|\}$ **do**

      $d_\alpha := -2\Big(\frac{1}{n}\,\mathbf{1}^\top(\prod_\alpha \boldsymbol{g}^\alpha) - \boldsymbol{a}^\top(\prod_\beta \boldsymbol{c}^\beta)\Big)$ `//` $d_\alpha = \hat{\mathrm{C}}[g'] - 1$

      $\boldsymbol{b}_{\mathrm{uc}} := (\mathbf{C}^\alpha_{\vartheta\chi})^{-1}\Big(\frac{1}{n}(\boldsymbol{\Phi}^\alpha \cdot \mathbf{1}(\prod_{\beta\neq\alpha}\boldsymbol{g}^\alpha)^\top)\,\mathbf{1}\Big) - \mathbf{B}^\alpha\Big(\boldsymbol{a}\cdot(\prod_{\beta\neq\alpha}\boldsymbol{c}^\beta)\Big)$ `//` eq.5.25

      $\boldsymbol{b}^\alpha := \boldsymbol{b}_{\mathrm{uc}} \,/\, \sqrt{\boldsymbol{b}^\top_{\mathrm{uc}}\mathbf{C}^\alpha_{\vartheta\chi}\boldsymbol{b}_{\mathrm{uc}}}$ `// normalize` g^α `to length 1`

      $\boldsymbol{c}^\alpha := \mathbf{B}^{\alpha\top}\mathbf{C}^\alpha_{\vartheta\chi}\boldsymbol{b}^\alpha$ `// recompute coefficients`

      $\boldsymbol{g}^\alpha := \boldsymbol{\Phi}^{\alpha\top}\boldsymbol{b}^\alpha$ `// recompute` g^α`'s output`

      $d_\alpha := d_\alpha + 2\Big(\frac{1}{n}\,\mathbf{1}^\top(\prod_\alpha \boldsymbol{g}^\alpha) - \boldsymbol{a}^\top(\prod_\beta \boldsymbol{c}^\beta)\Big)$ `//` $d_\alpha = \hat{\mathrm{C}}[g'] - \hat{\mathrm{C}}[g]$

    end for `// end for loop: one` g^α `updated`

  end while `// end inner loop:` g `is new basis`

  $\mathbf{C}_\psi := \prod_\alpha\Big([\mathbf{B}^\alpha, \boldsymbol{b}^\alpha]^\top\mathbf{C}^\alpha_{\vartheta\chi}[\mathbf{B}^\alpha, \boldsymbol{b}^\alpha]\Big)\,; \;\; \det := \mathtt{det}(\mathbf{C}_\psi)\,; \;\; \boldsymbol{\mu}_\psi := \begin{bmatrix}\boldsymbol{\mu}_\psi \\ \frac{1}{n}\,\mathbf{1}^\top(\prod_\alpha \boldsymbol{g}^\alpha)\end{bmatrix}$

  if $|\det| > \epsilon$ **then**

    $\mathbf{B}^\alpha := [\mathbf{B}^\alpha, \boldsymbol{b}^\alpha]\,, \;\; \forall\alpha\,; \;\; \boldsymbol{a} := (\mathbf{C}_\psi)^{-1}\boldsymbol{\mu}_\psi$ `// if` g `lin. ind., update` f

  end if

end while `// end outer loop: no more linear independent` g

`output:` $\mathbf{B}^\alpha \in \mathbb{R}^{m_\alpha\times m}, \forall\alpha\,, \;\; \boldsymbol{a} \in \mathbb{R}^m$ `// PDF of` ζ`:` $f \approx \frac{d\zeta}{d\vartheta}$

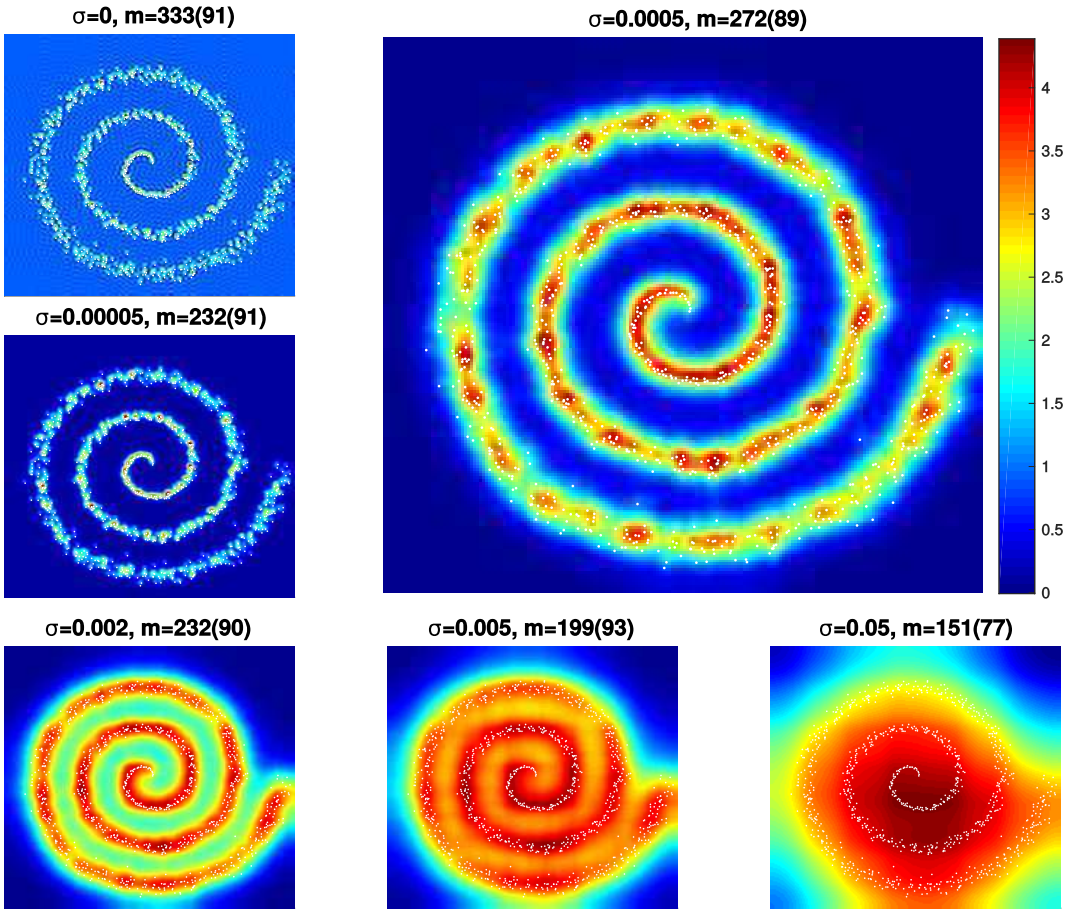


Figure 5.5: Density estimation (indicated by the background color) of samples drawn from a noisy spiral (white dots) with different regularization parameters σ. m denotes the number of basis functions before and (in brackets) after a compression.

Setting the derivative of the cost $\hat{\mathrm{C}}[g]$ w.r.t. parameter vector $\boldsymbol{b}^\alpha \in \mathbb{R}^{m_\alpha}$ of factor function g^α to zero yields the unconstrained solution $\boldsymbol{b}_{\mathrm{uc}} \in \mathbb{R}^{m_\alpha}$:

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{b}^\alpha}\hat{\mathrm{C}}[g] &\approx 2\,\mathbf{C}^\alpha_{\vartheta\chi}\boldsymbol{b}^\alpha + 2\,\mathbf{C}^\alpha_{\vartheta\chi}\mathbf{B}^\alpha\Big(\boldsymbol{a}\cdot\prod_{\beta\neq\alpha}\overbrace{\mathbf{B}^{\beta\top}\mathbf{C}^\beta_{\vartheta\chi}\boldsymbol{b}^\beta}^{\boldsymbol{c}^\beta\in\mathbb{R}^m}\Big) - 2\,\langle\boldsymbol{\phi}^\alpha\cdot\prod_{\beta\neq\alpha}g^\beta,1\rangle_\zeta \overset{!}{=} 0\,, \\ \Rightarrow \boldsymbol{b}_{\mathrm{uc}} &= (\mathbf{C}^\alpha_{\vartheta\chi})^{-1}\langle\boldsymbol{\phi}^\alpha\cdot\prod_{\beta\neq\alpha}g^\beta,1\rangle_\zeta - \mathbf{B}^\alpha\Big(\boldsymbol{a}\cdot\prod_{\beta\neq\alpha}\boldsymbol{c}^\beta\Big)\,. \end{aligned} \tag{5.25}$$

The constrained solution can be determined by normalization $\boldsymbol{b}^\alpha := \boldsymbol{b}_{\mathrm{uc}}/\sqrt{\boldsymbol{b}_{\mathrm{uc}}^\top\mathbf{C}^\alpha_{\vartheta\chi}\boldsymbol{b}_{\mathrm{uc}}}$.

CONVERGENCE CRITERIA

As in compression, the inner loop CONVERGES when the cost improvement of all dimensions drops below some threshold ε. The improvement for an update from g' to g in dimension α is computed exactly as in compression:

$$d_\alpha := \hat{\mathrm{C}}[g'] - \hat{\mathrm{C}}[g] = 2(\boldsymbol{b}^\alpha - \boldsymbol{b}'^\alpha)\,\mathbf{C}^\alpha_{\vartheta\chi}\,\boldsymbol{b}_{\mathrm{uc}}\,. \tag{5.26}$$

I observed empirically that d_α can take negative values, which I counteract by using $\max_\alpha\{|d_\alpha|\} \leq \varepsilon$ as convergence criterion of the inner loop. Convergence of the outer loop is less obvious, as ultimately the norm $\|\frac{d\zeta}{d\vartheta}\|_\vartheta$ can not be estimated empirically. Instead, I propose to evaluate whether the newly found basis function g is

linearly independent of the previously found $\{\psi_i\}_{i=1}^m$. Linearly dependent g do not contribute to the function f and density estimation can therefore stop here. The convergence criterion may check this, for example, by comparing the determinant of the covariance matrix $\mathbf{C}_\psi := \langle \boldsymbol{\psi}, \boldsymbol{\psi}^\top \rangle_{\vartheta\chi}$, that is, by stopping when $\mathtt{det}(\mathbf{C}_\psi) \leq \epsilon$.

The full procedure is shown in Algorithm 7 on Page 85. Note that in the inner loop I chose only to update the dimension α with the largest previous cost improvement d_α. This update scheme has worked empirically just as well as a for loop over all (permuted) dimensions, but at a fraction of the computational cost. Although I can not give any guarantees, I recommend it for cases with high dimensional data.

I tested the approach on 2-dimensional noisy spiral training data. Spirals can not be factorized and are therefore a challenge to Algorithm 7. To evaluate the behavior in areas of different density, the Gaussian noise increased from the inside to the outside. The $n = 1000$ training samples were generated the following:

$$\boldsymbol{z}_t \quad := \quad 3 \begin{bmatrix} \cos(6\pi\sqrt{t/n}) \\ \sin(6\pi\sqrt{t/n}) \end{bmatrix} \quad + \quad \mathcal{N}(\mathbf{0}, \tfrac{t}{8n}\mathbf{I}) \,, \qquad \forall t \in \{1, \ldots, n\} \,. \tag{5.27}$$

Figure 5.5 shows the estimated density $f \in \mathcal{F}^m$ for different regularization parameters $\sigma_\alpha := \sigma, \forall \alpha$. Note that σ determines the granularity of the estimate similar to the width-parameter of kernel density estimation. Although the algorithm is not forced to obey the non-negativity constraint of PDF, I observed only few locations with very small violations. The only exceptions are unregularized estimates ($\sigma = 0$, upper left plot), where the wave-like structure of the background often dips significantly below zero. However, I also observed that the resulting $f \in \mathcal{F}^m$ were not very compact. Compressing f after estimation often reduced the number of basis functions m by half or more. In Figure 5.5 the number of bases m is given both originally and in brackets after compression with virtually no loss ($\epsilon < 10^{-16}$).

5.2.2 Regression with LFF

Deductive control must be grounded in reality by transition and reward models. For LFF, both models can be learned by regression (see Section 5.3.3 for details on learning transition models). The least squares regression problem[9] for function $y \in L^2(\mathcal{Z}, \vartheta)$, as defined in Section 3.2.1, for the next factored basis $g \in \mathcal{F}$ is:

$$\inf_{g \in \mathcal{F}} \|y - f - g\|_\zeta^2 \qquad \text{s.t.} \quad \|g^\alpha\|_{\vartheta^\alpha} = 1, \quad \forall \alpha \in \{1, \ldots, d\} \,. \tag{5.28}$$

In difference to previous sections, for example, density estimation in Equation 5.24, the integral in the cost function is measured w.r.t. training distribution ζ, rather than uniform distribution ϑ. Note that ζ must not factorize and neither do inner products measured w.r.t. ζ. This denies regression some of the mathematical elegance seen in previous LFF algorithms. Furthermore, as ζ often samples $\mathcal{Z}$ very unevenly, the target function y's output in regions without training samples is undefined and must therefore be regularized. As before, I assume virtual samples distributed according

[9] Note that there is an alternative, simpler cost function for regression w.r.t. ϑ: $\inf_g \|y - f - g\|_\vartheta^2 = \|g\|_\vartheta^2 + 2\langle g, f \rangle_\vartheta - 2\langle g, \frac{d\vartheta}{d\zeta} \cdot y \rangle_\zeta + c$. Under the constraints $\|g^\alpha\|_{\vartheta^\alpha} = 1$, this can be solved very efficiently with the techniques used in density estimation. However, it requires to estimate the PDF $\frac{d\zeta}{d\vartheta}$ first, and may suffer in unforeseen ways from importance weighting with $\frac{d\vartheta}{d\zeta}(\boldsymbol{z}) = 1/\frac{d\zeta}{d\vartheta}(\boldsymbol{z})$. Due to time constraints, I leave the evaluation and comparison of this algorithm to future works.

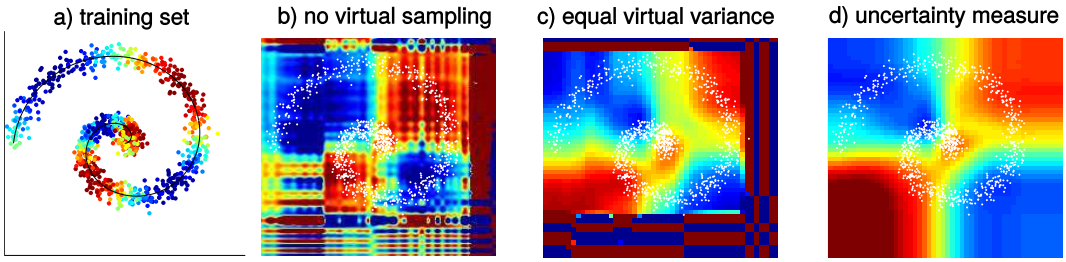


Figure 5.6: The effect of virtual sampling on LFF regression. a) The 2-dimensional training samples are drawn from a noisy spiral and labeled by a sin-wave as indicated by colors. b) Without virtual samples (or with very low variances, e.g. $\sigma = 10^{-16}$ used here), regression over-fits. c) Virtual samples with equal variance everywhere make smooth predictions within the box with available training data, but yield extreme outputs in unseen regions. d) Using the proposed uncertainty measure $\frac{d\vartheta}{d\zeta}$ to scale the virtual variance makes smooth predictions for all regions of $\mathcal{Z}$.

to Equation 3.18 on Page 31. The cost function $\hat{\mathrm{C}}[g]$ with virtual sampling is:

$$\iint \zeta(d\boldsymbol{z})\,\chi(d\boldsymbol{x}|\boldsymbol{z})\Big(f(\boldsymbol{x})+g(\boldsymbol{x})-y(\boldsymbol{z})\Big)^2 \quad = \quad \|g\|^2_{\zeta\chi}+2\langle g,f\rangle_{\zeta\chi}-2\langle g,y\rangle_{\zeta}+c\,, \tag{5.29}$$

where $c \in \mathbb{R}$ is a constant that is independent of g.

Virtual samples from Gaussian distributions centered around a training set drawn from ζ do not evenly regularize all regions of $\mathcal{Z}$, though. Regions with many training samples accumulate many virtual samples, too, whereas regions without training samples often receive (almost) none. In practice this can average too many "close-by" function outputs in densely sampled regions and fails to enforce reasonable predictions far away from the training data. To allow target functions with strongly differing outputs for "close-by" samples, I propose to *scale* the virtual variance with an UNCERTAINTY MEASURE. At sample $\boldsymbol{z} \in \mathcal{Z}$, the covariance matrix $\boldsymbol{\Sigma}$ of the virtual distribution is scaled with $\frac{d\vartheta}{d\zeta}(\boldsymbol{z})$ (see Figure 3.3 in Section 3.2.4), that is,

UNCERTAINTY MEASURE

$$\int \chi(d\boldsymbol{y}|\boldsymbol{z})\,(\boldsymbol{y}-\boldsymbol{z}) = \boldsymbol{0}\,, \quad \int \chi(d\boldsymbol{y}|\boldsymbol{z})\,(\boldsymbol{y}-\boldsymbol{z})(\boldsymbol{y}-\boldsymbol{z})^\top = \tfrac{d\vartheta}{d\zeta}(\boldsymbol{z})\cdot\boldsymbol{\Sigma}\,. \tag{5.30}$$

The effect of the uncertainty measure on LFF regression is shown in Figure 5.6. For arbitrary $f, g \in L^2(\mathcal{Z},\vartheta)$ and diagonal covariance matrices $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$ with diagonal elements $\{\sigma^2_\alpha\}^d_{\alpha=1}$, on can use the first Taylor approximation to compute

$$\begin{aligned}\langle f,g\rangle_{\zeta\chi} \;&\approx\; \int \zeta(d\boldsymbol{z})\Big(f(\boldsymbol{z})\,g(\boldsymbol{z}) + \boldsymbol{\nabla} f(\boldsymbol{z})^\top \overbrace{\textstyle\int \chi(\boldsymbol{y}|\boldsymbol{z})(\boldsymbol{y}-\boldsymbol{z})(\boldsymbol{y}-\boldsymbol{z})^\top}^{\frac{d\vartheta}{d\zeta}(\boldsymbol{z})\cdot\boldsymbol{\Sigma}} \boldsymbol{\nabla} g(\boldsymbol{z})\Big) \\ &=\; \langle f,g\rangle_\zeta + \sum_{\alpha=1}^d \sigma^2_\alpha \langle \nabla_\alpha f, \nabla_\alpha g\rangle_\vartheta\,. \end{aligned} \tag{5.31}$$

Note that while the inner product $\langle f,g\rangle_\zeta$ must be estimated empirically by an average over the training samples, the regularization terms factorize for factored functions $f,g \in \mathcal{F}$: $\langle \nabla_\alpha f, \nabla_\alpha g\rangle_\vartheta = \langle \nabla_\alpha f^\alpha, \nabla_\alpha g^\alpha\rangle_{\vartheta^\alpha} \prod_{\beta\neq\alpha} \langle f^\beta, g^\beta\rangle_{\vartheta^\beta}$. The regularized optimization problem of Equation 5.28 is therefore approximatively:

$$\inf_{g\in\mathcal{F}} \underbrace{\|y-f-g\|^2_\zeta \;+\; \sum_{\alpha=1}^d \sigma^2_\alpha \|\nabla_\alpha g + \nabla_\alpha f\|^2_\vartheta}_{\hat{\mathrm{C}}[g]} \qquad \text{s.t.} \quad \|g^\alpha\|_{\vartheta^\alpha} = 1\,, \quad \forall\alpha\,. \tag{5.32}$$

The derivation of the regression algorithm in Böhmer and Obermayer (2015) on Page 229 contains a small, inconsequential error and I repeat the correct derivation in the following for the sake of completeness. The corrected algorithm extracts more basis functions, but prediction quality did not appear to improve significantly.

The derivatives of all summands in Equation 5.32 w.r.t. to parameter vector $\boldsymbol{b}^\alpha$ of factor function g^α of the updated dimension α are:

$$\frac{\partial}{\partial \boldsymbol{b}^\alpha} \|y - f - g\|_\zeta^2 \;=\; 2\langle \boldsymbol{\phi}^\alpha \cdot \prod_{\beta\neq\alpha} g^\beta,\, \prod_{\beta\neq\alpha} g^\beta \cdot \boldsymbol{\phi}^\alpha\rangle_\zeta \boldsymbol{b}^\alpha - 2\langle \boldsymbol{\phi}^\alpha \cdot \prod_{\beta\neq\alpha} g^\beta, y - f\rangle_\zeta \,, \tag{5.33}$$

$$\frac{\partial}{\partial \boldsymbol{b}^\alpha} \|\nabla_\gamma g + \nabla_\gamma f\|_\vartheta^2 \;=\; 2 \begin{cases} \dot{\mathbf{C}}^\alpha \boldsymbol{b}^\alpha + \dot{\mathbf{C}}^\alpha \mathbf{B}^\alpha \Big[\boldsymbol{a} \cdot \prod_{\beta\neq\alpha} \overbrace{\mathbf{B}^{\beta\top} \mathbf{C}^\beta \boldsymbol{b}^\beta}^{\boldsymbol{c}^\beta \in \mathbb{R}^m} \Big] & , \text{if } \alpha = \gamma \\ \mathbf{C}^\alpha \boldsymbol{b}^\alpha \underbrace{\boldsymbol{b}^{\gamma\top} \dot{\mathbf{C}}^\gamma \boldsymbol{b}^\gamma}_{\|\nabla g^\gamma\|^2_{\vartheta^\gamma} \in \mathbb{R}} + \mathbf{C}^\alpha \mathbf{B}^\alpha \Big[\boldsymbol{a} \cdot \underbrace{\mathbf{B}^{\gamma\top} \dot{\mathbf{C}}^\gamma \boldsymbol{b}^\gamma}_{\dot{\boldsymbol{c}}^\gamma \in \mathbb{R}^m} \cdot \prod_{\beta\neq\alpha\neq\gamma} \boldsymbol{c}^\beta \Big] & , \text{if } \alpha \neq \gamma \end{cases} \tag{5.34}$$

Algorithm 8 – regression with LFF

```
Input:  X ∈ ℝ^{d×n},  y ∈ ℝ^n,  σ² ∈ ℝ^d  ε, ε ∈ ℝ
C^α := ⟨φ^α, φ^α⟩_{ϑ^α},  Ċ^α := ⟨∇φ^α, ∇φ^α⟩_{ϑ^α},  ∀α        // analytical covariance matrices
Φ^α_{jt} := φ^α_j(X_{αt}),  ∀α, ∀j, ∀t                          // optional cache of sample-expansion
f := 0 ∈ ℝ^n;  a := ∅;  B^α := ∅, ∀α;  Ψ := ∞;  m := 0          // initialize empty f ∈ F^0
while det((1/n) Ψ Ψ^⊤) > ε  do
  b^α := 1^α ∈ ℝ^{m_α}, ∀α;  g^α := 1 ∈ ℝ^n, ∀α                 // initialize all g^α as constant
  c^α := 1 ∈ ℝ^m, ∀α;  ċ^α := 1 ∈ ℝ^m, ∀α                       // initialize coefficients
  h := ∞ ∈ ℝ^d;  d := 0 ∈ ℝ^d                                   // initialize estimated improvement
  while max_α{|h_α|} > ε  do
    for α ∈ arg min_α{|h_α|}  do
      b' := b^α                                                 // remember old parameter vector
      C̄ := Φ^α [ Φ^{α⊤} · ∏_{β≠α} (g^β)² 1^⊤ ] + σ²_α Ċ^α + Σ_{β≠α} σ²_β d_β C^α   // (eq. 5.35)
      ū := Φ^α [ (y − f) · ∏_{β≠α} g^β ]                        // ū ≈ ⟨φ^α · ∏_{β≠α} g^β, y − f⟩_ζ
      ū := ū + σ²_α Ċ^α B^α ( a · ∏_{β≠α} c^β ) + C^α B^α ( a · Σ_{γ≠α} ( σ²_γ ċ^γ · ∏_{β≠α≠γ} c^β ) )   // (eq. 5.36)
      b_uc := C̄^{-1} ū;  b^α := b_uc / √(b_uc^⊤ C^α b_uc)       // update factor function g^α
      h_α := 2(b^α − b')^⊤ C̄ b_uc + b'^⊤ C̄ b' − b^{α⊤} C̄ b^α    // calculate cost improvement h_α
      c^α := B^{α⊤} C^α b^α;  ċ^α := B^{α⊤} Ċ^α b^α             // update coefficients
      g^α := Φ^{α⊤} b^α;  d_α := b^{α⊤} Ċ^α b^α                 // update g^α's predictions
    end for                                                     // end for-loop: function g^α update
  end while                                                     // end inner loop: cost function converged and thus g optimized
  B^α := [B^α, b^α], ∀α;  Ψ := [ ∏_{α=1}^d B^{α⊤} Φ^α ];  m++   // adding g to the bases functions of f
  a := (Ψ Ψ^⊤)^{-1} Ψ y;  f := Ψ^⊤ a                             // project y onto new bases
end while                                                       // end outer loop: new g no longer linear independent, thus f ≈ y
Output:  a ∈ ℝ^m,  {B^α ∈ ℝ^{m_α × m}}_{α=1}^d                  // return parameters of f ∈ F^m
```

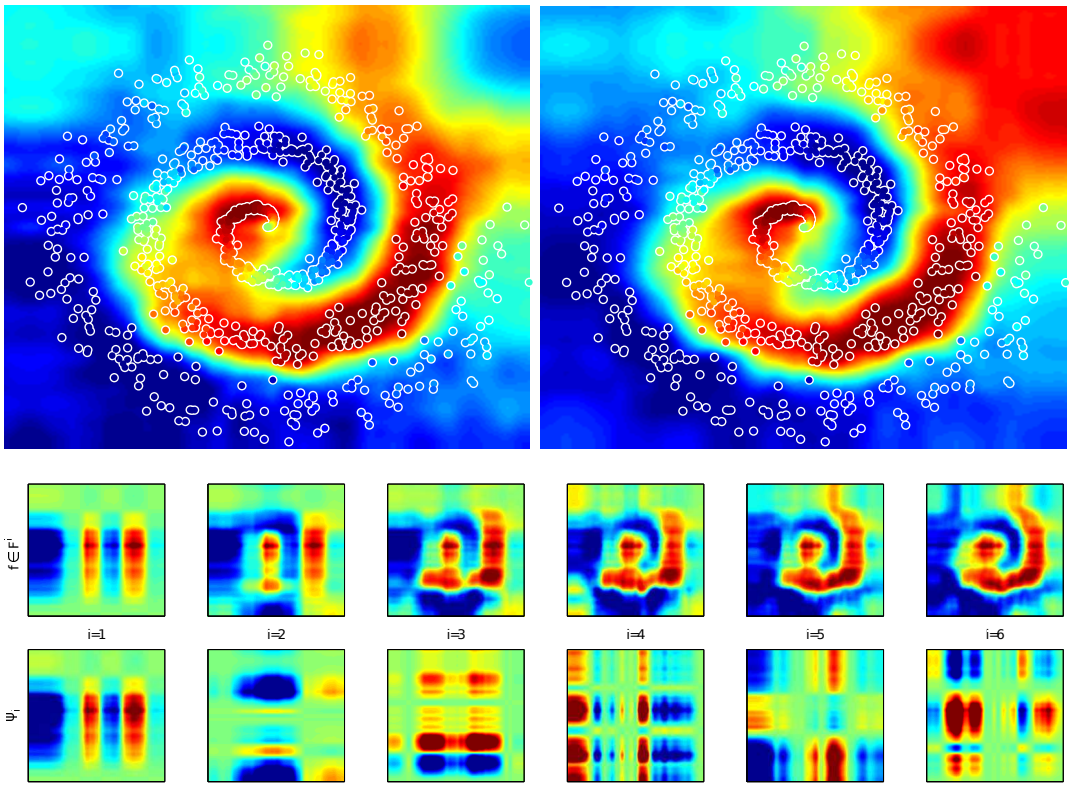


Figure 5.7: Two functions learned by LFF regression (background color in top plots, reproduced from Böhmer and Obermayer, 2015). The training data is the same noisy spiral (white circles), labeled by a sinus (color within the circles). Below are the first 6 factored basis functions of the left function and OLS predictions using the first i bases. Note that the function can already be recognized using 5–6 bases.

Setting these derivatives to zero yields the unconstrained solution $\boldsymbol{b}_{\text{uc}} := (\bar{\mathbf{C}}^\alpha)^{-1}\bar{\boldsymbol{u}}^\alpha$:

$$\bar{\mathbf{C}}^\alpha := \frac{1}{n}\sum_{t=1}^{n} \boldsymbol{\phi}^\alpha_{(z_{\alpha t})}\Big(\prod_{\beta\neq\alpha} g^\beta_{(z_{\beta t})}\Big)^2 \boldsymbol{\phi}^{\alpha\top}_{(z_{\alpha t})} + \sigma^2_\alpha \dot{\mathbf{C}}^\alpha + \Big(\sum_{\beta\neq\alpha}\sigma^2_\beta \|\nabla g^\beta\|^2_{\vartheta^\beta}\Big)\,\mathbf{C}^\alpha \quad (5.35)$$

$$\bar{\boldsymbol{u}}^\alpha := \frac{1}{n}\sum_{t=1}^{n} \boldsymbol{\phi}^\alpha_{(z_{\alpha t})}\big(y_{(z_{\alpha t})} - f_{(z_{\alpha t})}\big)\prod_{\beta\neq\alpha} g^\beta_{(z_{\beta t})} + \sigma^2_\alpha \dot{\mathbf{C}}^\alpha \mathbf{B}^\alpha\big[\boldsymbol{a}\cdot\prod_{\beta\neq\alpha}\boldsymbol{c}^\beta\big] \quad (5.36)$$
$$+ \mathbf{C}^\alpha \mathbf{B}^\alpha\Big[\boldsymbol{a}\cdot\sum_{\gamma\neq\alpha}\big(\sigma^2_\gamma \dot{\boldsymbol{c}}^\gamma \cdot \prod_{\beta\neq\alpha\neq\gamma}\boldsymbol{c}^\beta\big)\Big]\,.$$

The constraint solution is therefore $\boldsymbol{b}^\alpha \leftarrow \boldsymbol{b}_{\text{uc}}/\sqrt{\boldsymbol{b}_{\text{uc}}^\top \mathbf{C}_\vartheta \boldsymbol{b}_{\text{uc}}}$ as in Section 5.1.3. The cost improvement from g' to g by updating dimension α is also very similar, but regularized norms $\boldsymbol{b}^{\alpha\top}\bar{\mathbf{C}}\,\boldsymbol{b}^\alpha$ of a factored function $g \in \mathcal{F}$ are not constant here:

$$\hat{\mathrm{C}}[g'] - \hat{\mathrm{C}}[g] \quad = \quad 2(\boldsymbol{b}^\alpha - \boldsymbol{b}'^\alpha)^\top \bar{\mathbf{C}}^\alpha \boldsymbol{b}_{\text{uc}} + \boldsymbol{b}'^{\alpha\top}\bar{\mathbf{C}}^\alpha \boldsymbol{b}' - \boldsymbol{b}^{\alpha\top}\bar{\mathbf{C}}^\alpha \boldsymbol{b}\,.$$

The inner loop converges when the this improvement is dropping below ϵ for each updated dimension α, the outer loop when the converged basis function g is linearly dependent on previous bases. Algorithm 8 on Page 89 depicts all steps in detail.

In Böhmer and Obermayer (2015), I test the algorithm using spiral toy-data and established benchmark tests. Figure 5.7 shows two LFF learned from the same artificial training data drawn from a 2-dimensional noisy spiral and labeled by a sinus.

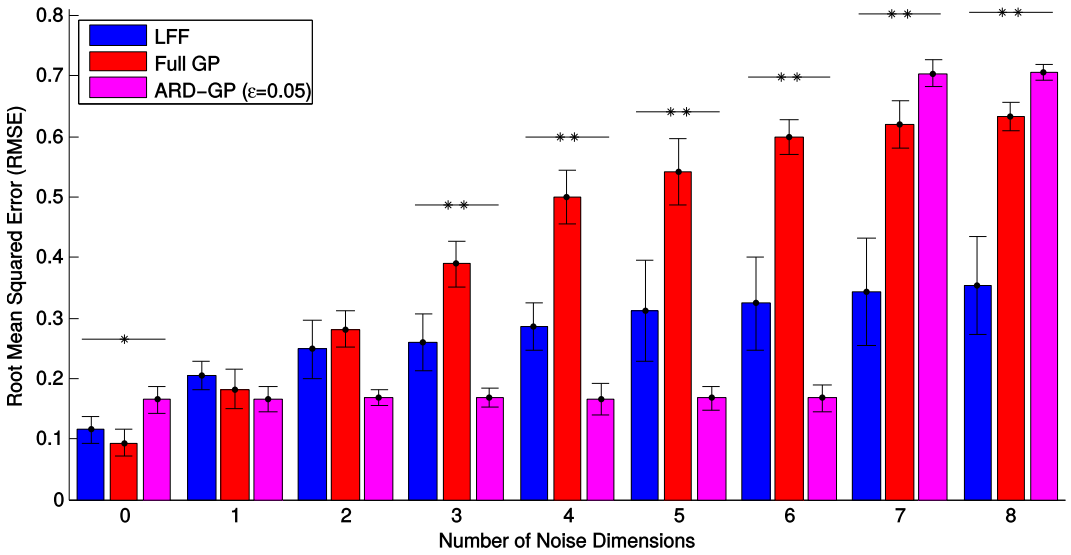


Figure 5.8: Comparison of LFF regression with Gaussian processes (GP) on the spiral toy-example from Figure 5.7 with additional dimensions, containing Gaussian i.i.d. noise (adapted and extended from Böhmer and Obermayer, 2015). *Full GP* denotes the a GP with Gaussian kernel, and *ARD-GP* a GP with Gaussian *automatic relevance detection* (ARD) kernel, that is, a diagonal covariance matrix that is optimized during training. Single stars denote significant ($p < 0.05$), double stars highly significant ($p < 0.005$) differences between *LFF* and *Full GP* RMSE.

The task was chosen because spirals can not be factorized and present therefore a challenging task for LFF algorithms. Algorithm 8 (in difference to the algorithm in Böhmer and Obermayer, 2015) will only predict one function for a given training set, due to a heuristic choice of the updated dimension α. However, Figure 5.7 demonstrates that other heuristics will most likely yield *similar* functions, by choosing the updated g^α randomly through permutation (see also Section 5.1.3). LFF regression makes here different predictions each time the algorithm is run. Note that the two functions on the top differ only in regions without training samples, though, which implies similar predictions near the training samples regardless of the update sequence. The learned factored basis functions plotted below also demonstrate that a recognizable function can be constructed with very few bases.

To evaluate the performance of Algorithm 8 *quantitatively*, I compared the ROOT MEAN SQUARED ERROR (RMSE) on an unseen test set drawn from the same distribution against Gaussian processes (GP, see Rasmussen and Williams, 2006; Bishop, 2006). These are a state-of-the-art kernel technique for regression. I used both standard GP with a Gaussian kernel (with predetermined uniform variance) and AUTOMATIC RELEVANCE DETECTION (ARD), which adapt the diagonal covariance matrix of a Gaussian kernel during training. The latter converges when the gradient of the kernel covariance shrinks below some threshold ε. Figure 5.8 shows this comparison at the artificial 2-dimensional example from Figure 5.7 with additional "noise" dimensions which were determined randomly, independent of the spiral. One can see that LFF regression handles these distracting dimensions much better than standard GP. ARD-GP, on the other hand, can ignore up to 6 noise dimensions by adjusting its kernel covariance matrix. In principle, LFF appear ideally suited for

ARD-KERNEL

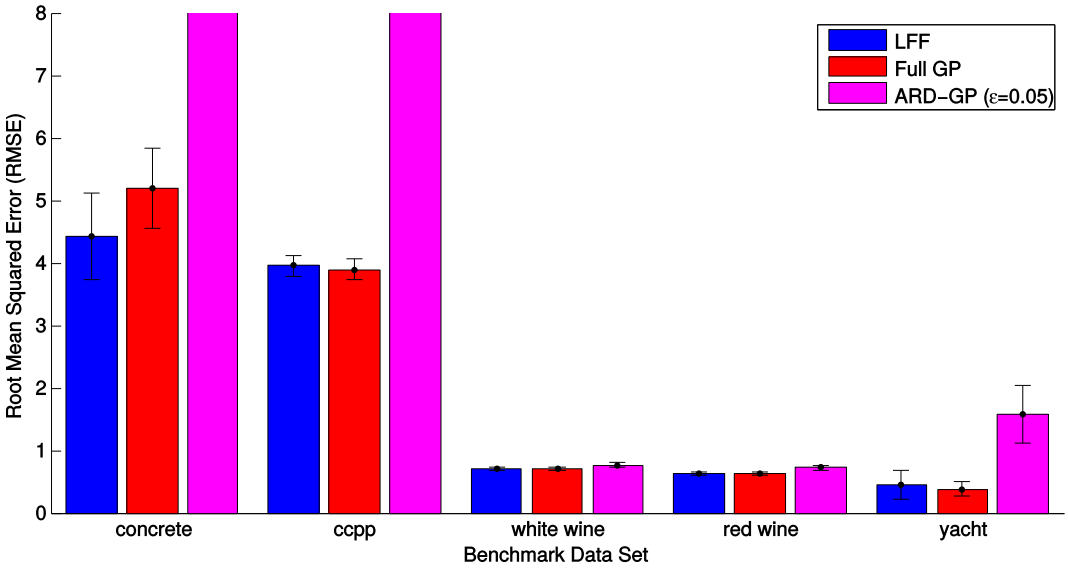


Figure 5.9: Comparison as in Figure 5.8, but on regression benchmark data sets (adapted and extended from Böhmer and Obermayer, 2015, see also Page 216pp). ARD-GP did not converge on the *concrete* and *ccpp* data sets, and RMSE were large.

sparse solutions, that is, for functions that depend only on a subset of the input dimensions. Without explicitly enforcing sparse solutions, however, LFF regressions will not learn them. I discuss this topic further in Section 5.5.1, and give a potential formulation as well as a discussion of its practical problems.

Beyond this artificial case, Figure 5.9 demonstrates that LFF regression RMSE performance is on a par with standard Gaussian processes. When compared on five standard regression tasks (see Böhmer and Obermayer, 2015, in Appendix B.2 for details), the only statistical significant difference (*concrete*) did not hold up after considering multiple testing. ARD-GP can do similarly well (*white* and *red wine*), but did not always converge properly (*concrete* and *ccpp*). However, I used the GPML toolbox[10] to perform ARD-GP regression, and can not rule out that this failure may be correctable with more fine-tuning.

5.2.3 Inductive value estimation with LFF

To combine inductive and deductive learning with LFF, as introduced in Section 5.1.1, one requires one last inductive learning method: (Q)-value estimation. Here one wishes to minimize the TD-error w.r.t. the training state(-action) distribution ζ. Straight forward *Bellman residual minimization* (BRM, Schweitzer and Seidmann, 1985) requires a bias free estimation of the term $\|\hat{\mathrm{P}}[\hat{\Gamma}_\pi[g]]\|^2_\zeta$, which is impossible without *double sampling*, as explained in Section 3.3.2. Therefore, I use a bound similar to *least-squares temporal difference learning* (LSTD, Bradtke and Barto, 1996, see Section 3.3.2). In the following, $\hat{\mathrm{P}}^\pi$ denotes either the transition operator $\hat{\Gamma}_\pi[\hat{\mathrm{P}}]$ (for $\zeta := \xi$) or $\hat{\mathrm{P}}[\hat{\Gamma}_\pi]$ (for $\zeta := \xi\pi$), and I define the TD-error of current estimate $f \in \mathcal{F}^m$ as $\delta := r + \gamma\hat{\mathrm{P}}^\pi[f] - f \in L^2(\mathcal{Z}, \zeta)$. The BRM cost function can be bounded by:

[10] By Carl Edward Rasmussen and Hannes Nickisch, (`http://gaussianprocess.org/gpml/code`).

$$\begin{aligned}\|\mathrm{td}\|_\zeta^2 &= \|(g+f)-(r+\gamma\hat{\mathrm{P}}^\pi[g+f])\|_\zeta^2 = \|g-\gamma\hat{\mathrm{P}}^\pi[g]\|_\zeta^2 - 2\langle g-\gamma\hat{\mathrm{P}}^\pi[g],\delta\rangle_\zeta + \|\delta\|_\zeta^2 \\ &= \|g\|_\zeta^2 - 2\gamma\langle g,\hat{\mathrm{P}}^\pi[g]\rangle_\zeta + \gamma^2\|\hat{\mathrm{P}}^\pi[g]\|_\zeta^2 - 2\langle g,\mathrm{td}\rangle_\zeta + 2\gamma\langle\hat{\mathrm{P}}^\pi[g],\mathrm{td}\rangle_\zeta + \|\mathrm{td}\|_\zeta^2 \\ &< 2\|g\|_\zeta^2 - 2\gamma\langle g,\hat{\mathrm{P}}^\pi[g]\rangle_\zeta - 2\langle g-\gamma\hat{\mathrm{P}}^\pi[g],\delta\rangle_\zeta + \|\delta\|_\zeta^2 \quad =: \quad \hat{\mathrm{C}}[g]\,.\end{aligned}$$

The inequality holds due to $\gamma^2 < 1$ and $\|\hat{\mathrm{P}}^\pi[g]\|_\zeta^2 \leq \|g\|_\zeta^2$, which follows directly from Lemma C.2 on Page 233. Note that this bound only holds when ζ is the steady-state distribution of P^π, that is, when training samples are drawn from a Markov chain executing policy π. As in LSTD, however, I assume that other distributions, and in particular other policies, will produce reasonable estimates as well.

Deriving the gradient of $\hat{\mathrm{C}}[g]$ w.r.t. parameter vector $\boldsymbol{b}^\alpha \in \mathbb{R}^{m_\alpha}$ of optimized factor function g^α and setting it to zero yields the unconstrained solution $\boldsymbol{b}_{\mathrm{uc}}$:

$$\begin{aligned}\frac{\partial\hat{\mathrm{C}}[g]}{\partial\boldsymbol{b}^\alpha} &= 4\overbrace{\langle\boldsymbol{\phi}^\alpha\cdot\prod_{\beta\neq\alpha}g^\beta,\prod_{\beta\neq\alpha}g^\beta\cdot\boldsymbol{\phi}^{\alpha\top}\rangle_\zeta}^{\mathbf{C}_\alpha}\boldsymbol{b}^\alpha - 2\gamma\overbrace{\langle\boldsymbol{\phi}^\alpha\cdot\prod_{\beta\neq\alpha}g^\beta,\hat{\mathrm{P}}^\pi\Big[\prod_{\beta\neq\alpha}g^\beta\cdot\boldsymbol{\phi}^{\alpha\top}\Big]\rangle_\zeta}^{\mathbf{D}_\alpha}\boldsymbol{b}^\alpha \\ &\quad - 2\gamma\,\mathbf{D}_\alpha^\top\boldsymbol{b}^\alpha - \underbrace{2\,\langle\boldsymbol{\phi}^\alpha\cdot\prod_{\beta\neq\alpha}g^\beta - \gamma\hat{\mathrm{P}}^\pi\Big[\boldsymbol{\phi}^\alpha\cdot\prod_{\beta\neq\alpha}g^\beta\Big],\mathrm{td}\rangle_\zeta}_{\boldsymbol{d}_\alpha}\,, \\ \Rightarrow\boldsymbol{b}_{\mathrm{uc}} &= (2\mathbf{C}_\alpha-\gamma(\mathbf{D}_\alpha+\mathbf{D}_\alpha^\top))^{-1}\boldsymbol{d}_\alpha\,. \end{aligned} \tag{5.37}$$

These matrices can be computed by averages[11] over the training set:

$$\begin{aligned}\mathbf{C}_\alpha &\approx \tfrac{1}{n}\sum_{t=1}^n\boldsymbol{\phi}^\alpha_{(z_t)}\Big(\prod_{\beta\neq\alpha}g^\beta_{(z_t)}\Big)^2\boldsymbol{\phi}^{\alpha\top}_{(z_t)}\,, \qquad \mathbf{D}_\alpha \approx \tfrac{1}{n}\sum_{t=1}^n\boldsymbol{\phi}^\alpha_{(z_t)}\Big(\prod_{\beta\neq\alpha}g^\beta_{(z_t)}\,g^\beta_{(z_{t+1})}\Big)\boldsymbol{\phi}^{\alpha\top}_{(z_{t+1})}\,, \\ \boldsymbol{d}_\alpha &\approx \tfrac{1}{n}\sum_{t=1}^n\Big(\boldsymbol{\phi}^\alpha_{(z_t)}\prod_{\beta\neq\alpha}g^\beta_{(z_t)} - \gamma\boldsymbol{\phi}^\alpha_{(z_{t+1})}\prod_{\beta\neq\alpha}g^\beta_{(z_{t+1})}\Big)\Big(r_t+\gamma f(\boldsymbol{z}_{t+1})-f(\boldsymbol{z}_t)\Big)\,. \end{aligned} \tag{5.38}$$

As in the case of regression, the cost function should be regularized by virtual samples to avoid over-fitting. However, in concert with a deductive (Q)-value estimation, the deductive part should provide sufficient regularization of both f and g. I therefore omitted the derivation for virtual samples here, although a stand-alone inductive (Q)-value estimation for LFF would require it.

Due to time constraints, I was not able to implement and test the algorithm. However, given my experience with the general LFF framework, demonstrated in this chapter, I am very confident that temporal difference learning will behave similar to regression. An exhaustive evaluation is left to future work, though.

[11] In the case of Q-value estimation, the future state-action pair $z_{t+1} \in \mathcal{Z}$ depends on the *evaluated*, rather than the sampling policy. Applying this to a function may require an integral over that function, for example, $\boldsymbol{\phi}^\beta_{(z_{t+1})} := \int\pi(d\boldsymbol{a}|\boldsymbol{x}_{t+1})\,\boldsymbol{\phi}^\beta_{(x_{t+1},a)}$. Using greedy policy improvement as in LSPI, on the other hand, only selects one action and does thus not require a costly integration. However, I have shown in Section 3.3.3 that a greedy approach can become instable for larger γ.

5.3 Deductive factored learning

Inductive learning of control is fundamentally restricted to situations that have been experienced during training. Representation and regularization can generalize values and policies to state-actions "similar" to the training set, but this requires extensive domain knowledge when designed by an expert (see Chapter 3) or must itself be learned inductively (with the same drawbacks, see Chapter 4).

In this chapter I want to employ deductive planning to generalize values (and thus policies) to previously unseen situations. It is important to notice that this aims beyond the usual distinction between model-based and model-free methods:

- Model-free (Q-)value estimation algorithms (like LSTD, Bradtke and Barto, 1996, see Section 3.3.2) optimize the temporal-difference error w.r.t. the state (-action) distribution ζ of the training data, that is, $\inf_f \|r + \gamma \hat{\mathrm{P}}^\pi[f] - f\|^2_\zeta$.

- Model-based algorithms (like Dyna_Q, Sutton, 1990, see Section 3.3.1), on the other hand, can generate artificial samples from uniform distribution ϑ, that is, optimize $\inf_f \|r' + \gamma \hat{\mathrm{P}}'^\pi[f] - f\|^2_\vartheta$. Transitions and reward is predicted by an inductively learned reward function $r' \in L^2(\mathcal{Z}, \vartheta)$ and policy-dependent transition operator $\hat{\mathrm{P}}'^\pi : L^2(\mathcal{Z}, \vartheta) \to L^2(\mathcal{Z}, \vartheta)$.

The latter *appears* to generalize to unseen states, but is based on predictions that are only valid around the training samples, too. Intuitively one can imagine that samples with a small PREDICTION CERTAINTY *propagate* values incorrectly and lead to bad value estimates everywhere. In principle the solution could be stabilized by weighting the sampled transitions with the associated certainty. However, this stability strongly reduces the intended *generalization* of the model-based approach. For example, using the training data PDF $\frac{d\zeta}{d\vartheta}$ as a measure of certainty (see Figure 3.3 on Page 32), the weighted model-based approach simply optimizes the inductive, model-free cost function. In a more practical example, it has been proven that for LSTD model-free and model-based approaches yield exactly the same solution (see Parr et al., 2008, and Section 3.3.2). Generalization of values to *unseen* states must therefore generalize the transition and reward models beyond the inductive limitations of classical machine learning. This requires STRUCTURAL ASSUMPTIONS on the transition model.

In this section I demonstrate how to use structural assumptions, in particular a dynamic Bayesian network (DBN), for deductive planning with LFF. I begin with a deductive extension of LSPI in Section 5.3.1. Next I introduce DBN for reinforcement learning in Section 5.3.2 and show how to learn them inductively with LFF in Section 5.3.3. Section 5.3.4 discusses how to exploit the learned DBN for efficient deductive (Q-)value estimation in large state-action spaces. Note that I do not present a policy improvement step that explicitely exploits the DBN. In Section 5.5.4 I explain why this may constitute the largest obstacle to efficient deductive learning, and discuss potential directions of future research.

The approach presented in this section has been the primary motivation to investigate LFF in the first place. However, developing the inductive framework for LFF presented in Sections 5.1 and 5.2 took longer than anticipated. Many approaches derived in this section (except for Section 5.3.1) have therefore, although mathematically sound, not been verified empirically yet.

5.3.1 Factored approximate planning

Before I make structural assumptions, I want to show how the methodology of linear value estimators (like LSTD) and policy improvement (like LSPI) can be applied deductively, that is, with no direct interaction between agent and environment. The work presented in this section illustrates my starting point to factored reinforcement learning and was published in an earlier form as the FACTORED APPROXIMATE PLANNING algorithm (FAPI, Böhmer and Obermayer, 2013). FAPI

To clarify the nomenclature, I assume a d-dimensional state space $\mathcal{X}$ with uniform measure ϑ^x, and a b-dimensional action space $\mathcal{A}$ with uniform measure ϑ^a, that is, $\vartheta = \vartheta^x \cdot \vartheta^a$. All state-action variables can be discrete, but in the following I will assume them to be continuous[12] and each variable to be encoded by Fourier cosine bases $\{\phi_j^\alpha\}_{j=1}^{m_\alpha}$. To represent values and models, I assume in this section to have access to $\bar{m}$ basis functions $\bar{\psi}_i$ over state space $\mathcal{X}$ and m basis functions ψ_j over state-action space $\mathcal{X} \times \mathcal{A}$:

$$\{\bar{\psi}_i\}_{i=1}^{\bar{m}} \subset L^2(\mathcal{X}, \vartheta^x) \qquad \text{and} \qquad \{\psi_j\}_{j=1}^{m} \subset L^2(\mathcal{X} \times \mathcal{A}, \vartheta)\,. \tag{5.39}$$

These bases are also assumed to be factored functions and $\{\psi_j\}_{j=1}^m$ should be well suited to approximate both the reward function $r \in L^2(\mathcal{X}\times\mathcal{A}, \vartheta)$ and all transitioned state-bases $\hat{\mathrm{P}}[\bar{\psi}_i] \in L^2(\mathcal{X} \times \mathcal{A}, \vartheta)$:

$$\mathrm{span}(\{\psi_j\}_{j=1}^m) \quad \approx \quad \mathrm{span}(\{r\} \cup \{\hat{\mathrm{P}}[\bar{\psi}_i]\}_{i=1}^{\bar{m}})\,. \tag{5.40}$$

Moreover, I assume to know linear models for the reward and transition operator:

$$r(\boldsymbol{z}) \overset{!}{\approx} \boldsymbol{r}^\top \boldsymbol{\psi}(\boldsymbol{z})\,, \qquad \hat{\mathrm{P}}[\bar{\boldsymbol{\psi}}](\boldsymbol{z}) \overset{!}{\approx} \mathbf{P}^\top \boldsymbol{\psi}(\boldsymbol{z})\,, \qquad \forall \boldsymbol{z} \in \mathcal{X} \times \mathcal{A}\,. \tag{5.41}$$

In Section 5.3.3 I describe in detail how transition model $\mathbf{P}$ and bases ψ_j can be constructed efficiently, given a *dynamic Bayesian network* (DBN). In Böhmer and Obermayer (2013), I introduce additional assumptions to estimate the Q-value function for a given policy. Later I discovered that it is sufficient to estimate the value function instead. This requires only the above assumptions and is also faster, as less bases are needed to represent the value. Both algorithms behave similarly, and I will restrict myself here to FAPI with value estimation. Interested readers are referred to Appendix B.1 for the more cumbersome Q-value estimation algorithm.

As in the derivation of LSTD in Section 3.3.2, I start with estimating a function $f(\boldsymbol{x}) := \boldsymbol{a}^\top \bar{\boldsymbol{\psi}}(\boldsymbol{x}), \forall \boldsymbol{x} \in \mathcal{X}$, that approximates *one* application of the Bellman operator onto the given (previously estimated) value function $v(\boldsymbol{x}) := \boldsymbol{a}'^\top \bar{\boldsymbol{\psi}}(\boldsymbol{x}) \in \mathcal{F}^{\bar{m}}$: VALUE ESTIMATION

$$\inf_f \left\| \hat{\Gamma}_\pi[r] + \gamma \hat{\Gamma}_\pi[\hat{\mathrm{P}}[v]] - f \right\|_{\vartheta^x}^2 \quad \Rightarrow \quad \boldsymbol{a} = \underbrace{\langle \bar{\boldsymbol{\psi}}, \bar{\boldsymbol{\psi}}^\top \rangle_{\vartheta^x}^{-1}}_{\bar{\mathbf{C}}^{-1}} \underbrace{\langle \bar{\boldsymbol{\psi}}, \hat{\Gamma}_\pi[\boldsymbol{\psi}^\top] \rangle_{\vartheta^x}}_{\bar{\mathbf{D}}^\pi} (\boldsymbol{r} + \gamma \mathbf{P} \boldsymbol{a}')\,. \tag{5.42}$$

Performing this repeatedly should[13] converge to the fix-point $f^*(\boldsymbol{x}) := \boldsymbol{a}^{*\top} \bar{\boldsymbol{\psi}}(\boldsymbol{x})$ of the projected Bellman operator $\hat{\Pi}^{\bar{\psi}}_{\vartheta^x}[\hat{\Gamma}_\pi[r + \gamma \hat{\mathrm{P}}[\cdot]]]$, which can be found by setting $\boldsymbol{a} = \boldsymbol{a}'$. The fix point and the corresponding Q-value function $q^\pi \in \mathcal{F}^m$ are

$$\boldsymbol{a}^* \;=\; (\bar{\mathbf{C}} - \gamma \bar{\mathbf{D}}^\pi \mathbf{P})^{-1} \bar{\mathbf{D}}^\pi \boldsymbol{r}\,, \quad q^\pi(\boldsymbol{z}) \;=\; (\boldsymbol{r} + \gamma \mathbf{P} \boldsymbol{a}^*)^\top \boldsymbol{\psi}(\boldsymbol{z})\,, \quad \forall \boldsymbol{z} \in \mathcal{X} \times \mathcal{A}\,. \tag{5.43}$$

[12] Discrete variables can be treated similarly with one discrete basis for each possible outcome.

[13] In difference to Proposition 3.6 on Page 35, this projected Bellman operator is not a contraction mapping w.r.t. norm $\|\cdot\|_\vartheta$. However, I have demonstrated in Section 3.3.3 that using a softmax policy improvement can stabilize LSPI. See Figure 3.4 on Page 37 for details.

The covariance matrix $\bar{\mathbf{C}}$ factorizes, that is, $\bar{\mathbf{C}} = \prod_{\alpha=1}^{d} \bar{\mathbf{B}}^{\alpha\top} \mathbf{C}^{\alpha} \bar{\mathbf{B}}^{\alpha} \in \mathbb{R}^{\bar{m}\times\bar{m}}$. Matrix $\bar{\mathbf{D}}^{\pi}$, on the other hand, depends on the non-factorizing policy π, which changes over the course of policy iteration. FAPI approximates the PDF $\frac{d\pi}{d\vartheta}(\boldsymbol{z}) \approx \boldsymbol{\varpi}^{\top}\boldsymbol{\psi}'(\boldsymbol{z}), \forall \boldsymbol{z} \in \mathcal{X}\times\mathcal{A}$, of π as a linear function of another set of factored basis functions $\{\psi'_k\}_{k=1}^{m'} \subset L^2(\mathcal{X}\times\mathcal{A}, \vartheta)$. These bases may be identical to $\{\psi_j\}_{j=1}^{m}$, but are in practice different to represent the policy better[14], or to adhere to additional constraints introduced in the policy improvement step (see below). Using the PDF in the policy operator, $\hat{\Gamma}_{\pi}[\psi_j] \approx \sum_{k=1}^{m'} \varpi_k \hat{\Gamma}_{\vartheta^a}[\psi'_k \cdot \psi_j]$, one can approximate matrix $\bar{\mathbf{D}}^{\pi} \in \mathbb{R}^{\bar{m}\times m}$:

$$\bar{D}^{\pi}_{ij} \;=\; \langle \bar{\psi}_i, \hat{\Gamma}_{\pi}[\psi_j]\rangle_{\vartheta^x} \;\approx\; \sum_{k=1}^{m'} \varpi_k \prod_{\alpha=1}^{d} \langle \bar{\psi}^{\alpha}_i, \psi'^{\alpha}_k \cdot \psi^{\alpha}_j\rangle_{\vartheta^\alpha} \prod_{\beta=d+1}^{d+b} \langle \psi'^{\beta}_k, \psi^{\beta}_j\rangle_{\vartheta^\beta}\,. \tag{5.44}$$

This involves a factorizing 3-tensor, which can be expensive to compute[15]. However, note that the point-wise product $\psi'^{\alpha}_k \cdot \psi^{\alpha}_j$ can be calculated like multiplying two LFF (Equation 5.11 on Page 79). For Fourier cosine bases $\boldsymbol{\phi}^{\alpha}$, the matrix $\bar{\mathbf{D}}^{\pi}$ can therefore be calculated analytically in two nested loops over k and α, that is:

$$\bar{\mathbf{D}}^{\pi} \;=\; \sum_{k=1}^{m'} \varpi_k \prod_{\alpha=1}^{d} \left(\bar{\mathbf{B}}^{\alpha\top} \tilde{\mathbf{B}}^{\alpha,k}\right) \cdot \prod_{\beta=d+1}^{d+b} \mathbf{1}\left(\boldsymbol{B}'^{\beta\top}_{:k} \mathbf{B}^{\beta}\right), \text{ with product parameters} \tag{5.45}$$

$$\tilde{B}^{\alpha,k}_{ij} \;:=\; \tfrac{1}{2}\sum_{q=1}^{i-1} B'^{\alpha}_{(i-q)k}\, B^{\alpha}_{qj} + \tfrac{1}{2}\sum_{q=i+1}^{m_\alpha} B'^{\alpha}_{(q-i)k}\, B^{\alpha}_{qj}\,, \qquad \tilde{\mathbf{B}}^{\alpha,k} \in \mathbb{R}^{m_\alpha\times m}\,. \tag{5.46}$$

The computational complexity of calculating $\bar{\mathbf{D}}^{\pi}$ is therefore $\mathcal{O}(mm'dm_\alpha(m_\alpha+\bar{m}))$ and memory complexity is $\mathcal{O}(\bar{m}m + m_\alpha m)$. Computing the equivalent matrix $\mathbf{D}$ for LSPI (defined in Equation 3.28 on Page 34) with a training-set of n samples has complexity $\mathcal{O}(nm^2dm_\alpha)$ and memory complexity $\mathcal{O}(m^2)$. In most applications holds $m_\alpha < \bar{m} < m$, and FAPI is therefore usually more memory efficient, and also faster when $\bar{m}m' \ll nm$. This is almost always the case for trainings sets that fully explore the environment (e.g. guided by algorithms like E^3, Kearns and Singh, 2002), as n grows exponentially with $d+b$, that is, $n \in \mathcal{O}(2^{d+b})$.

POLICY IMPROVEMENT

For POLICY IMPROVEMENT, FAPI chooses the next policy π_{i+1} such that the *average value* of previous Q-value function q^{π_i} (Equation 5.43) is maximized:

$$\sup_{\pi_{i+1}} \langle q^{\pi_i}, 1\rangle_{\vartheta^x \pi_{i+1}} \;\approx\; \sup_{\boldsymbol{\varpi}} \boldsymbol{\varpi}^{\top} \langle \boldsymbol{\psi}', \boldsymbol{\psi}^{\top}\rangle_{\vartheta} \left(\boldsymbol{r} + \gamma \mathbf{P}\boldsymbol{a}^*\right), \tag{5.47}$$

LP

This objective is a LINEAR OPTIMIZATION PROBLEM (LP, see Boyd and Vandenberghe, 2004, for an introduction). To ensure $\boldsymbol{\varpi}^{\top}\boldsymbol{\psi}'$ approximates the PDF $\frac{d\pi_{i+1}}{d\vartheta}$, the optimization must also adhere to the constraints of Radon-Nikodym derivatives:

$$\tfrac{d\pi_{i+1}}{d\vartheta}(\boldsymbol{x},\boldsymbol{a}) \;\geq\; 0 \quad \text{and} \quad \int \vartheta^a(d\boldsymbol{a}')\, \tfrac{d\pi_{i+1}}{d\vartheta}(\boldsymbol{x},\boldsymbol{a}') \;=\; 1, \quad \forall \boldsymbol{x}\in\mathcal{X}, \forall \boldsymbol{a}\in\mathcal{A}\,. \tag{5.48}$$

The first constraint can be guaranteed by restricting $\boldsymbol{\varpi}$ to be nonnegative, given all factored basis functions $\{\psi'_k\}_{k=1}^{m'}$ for the policy are nonnegative as well, that is,

[14] Note that this PDF of π is a LFF and can be compressed to speed up value estimation.

[15] When I published Böhmer and Obermayer (2013), I did not know about point-wise multiplication of LFF and suggested to pre-compute and store the more compact tensor of bases $\boldsymbol{\phi}^{\alpha}$, which requires $\mathcal{O}(dm_\alpha^3)$ memory: $\langle \psi^{\alpha}_i, \psi'^{\alpha}_k \cdot \psi^{\alpha}_j\rangle_{\vartheta^\alpha} = \sum_{u=1}^{m_\alpha}\sum_{v=1}^{m_\alpha}\sum_{w=1}^{m_\alpha} B^{\alpha}_{ui} B'^{\alpha}_{vk} B^{\alpha}_{wj} \langle \phi^{\alpha}_u, \phi^{\alpha}_v \cdot \phi^{\alpha}_w\rangle_{\vartheta^\alpha}$.

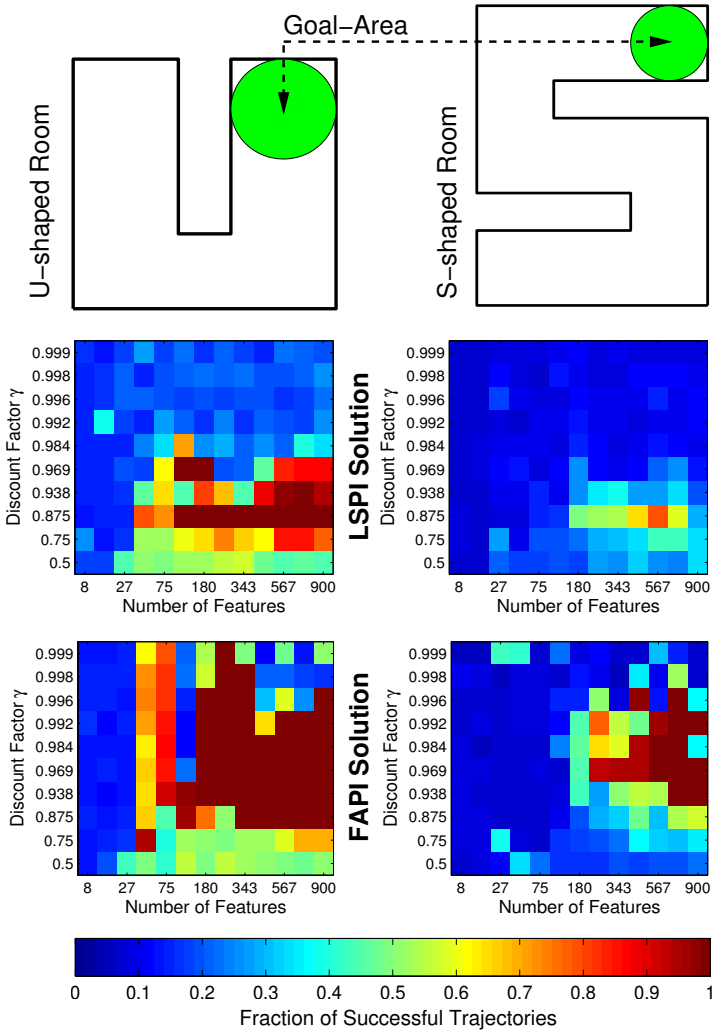


Figure 5.10: Two environments (top row) for navigation towards a goal-area. The position is the continuous 2d-state, the continuous action determines the direction of movement. Factored bases functions were pre-defined as the product of all Fourier bases up to some spatial and directional frequency. The middle row shows the fraction of successful trajectories of a policy learned by LSPI, with varying features and discounts. The bottom row shows the same for the published FAPI algorithm. Adapted from Böhmer and Obermayer (2013). See Appendix B.1 for details.

$\boldsymbol{\psi}'(\boldsymbol{z}) \succeq \mathbf{0}, \forall \boldsymbol{z} \in \mathcal{X} \times \mathcal{A}$. The second constraint can often not be fulfilled perfectly, but one can approximate it by enforcing the least-squares objective

$$\inf_{\boldsymbol{\varpi}} \left\| \boldsymbol{\varpi}^\top \hat{\Gamma}_{\vartheta^a}[\boldsymbol{\psi}'] - 1 \right\|^2_{\vartheta^x} \quad \Rightarrow \quad \boldsymbol{\varpi}^\top \langle \hat{\Gamma}_{\vartheta^a}[\boldsymbol{\psi}'], \hat{\Gamma}_{\vartheta^a}[\boldsymbol{\psi}'^\top] \rangle_{\vartheta^x} \overset{!}{=} \langle 1, \boldsymbol{\psi}'^\top \rangle_\vartheta \,. \tag{5.49}$$

Policy improvement yields therefore a LP with equality and inequality constraints:

$$\begin{aligned} \sup_{\boldsymbol{\varpi}} \quad & \boldsymbol{\varpi}^\top \langle \boldsymbol{\psi}', \boldsymbol{\psi}^\top \rangle_\vartheta \, (\boldsymbol{r} + \gamma \mathbf{P} \boldsymbol{a}^*) \\ \text{s.t.} \quad & \boldsymbol{\varpi} \succeq \mathbf{0} \quad \text{and} \quad \boldsymbol{\varpi}^\top \langle \hat{\Gamma}_{\vartheta^a}[\boldsymbol{\psi}'], \hat{\Gamma}_{\vartheta^a}[\boldsymbol{\psi}'^\top] \rangle_{\vartheta^x} \overset{!}{=} \langle 1, \boldsymbol{\psi}'^\top \rangle_\vartheta \,. \end{aligned} \tag{5.50}$$

All inner products factorize and can therefore be computed very efficiently. The LP can be solved using a variety of algorithms, many of which are implemented in MATLAB's `linprog` function. Note that the bases $\{\psi'_k\}_{k=1}^{m'}$ for the policy have to be nonnegative[16], for example, Gaussian bases depicted on the right of Figure 5.2.

Figure 5.10 from Böhmer and Obermayer (2013) shows the performance of FAPI compared with LSPI. The above value estimation algorithm behaves very similar and a quantitative evaluation is omitted here. The navigation task in Figure 5.10

[16] In Section 5.5.4 I discuss a potential extension for arbitrary factored basis functions $\boldsymbol{\psi}'$.

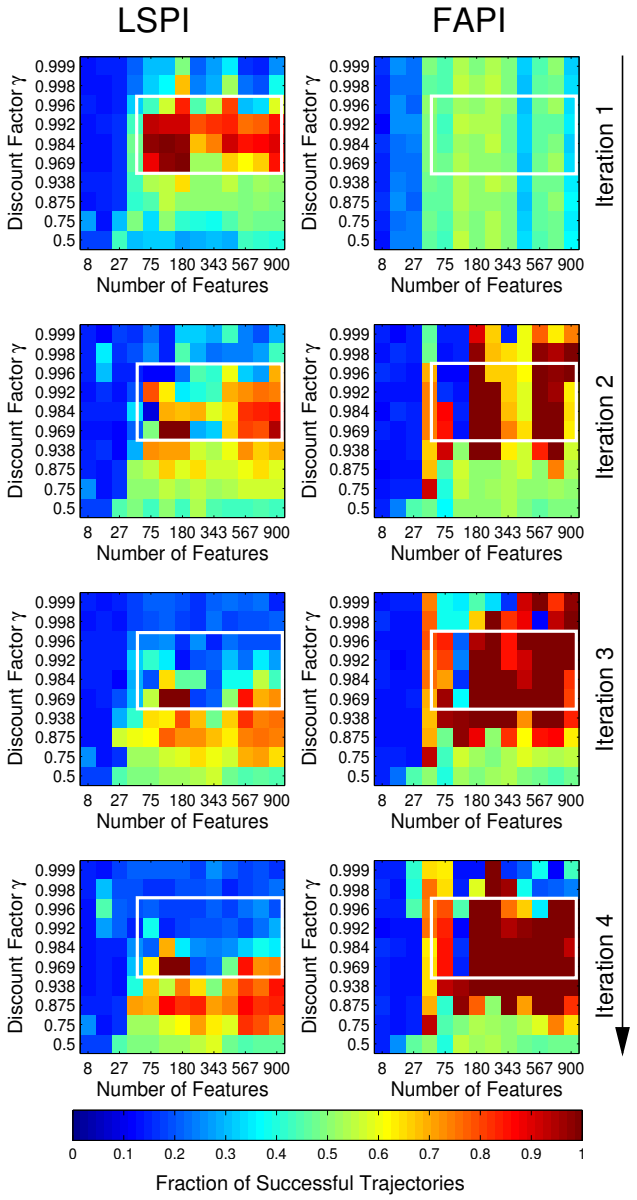
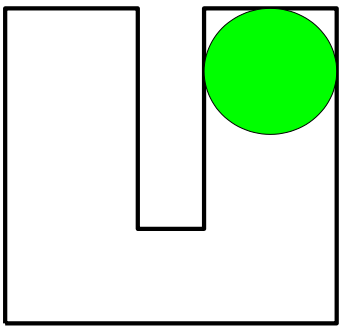


Figure 5.11: Development of navigation performance over iterations of policy improvement in LSPI and FAPI at the example of the U-shaped room in Figure 5.10 (see also the sketch ploted below). The white box denotes the parameter region in which the first iteration of LSPI works well. Note that LSPI becomes unstable here, whereas FAPI performance steadily increases. Adapted from Böhmer and Obermayer (2013). See Appendix B.1 for details.



has two continuous state (the position) and one continuous action dimension (the direction of movement). Performance is measured as the fraction of test trajectories that reach the rewarded goal area from random starting positions within 100 steps. The experiment was repeated in two environments (upper row) with LSPI (with greedy policy improvement, middle row) and FAPI (lower row). Each algorithm was tested with varying number of state-action features m (x-axis) and discount factors γ (y-axis). Note that both algorithms solve the task within (almost) the same range of features, that is, for $m \geq 48$ in the U-shaped room and $m \geq 180$ in the S-shaped room, although the latter shows a clear advantage of FAPI. Comparing the performance for different discount factors, on the other hand, reveals very different behavior. Here LSPI becomes unstable for large γ, whereas FAPI shows stable behavior in a much larger range. This effect is exemplified in Figure 5.11 by comparing the performance of policies learned by LSPI and FAPI over the course of

policy iteration. Note that LSPI performs initially well for larger γ (marked by a white box), but this advantage degrades quickly, whereas the performance of FAPI steadily improves. One explanation may be that FAPI policy improvement resembles a softmax due to the constraints introduced by the factored basis functions ψ'_k (see Lemma 3.4). Figure 3.4 on Page 37 shows how the introduction of softmax policy improvements stabilize LSPI for large γ. Although I did not test it, I expect therefore the stability of FAPI to depend largely on the detail in which the policy can be represented, that is, the number and shape of the non-negative basis functions ψ'_k used in FAPI policy improvement.

Although the above argument demonstrates the advantage of a *restricted* set of basis functions for FAPI policy improvement, the reliance on pre-determined bases is also its Achilles' heel. In the following subsections I will show how $\{\psi_j\}_{j=1}^{m}$ and $\mathbf{P}$ can be derived from $\{\bar{\psi}_i\}_{i=1}^{\bar{m}}$ and a DBN of the environment, but both $\{\bar{\psi}_i\}_{i=1}^{\bar{m}}$ and $\{\psi'_i\}_{k=1}^{\bar{m}'}$ must be specified before planning. General bases of $L^2(\mathcal{Z}, \vartheta)$ grow exponential in the dimensionality $d + b$ of states and actions, but choosing factored basis functions too restricted will restrain value functions and policies learned by FAPI. Without significant insight in the problem domain, the presented algorithm can therefore not solve planning problems in large domains. This would require to perform both value estimation and policy improvement directly in the space of LFF, and is discussed further in Sections 5.3.4 and 5.5.4.

5.3.2 Dynamic Bayesian networks (DBN)

One approach to generalize transitions of unseen states are DYNAMIC BAYESIAN NETWORKS (DBN, Dagum et al., 1992; Boutilier et al., 1999). These graphical models represent variables[17] as nodes and transition dependencies as edges of a directed graph. The transition probabilities P^α of each variable α depend here *only*[18] on the PARENT-VARIABLES $\boldsymbol{s}^\alpha \subset \boldsymbol{x} \cup \boldsymbol{a}$, that is, x_α is CONDITIONALLY INDEPENDENT given $\boldsymbol{s}^\alpha$. This implies two assumptions: (i) the transition of each state-variable is drawn independently, i.e. $P(d\boldsymbol{x}'|\boldsymbol{z}) = \prod_{\alpha=1}^{d} P^\alpha(dx'_\alpha|\boldsymbol{z})$, and (ii) the transition of each state-variable depends only on a small set of variables, i.e. $P^\alpha(dx'_\alpha|\boldsymbol{z}) = P^\alpha(dx'_\alpha|\boldsymbol{s}^\alpha)$:

$$P(d\boldsymbol{x}'|\boldsymbol{z}) \;:=\; \prod_{\alpha=1}^{d} P^\alpha(dx'_\alpha|\boldsymbol{s}^\alpha), \quad \boldsymbol{s}^\alpha \subset \boldsymbol{z}\,, \quad \forall \boldsymbol{z} \in \mathcal{X} \times \mathcal{A}\,, \quad \forall dx'_\alpha \in \mathcal{B}(\mathcal{X}_\alpha)\,. \tag{5.51}$$

These factored transition models are usually treated as prior knowledge and the majority of factored MDP literature aims to exploit the so called FACTORED MDP to solve control problems in large, discrete domains (see e.g. Guestrin et al., 2003b). FACTORED MDP

Efficient representation of the transition model does not imply efficient computation of the corresponding value function, though. For example, LSTD requires inner products $\langle \phi_i, \hat{\mathrm{P}}^\pi[\phi_j] \rangle_\vartheta$, which have to be estimated as a mean of $\phi_i(\boldsymbol{z}) \cdot \hat{\mathrm{P}}^\pi[\phi_j](\boldsymbol{z})$ over all state(-action)s $\boldsymbol{z} \in \mathcal{Z}$. In large state spaces, this is not feasible. However, the inner product can be computed efficiently if all basis functions have only a limited scope, that is, if each basis function ϕ_i depends only on a small subset of variables $\bar{\boldsymbol{s}}^i \subset \boldsymbol{z}$ (Koller and Parr, 1999; Poupart et al., 2002). Although not all factored MDP

[17] In the following I will use the terms *state(-action) dimension* and *variable* interchangeably.

[18] There exist another definition of DBN that includes dependencies between future variables (Boutilier et al., 1999). I restrict myself here to the independent case to allow factorization. The possibility of "deep" DBN to model these dependencies is further explored in Section 5.5.3.

have value functions that can be approximated this way (see Allender et al., 2002, for a counter-example), the representation appears to be sufficient for many domains (see below). Computationally, the inner product $\langle \phi_i, \phi_j \rangle_\vartheta$ can be estimated as an integral over $\bar{\boldsymbol{s}}^i \cup \bar{\boldsymbol{s}}^j$. In principle, this would also allow to compute the inner product $\langle \phi_i, \hat{\mathrm{P}}^\pi[\phi_j] \rangle_\vartheta$ as an integral over $\bar{\boldsymbol{s}}_i$ and all *parent variables* of $\bar{\boldsymbol{s}}^j$. Without a factorizing policy, however, the integral can not be simplified this way. So far the only proposed remedy for this problem is a DECISION LIST, a list of sparse conditionals and associated actions in discrete state-action spaces (Koller and Parr, 2000).

ALP In practice these integrals can be avoided by using APPROXIMATE LINEAR PROGRAMMING (ALP, Guestrin et al., 2001a). ALP formulates the value estimation problem under a MAX-NORM PROJECTION, i.e. $\inf_f \|\hat{\Pi}_\infty[r + \hat{\mathrm{P}}^\pi[f]] - f\|_\infty$, as a linear program (LP, see Boyd and Vandenberghe, 2004, for an introduction). Each state-action pair translates into an inequality constraint and one can use variable substitution to reduce the number of constraints to polynomial in the number of discrete state-action variables. This technique became very popular and has been modified to solve a variety of tasks. Examples are multi-agent planning (Guestrin et al., 2001b), partially observable factored MDP (Guestrin et al., 2001c), relational MDP (Guestrin et al., 2003a) and uncertain models (Delgado et al., 2009). ALP has been extended to continuous variables (HALP, Guestrin et al., 2004), which is restricted to certain parameterizable transition distributions (Hauskrecht and Kveton, 2003, 2006). HALP remains popular to solve factored MDP and alternative approaches are rare (but do exist, for example using restricted Boltzmann machines, Sallans and Hinton, 2004).

5.3.3 Learning DBN with LFF

This thesis proposes LFF as representations to solve control problems deductively. LFF have four advantages over the factored ALP approaches described above:

- The (Q-)value function can be approximated more precisely, as the scope of a factored function is not restricted (in contrast to the sparse basis functions used in HALP, Allender et al., 2002).
- Continuous DBN transition models can be represented by d multivariate LFF.
- These LFF can be learned by regression over the associated parent variables. Transitions models can therefore be learned inductively without the need to explore every state-action of the environment.
- The computations necessary to estimate (Q-)value functions can be broken down to a series of point-wise multiplications, compressions and factorizing integrals. Deductive value estimation in exponentially growing state-action spaces should therefore be possible in sub-exponential time.

In the following I will address these advantages, starting with the representation of DBN transition models by LFF. Using the DBN definition in Equation 5.51, then for every LFF $f \in \mathcal{F}^m \subset L^2(\mathcal{X}, \vartheta^x)$ holds, $\forall \boldsymbol{z} \in \mathcal{X} \times \mathcal{A}$:

$$\hat{\mathrm{P}}[f](\boldsymbol{z}) \;=\; \sum_{i=1}^{m} a_i \int \prod_{\alpha=1}^{d} P^\alpha(dx'_\alpha | \boldsymbol{s}^\alpha)\, \psi_i^\alpha(x'_\alpha) \;=\; \sum_{i=1}^{m} a_i \prod_{\alpha=1}^{d} \sum_{j=1}^{m_\alpha} B_{ji}^\alpha \underbrace{\hat{\mathrm{P}}^\alpha[\phi_j^\alpha](\boldsymbol{s}^\alpha)}_{\approx\, \hat{\varphi}_j^\alpha \in \mathcal{F}^{\hat{m}_\alpha}} . \quad (5.52)$$

Note that the transition model P^α for each state-variable x_α can be represented by m_α functions $\hat{\mathrm{P}}^\alpha[\phi^\alpha_j] \in L^2(\mathcal{S}^\alpha, \vartheta)$, where $\mathcal{S}^\alpha$ is the subspace of $\mathcal{X}$ spanned by the parent variables $\boldsymbol{s}^\alpha$ of x_α. These functions can be estimated by LFF using regression (see Section 5.2.2). After learning, all LFF representing $\hat{\mathrm{P}}^\alpha$ can be compressed into a multivariate LFF (see Section 5.1.4), which will be denoted in the following as:

$$\hat{\mathrm{P}}^\alpha[\phi^\alpha_j](\boldsymbol{s}^\alpha) \quad \approx \quad \hat{\varphi}^\alpha_j(\boldsymbol{z}) \quad := \quad \sum_{i=1}^{\hat{m}_\alpha} \hat{A}^\alpha_{ij} \prod_{\beta=1}^{d+b} \sum_{k=1}^{m_\beta} \hat{B}^{\alpha,\beta}_{ki}\, \phi^\beta_k(z_\beta)\,, \qquad \forall \boldsymbol{z} \in \mathcal{X} \times \mathcal{A}\,. \tag{5.53}$$

The collection of d multivariate functions $\{\hat{\boldsymbol{\varphi}}^\alpha \in \mathcal{F}^{\hat{m}_\alpha \times m_\alpha}\}_{\alpha=1}^d$ fully describes the DBN P and can be learned from interaction of the environment. Applying such a DBN transition operator to a LFF $f \in \mathcal{F}^m \subset L^2(\mathcal{X}, \vartheta^x)$ over state space $\mathcal{X}$ yields:

$$\begin{aligned}
\hat{\mathrm{P}}[f](\boldsymbol{z}) \quad &= \quad \sum_{i=1}^{m} a_i \prod_{\alpha=1}^{d} \sum_{j=1}^{m_\alpha} B^\alpha_{ji} \overbrace{\hat{\mathrm{P}}^\alpha[\phi^\alpha_j](\boldsymbol{s}^\alpha)}^{\approx \hat{\varphi}^\alpha_j(\boldsymbol{z})} \\
&\approx \quad \sum_{i=1}^{m} a_i \prod_{\alpha=1}^{d} \underbrace{\sum_{k=1}^{\hat{m}_\alpha} (\hat{\mathbf{A}}^\alpha \mathbf{B}^\alpha)_{ki} \prod_{\beta=1}^{d+b} \sum_{j=1}^{m_\beta} \hat{B}^{\alpha,\beta}_{jk}\, \phi^\beta_j(z_\beta)}_{(\mathbf{B}^{\alpha\top} \hat{\boldsymbol{\varphi}}^\alpha)_i \in \mathcal{F}^{\hat{m}_\alpha}} \Big)\,.
\end{aligned} \tag{5.54}$$

Computing the transition operator of a LFF f requires therefore the point-wise multiplication of d multivariate LFF $\mathbf{B}^{\alpha\top} \hat{\boldsymbol{\varphi}}^\alpha \in \mathcal{F}^{\hat{m}_\alpha \times m}$. It will in practice be necessary to compress the resulting functions in between multiplications[19], to prevent an exponential growth of factored basis functions.

For example, in the context of the FAPI algorithm in Section 5.3.1, the "global" DBN transition operator $\hat{\mathrm{P}}$ can be approximated from "local" models $\{\hat{\boldsymbol{\varphi}}^\alpha\}_{\alpha=1}^d$:

$$\hat{\mathrm{P}}[\bar{\boldsymbol{\psi}}](\boldsymbol{z}) \quad \approx \quad \langle \tilde{\boldsymbol{\psi}}, \boldsymbol{\psi}^\top \rangle_\vartheta \langle \boldsymbol{\psi}, \boldsymbol{\psi}^\top \rangle_\vartheta^{-1} \boldsymbol{\psi}(\boldsymbol{z}) \quad =: \quad \mathbf{P}^\top \boldsymbol{\psi}(\boldsymbol{z})\,, \quad \forall \boldsymbol{z} \in \mathcal{X} \times \mathcal{A}\,. \tag{5.55}$$

Here $\tilde{\boldsymbol{\psi}} \in \mathcal{F}^{\tilde{m} \times \bar{m}}$ denotes the multivariate LFF which predicts $\hat{\mathrm{P}}[\bar{\boldsymbol{\psi}}]$:

$$\tilde{\boldsymbol{\psi}}(\boldsymbol{z}) \quad := \quad \tilde{\mathbf{A}}^\top \Big(\prod_{\alpha=1}^{d+b} \tilde{\mathbf{B}}^{\alpha\top} \boldsymbol{\phi}^\alpha(z_\alpha) \Big) \quad \overset{!}{\approx} \quad \prod_{\alpha=1}^{d} \bar{\mathbf{B}}^{\alpha\top} \hat{\boldsymbol{\varphi}}^\alpha(\boldsymbol{z}) \quad = \quad \hat{\mathrm{P}}[\bar{\boldsymbol{\psi}}](\boldsymbol{z})\,, \tag{5.56}$$

$\forall \boldsymbol{z} \in \mathcal{X} \times \mathcal{A}$. The parameter matrix $\mathbf{P}$ can be computed very efficient after $\tilde{\boldsymbol{\psi}} \in \mathcal{F}^{\tilde{m}}$ has been determined, as both inner products factorize:

$$\mathbf{P} \quad = \quad \Big(\prod_{\alpha=1}^{d+b} \mathbf{B}^{\alpha\top} \mathbf{C}^\alpha \mathbf{B}^\alpha \Big)^{-1} \Big(\prod_{\alpha=1}^{d+b} \mathbf{B}^{\alpha\top} \mathbf{C}^\alpha \tilde{\mathbf{B}}^\alpha \Big)\, \tilde{\mathbf{A}}\,. \tag{5.57}$$

Moreover, the calculation of $\tilde{\boldsymbol{\psi}}$ could be used to *define* the factored state-action bases $\boldsymbol{\psi}$ (as indicated in Equation 5.40), using $\tilde{\mathbf{A}}$ as $\mathbf{P}$ and all $\tilde{\mathbf{B}}^\alpha$ as $\mathbf{B}^\alpha$.

Figure 5.12 on the next page shows the advantages and limits of learning such a DBN with LFF. The major advantage over classical model-based learning is that here each sub-model P^α requires only training data from one state-variable. The horizontal P^1 can be learned by executing the actions "left" and "right" near all horizontal positions, and the vertical P^2 must execute the actions "up" and "down" near all vertical positions. A random-walk in the yellow, L-shaped region yields therefore

[19] The order of multiplications is irrelevant. However, following the dependency structure of the DBN should keep unrelated variables separated as long as possible, and although I have not tested this, I expect this order to reduce the numerical error induced by the repeated compressions.

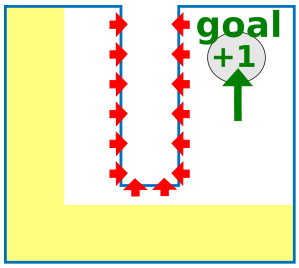


Figure 5.12: Sketch of a U-shaped environment for navigation. Movement in x_1 and x_2 can be modeled independently, that is, $P(d\boldsymbol{x}'|\boldsymbol{x},a) = P^1(dx'_1|\boldsymbol{x},a) \cdot P^2(dx'_2|\boldsymbol{x},a)$. Transition of each state-variable depends almost everywhere only on the respective variable and the direction of movement (i.e. action a), e.g. $P^1(dx'_1|\boldsymbol{x},a) = P^1(dx'_1|x_1,a)$. These models can be estimated perfectly using samples from the yellow area. The resulting transition model P predicts most state-actions correctly, except when the agent collides with the central wall (indicated by red arrows).

sufficient training data to generalize transitions over the *entire* environment. The disadvantage of sparse DBN becomes apparent at the central wall that extends in the otherwise square room. Transitions shown as red arrows are blocked by the wall and violate thus the predictions of the model $P = P^1 \cdot P^2$. These violations are a hallmark of *every* sparse DBN in an environment that can not be perfectly factored by it. However, as long as the model is right in *most* states, on could use a combination of deductive and inductive learning (see Section 5.1.1) to correct these model errors.

5.3.4 Deductive value estimation with LFF

In Section 5.3.1, FAPI shows how value estimation and policy improvement can be performed deductively. However, the dependence on pre-defined basis functions prohibits an application in high dimensional state and action spaces without extensive expert knowledge. In this section I will define a possible deductive algorithm based on LFF techniques, that estimates Q-values directly in the space of LFF.

Deductive Q-value estimation requires to solve $\inf_f \|r + \gamma \hat{\mathrm{P}}[\hat{\Gamma}_\pi[f]] - f\|^2_\vartheta$. This is usually a problem for factored MDP (Koller and Parr, 2000), as the policy π (and thus the function $\hat{\Gamma}_\pi[f]$) must not be restricted to a sparse scope. LFF can cope with this problem by representing the policy (i.e. the PDF $\frac{d\pi}{d\vartheta}$) also as a LFF $h^\pi \in \mathcal{F}^{m_\pi}$. Applying the policy operator to a function $f \in \mathcal{F}^m \subset L^2(\mathcal{X} \times \mathcal{A}, \vartheta)$ yields $\hat{\Gamma}_\pi[f] = \hat{\Gamma}_\vartheta[\frac{d\pi}{d\vartheta} \cdot f] \approx \hat{\Gamma}_\vartheta[h^\pi \cdot f] =: f^\pi \in \mathcal{F}^{m m_\pi} \subset L^2(\mathcal{X}, \vartheta)$. The function f^π, resulting from point-wise multiplication and marginalization (see Section 5.1.2), has $m\, m_\pi$ basis functions, but can often be compressed significantly (see Section 5.1.3).

The Q-value can be estimated in two different ways. First, one could reframe Section 5.2.3 as deductive Q-value estimation by using the cost $\|\mathrm{td}\|^2_\vartheta$ instead of $\|\mathrm{td}\|^2_\zeta$. This implies to compute the following matrix and vector in the inner loop:

$$\bar{\mathbf{D}}_\alpha := \langle \boldsymbol{\phi}^\alpha \cdot \prod_{\beta \neq \alpha} g^\beta, \hat{\mathrm{P}}\Big[\hat{\Gamma}_\pi\big[\prod_{\beta \neq \alpha} g^\beta \cdot \boldsymbol{\phi}^{\alpha\top}\big]\Big] \rangle_\vartheta \quad \in \quad \mathbb{R}^{m_\alpha \times m_\alpha} \tag{5.58}$$

$$\bar{\boldsymbol{d}}_\alpha := \langle \boldsymbol{\phi}^\alpha \cdot \prod_{\beta \neq \alpha} g^\beta, \mathrm{td} \rangle_\vartheta - \gamma \langle \hat{\mathrm{P}}\Big[\hat{\Gamma}_\pi\big[\boldsymbol{\phi}^\alpha \cdot \prod_{\beta \neq \alpha} g^\beta\big]\Big], \mathrm{td} \rangle_\vartheta \quad \in \quad \mathbb{R}^{m_\alpha}. \tag{5.59}$$

Given a LFF approximation of $\hat{\mathrm{P}}[\hat{\Gamma}_\pi[\boldsymbol{\phi}^\alpha \cdot \prod_{\beta \neq \alpha} g^\beta]]$, the inner products factorize and can therefore be computed efficiently. The approximation can be performed using the PDF $h^\pi \in \mathcal{F}^m$ (see above) and the LFF approximation of DBN P, shown in Equation 5.54. Repeating this each iteration of the inner loop may be computationally expensive and although the algorithm should scale polynomial in the dimensionality of the state-action space, practical application may not be feasible.

Alternatively, Algorithm 9 estimates the Q-value by borrowing the concept of Bellman-error basis functions (BEBF, Parr et al., 2007, see Section 4.1.2). Here each new basis function g is constructed to resemble the TD-error of the current solution f. Applied to LFF, the deductive optimization problem of the inner loop is

$$\inf_{g\in\mathcal{F}} \Big\| \underbrace{r + \gamma \hat{\mathrm{P}}\big[\hat{\Gamma}_\pi[f]\big] - f}_{\mathtt{td}\,\in\,\mathcal{F}^m} - g \Big\|_\vartheta^2 \qquad \text{s.t.} \quad \|g^\alpha\|_{\vartheta^\alpha} = 1\,, \quad \forall \alpha \in \{1,\dots,d+b\}\,. \tag{5.60}$$

The outer loop of Algorithm 9 is structured in three parts: first, an inner loop solves the above optimization problem for a given TD-error td. Second, the newly

Algorithm 9 – deductive Q-value estimation based on LFF DBN and reward

input: $r = \bar{\boldsymbol{a}}^\top \prod_{\alpha=1}^{d+b} \bar{\mathbf{B}}^{\alpha\top} \boldsymbol{\phi}^\alpha \in \mathcal{F}^{\bar{m}}$, $\{\hat{\boldsymbol{\varphi}}^\alpha \in \mathcal{F}^{\hat{m}_\alpha \times m_\alpha}\}_{\alpha=1}^d$, $h^\pi \in \mathcal{F}^{m_\pi}$, $\gamma \in [0,1)$

$\mathrm{err} := \infty$; $\boldsymbol{a} := \emptyset$; $\hat{\mathbf{A}} := \emptyset$; $\mathbf{B}^\alpha := \emptyset$; $\hat{\mathbf{B}}^\alpha := \emptyset$, $\forall\alpha$ // initialize

while $\mathrm{err} > \epsilon$ **do**

// Find next factored BEBF basis function g

$g^\alpha := 1$; $\boldsymbol{c}^\alpha := \mathbf{1}$; $d_\alpha := \infty$, $\forall\alpha$ // initialize new g

while $\max_\alpha\{d_\alpha\} > \varepsilon$ **do**

for $\alpha := \arg\max_\alpha\{|d_\alpha|\}$ **do**

$\boldsymbol{b}_{\mathtt{uc}} := [\bar{\mathbf{B}}^\alpha, \hat{\mathbf{B}}^\alpha, \mathbf{B}^\alpha]\Big(\Big[\gamma\hat{\mathbf{A}}\begin{matrix}\bar{\boldsymbol{a}}\\ \boldsymbol{a}\end{matrix}\\ -\boldsymbol{a}\Big] \cdot \prod_{\beta\neq\alpha} \boldsymbol{c}^\beta\Big)$ // $\boldsymbol{b}_{\mathtt{uc}} \overset{!}{=} (\mathbf{C}^\alpha)^{-1} \frac{\partial}{\partial \boldsymbol{b}^\alpha}\langle g, \mathtt{td}\rangle_\vartheta$

$d_\alpha := 2\,\boldsymbol{b}_{\mathtt{uc}}^\top \mathbf{C}^\alpha \boldsymbol{b}^\alpha$ // $d_\alpha = \|g' - \mathtt{td}\|_\vartheta^2 - \|g'\|_\vartheta^2 - \|\mathtt{td}\|_\vartheta^2$

$\boldsymbol{b}^\alpha := \boldsymbol{b}_{\mathtt{uc}} / \sqrt{\boldsymbol{b}_{\mathtt{uc}}^\top \mathbf{C}^\alpha \boldsymbol{b}_{\mathtt{uc}}}$ // enforce constraint $\|g^\alpha\|_{\vartheta^\alpha} \overset{!}{=} 1$

$d_\alpha := 2\,\boldsymbol{b}_{\mathtt{uc}}^\top \mathbf{C}^\alpha \boldsymbol{b}^\alpha - d_\alpha$ // $d_\alpha = \|g' - \mathtt{td}\|_\vartheta^2 - \|g - \mathtt{td}\|_\vartheta^2$

$\boldsymbol{c}^\alpha := [\bar{\mathbf{B}}^\alpha, \hat{\mathbf{B}}^\alpha, \mathbf{B}^\alpha]\mathbf{C}^\alpha \boldsymbol{b}^\alpha$ // recompute coefficients

end for

end while // end inner loop: g is new basis

// Propagate newly found basis g: $\hat{g} \approx \hat{\mathrm{P}}[\hat{\Gamma}_\pi[g]]$

$\tilde{g} := \hat{\Gamma}_{\vartheta^a}[h^\pi \cdot g] \in \mathcal{F}^{m_\pi}$; $\hat{\boldsymbol{\psi}}' := \mathbf{1} \in \mathcal{F}^{1\times m_\pi}$ // apply policy: $\tilde{g} \approx \hat{\Gamma}_\pi[g]$

for α **in** $\{1,\dots,d\}$ **do**

$\hat{\boldsymbol{\psi}}' := \mathtt{compress}\big(\hat{\boldsymbol{\psi}}' \cdot (\tilde{\mathbf{B}}^{\alpha\top}\hat{\boldsymbol{\varphi}}^\alpha)\big) \in \mathcal{F}^{m'\times m_\pi}$ // multiply $\hat{\mathrm{P}}^\alpha[\tilde{\boldsymbol{\psi}}^\alpha] = \tilde{\mathbf{B}}^{\alpha\top}\hat{\boldsymbol{\varphi}}^\alpha$

end for

$\hat{g} := \tilde{\boldsymbol{a}}^\top \hat{\boldsymbol{\psi}}' \in \mathcal{F}^{m'}$ // $\hat{g} \approx \hat{\mathrm{P}}[\tilde{g}]$ is propagated $\tilde{g}$

// Update current Q-value estimate $f = \boldsymbol{a}^\top \prod_{\alpha=1}^{d+b} \mathbf{B}^{\alpha\top}\boldsymbol{\phi}^\alpha$

$\mathbf{B}^\alpha := [\mathbf{B}^\alpha, \boldsymbol{b}^\alpha]$, $\forall\alpha$; $\hat{\boldsymbol{\psi}} := \mathtt{compress}\Big(\Big[\begin{matrix}\hat{\boldsymbol{\psi}}\\ \hat{g}\end{matrix}\Big]\Big)$ // update bases

$$\boldsymbol{a} := \Bigg(\overbrace{\Big(\prod_{\alpha=1}^{d+b} \mathbf{B}^{\alpha\top}\mathbf{C}^\alpha\mathbf{B}^\alpha\Big)}^{\mathbf{C}:=\langle\boldsymbol{\psi},\boldsymbol{\psi}^\top\rangle_\vartheta} - \gamma \overbrace{\Big(\prod_{\alpha=1}^{d+b} \mathbf{B}^{\alpha\top}\mathbf{C}^\alpha\hat{\mathbf{B}}^\alpha\Big)}^{\mathbf{D}:=\langle\boldsymbol{\psi},\hat{\boldsymbol{\psi}}^\top\rangle_\vartheta}\hat{\mathbf{A}}\Bigg)^{-1} \overbrace{\Big(\prod_{\alpha=1}^{d+b} \mathbf{B}^{\alpha\top}\mathbf{C}^\alpha\bar{\mathbf{B}}^\alpha\Big)}^{\boldsymbol{b}:=\langle\boldsymbol{\psi},r\rangle_\vartheta}\bar{\boldsymbol{a}}$$ // LSTD

$$\mathrm{err} := \Big[\gamma\hat{\mathbf{A}}\begin{matrix}\bar{\boldsymbol{a}}\\ \boldsymbol{a}\end{matrix}\\ -\boldsymbol{a}\Big]^\top \Big(\prod_{\alpha=1}^{d+b} [\bar{\mathbf{B}}^\alpha, \hat{\mathbf{B}}^\alpha, \mathbf{B}^\alpha]^\top \mathbf{C}^\alpha [\bar{\mathbf{B}}^\alpha, \hat{\mathbf{B}}^\alpha, \mathbf{B}^\alpha]\Big)\Big[\gamma\hat{\mathbf{A}}\begin{matrix}\bar{\boldsymbol{a}}\\ \boldsymbol{a}\end{matrix}\\ -\boldsymbol{a}\Big]$$ // $\mathrm{err} = \|\mathtt{td}\|_\vartheta^2$

end while // end outer loop: $\|\mathtt{td}\|_\vartheta^2 \le \epsilon$

output: $\boldsymbol{a} \in \mathbb{R}^m$, $\{\mathbf{B}^\alpha \in \mathbb{R}^{m_\alpha \times m}\}_{\alpha=1}^{d+b}$ // the final Q-value estimate f

found factored BEBF basis function $g \in \mathcal{F}$ is propagated, that is, $\hat{g} \approx \hat{\mathrm{P}}[\hat{\Gamma}_\pi[g]]$, using the above PDF h^π and the methodology of Section 5.3.3. Note that I use here the multivariate `compress` function, derived in Section 5.1.4. Both g and $\hat{g}$ are remembered with all previous bases as the multivariate LFF $\boldsymbol{\psi} := \prod_{\alpha=1}^{d+b} \mathbf{B}^{\alpha\top} \boldsymbol{\phi}^\alpha$ and $\hat{\boldsymbol{\psi}} := \hat{\mathbf{A}}^\top \prod_{\alpha=1}^{d+b} \hat{\mathbf{B}}^{\alpha\top} \boldsymbol{\phi}^\alpha \approx \hat{\mathrm{P}}[\hat{\Gamma}_\pi[\boldsymbol{\psi}]]$. The latter can additionally be compressed to eliminate any inefficient overlap. Lastly, the multivariate $\boldsymbol{\psi}$ and $\hat{\boldsymbol{\psi}}$ are used to compute the matrices $\mathbf{C}$, $\mathbf{D}$ and $\mathbf{b}$, analog to LSTD (Section 3.3.2) or FAPI (Section 5.3.1). As all matrices factorize, calculating the linear weights $\boldsymbol{a} \in \mathbb{R}^m$ of the Q-value function f with the current factored bases $\boldsymbol{\psi}$ is possible with linear complexity in the dimensionality d and b. The above results can also be used to define the TD-error indirectly as $\mathrm{td} = r + \gamma \boldsymbol{a}^\top \hat{\boldsymbol{\psi}} - \boldsymbol{a}^\top \boldsymbol{\psi}$. This TD-error is utilized in the inner loop and also allows to terminate the outer loop when $\|\mathrm{td}\|_\vartheta^2 \leq \epsilon$. Algorithm 9 scales polynomially with the state-action dimensionality as well, but should be significantly faster and probably more precise than the first method mentioned above.

5.4 Symbolically parameterized mixture models

I demonstrate in Section 5.3 how sparse DBN can be exploited to learn a transition model inductively and to estimate the Q-value deductively. I also show the limits of DBN to model many realistic domains, for example, navigation in environments like the one sketched in Figure 5.12 on Page 102. These limits often induce errors, which must be corrected by interaction with the environment (see Section 5.1.1).

This section extends the ability to model more realistic, complex environments with a mixture of DBN experts. The approach is based on conditionals to decide which expert is responsible to predict the transition for the state-action pair at hand. These conditionals must be deduced from some kind of "rules", which also tackles an extended definition of generalization: OBJECT ORIENTED ABSTRACTION allows to generalize over all environments that are composed of objects from known CLASSES. Here the properties of all present objects correspond to the state space. Each environment has therefore a different state dimensionality d. Expert DBN and their conditionals can be constructed from relationships between objects, that is, from a set of relational rules. This allows to generate a unique mixture-of-expert DBN for each environment that is composed of objects from known classes.

OBJECT ORIENTATION

5.4.1 Mixture of expert DBN

To increase the predictive power of DBN, I propose here a mixture-of-experts extension. Each of the K experts is a sparse DBN $P_k := \prod_{\alpha=1}^{d} P_k^\alpha$, which correctly predicts *some* transitions. MUTUALLY EXCLUSIVE CONDITIONS $\{c_k \in L^2(\mathcal{X} \times \mathcal{A}, \vartheta)\}_{k=1}^K$ determine *when* each expert governs the behavior. Condition $c_k(\boldsymbol{z})$ corresponds to the probability that expert k can predict the transitions of state-action pair $\boldsymbol{z} \in \mathcal{X} \times \mathcal{A}$:

CONDITIONS

$$c_k(\boldsymbol{z}) \;\geq\; 0\,, \quad \forall k \in \{1, \ldots, K\}\,, \qquad \sum_{k=1}^{K} c_k(\boldsymbol{z}) \;\leq\; 1\,, \qquad \forall \boldsymbol{z} \in \mathcal{X} \times \mathcal{A}\,. \tag{5.61}$$

Conceptually, conditions can be regarded as binary indicators of disjunct subsets of $\mathcal{X} \times \mathcal{A}$, which reflect the applicable dynamics in the environment. However, the above definition allows also to express uncertainty, either as a result of inductive learning or to enforce smoothness in the value functions. Additionally, there must be a DEFAULT DBN P_0, which is chosen whenever no condition applies, that is, with the

DEFAULT DBN

condition $c_0(\boldsymbol{z}) := 1 - \sum_{k=1}^{K} c_k(\boldsymbol{z})$. For example, Newton's laws of motion from the beginning of this chapter have two types of experts: the first law corresponds to the default DBN P_0, which models each object to keep moving independently. The third law describes how two objects bounce off each other in the case of collisions. Note that each combination of colliding objects, albeit governed by the same equations, has a different DBN P_k and condition c_k. A system with o objects can therefore be described by $K = \frac{o^2-o}{2}$ third-law experts, each exclusively describing the collision of two objects. Combining K experts yields the transition operator $\hat{\mathrm{P}}$, $\forall \boldsymbol{z} \in \mathcal{X} \times \mathcal{A}$:

$$\hat{\mathrm{P}}[f](\boldsymbol{z}) \quad := \quad \hat{\mathrm{P}}_0[f](\boldsymbol{z}) \quad + \quad \sum_{k=1}^{K} c_k(\boldsymbol{z}) \left(\hat{\mathrm{P}}_k[f](\boldsymbol{z}) - \hat{\mathrm{P}}_0[f](\boldsymbol{z}) \right). \tag{5.62}$$

Deductive value estimation can exploit this model of $K+1$ DBN in almost the same way as a single DBN. In Algorithm 9 on Page 103, the outer loop has to predict $\hat{\mathrm{P}}[\tilde{\boldsymbol{\psi}}] \approx \hat{\boldsymbol{\psi}} \in \mathcal{F}^{m' \times m_\pi}$ to calculate the equivalent of LSTD's matrix $\mathbf{D}$. For single DBN this multivariate function is computed by the compressed point-wise product of d LFF $\tilde{\mathbf{B}}^{\alpha\top} \hat{\boldsymbol{\varphi}}^{\alpha}, 1 \leq \alpha \leq d$. In the mixture-of-experts model, these d LFF are computed for each expert DBN P_k, yielding the compressed LFF $\hat{\boldsymbol{\psi}}_k \approx \hat{\mathrm{P}}_k[\tilde{\boldsymbol{\psi}}]$. The experts condition c_k is a LFF as well, and can therefore be applied by compressed point-wise multiplication, that is, $\hat{\boldsymbol{\psi}}'_k \approx c_k \cdot (\hat{\boldsymbol{\psi}}_k - \hat{\boldsymbol{\psi}}_0)$. Finally, the complete transition of $\tilde{\boldsymbol{\psi}}$ can be predicted by $\hat{\boldsymbol{\psi}} = \hat{\boldsymbol{\psi}}_0 + \sum_{k=1}^{K} \hat{\boldsymbol{\psi}}'_k \approx \hat{\mathrm{P}}[\tilde{\boldsymbol{\psi}}]$. Computational complexity is the same as for $K+1$ single DBN, that is, linear in the number of experts $\mathcal{O}(K)$, or quadratic in the number of Newtonian objects $\mathcal{O}(o^2)$.

I want to demonstrate the idea behind mixture-of-experts DBN at the example of SOKOBAN, depicted in Figure 5.13. Sokoban is a puzzle game by Hiroyuki Imabayashi, which has been published as a video game in 1982. The game is played in a grid-world and the state space contains all possible positions of the blue player

SOKOBAN

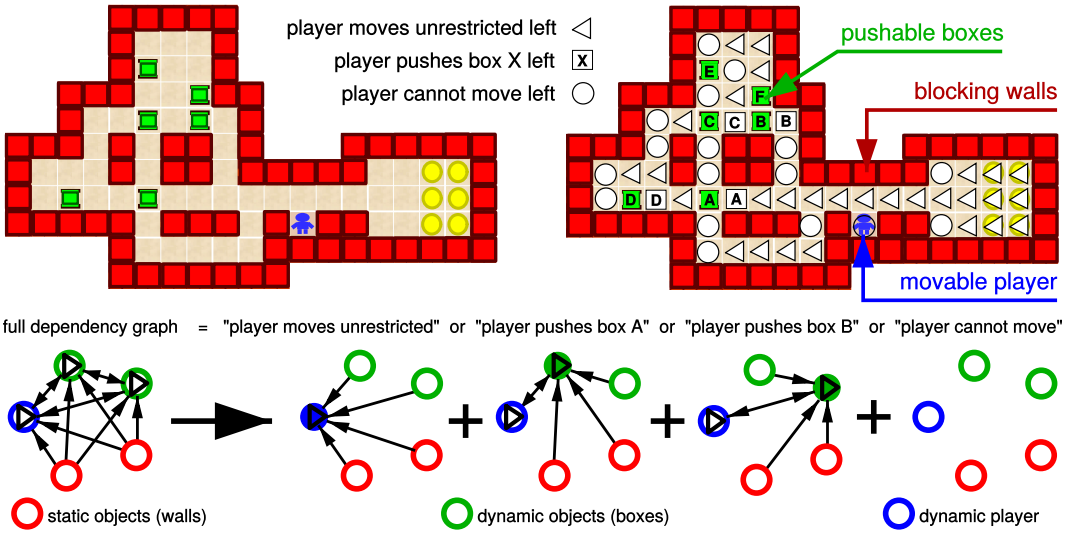


Figure 5.13: A SOKOBAN level (top left), in which the action "left" is executed (top right). The player (blue) can move freely (triangles) when it is not blocked (circles) by walls (red) or boxes (green). Pushing boxes is allowed when the field behind them is empty (one square for each box). Simplified transition dependencies are sketched in the bottom row. The original DBN is fully connected, but can be decomposed in sparse expert-DBN, which are only active under the above conditions.

and (a varying number of) green boxes. The player can be controlled by the four compass-actions and will move in the adjacent field, if (i) that field is not occupied by a red wall or a green box, or (ii) that field is occupied by a box, but the field behind the box is empty. In the second case, the box is pushed by one field as well. The top left of Figure 5.13 shows an example level with 6 Boxes. The goal of the game is to push all boxes on fields marked by yellow circles.

From the above dynamics, one can construct three types of experts: ($\bigcirc$) the "default" expert P_0 does not move anything, ($\triangleleft$) the "unrestricted" expert P_1 moves the agent one field in the compass direction indicated by action $\boldsymbol{a} \in \mathcal{A}$, and ($\square$) each box b has a "push" expert P_{b+1}, which moves both the box and the player by one field. The states in which the conditions of each expert are active for the "move left" action are shown on the top right of Figure 5.13. Note that the separation in experts also disentangled the dependency structure, sketched on the figures bottom. Each arrow shows a potential dependency of a variable (player or boxes) on a parent variable, which includes static walls that change only between levels. The original DBN (left) is almost completely connected, as each wall or box can get into the way of the player and/or the boxes. The dependencies of each expert, on the other hand, are much sparser. This indicates that their conditions c_k should correspond to compact LFF, which only have to consider few interactions between variables. In the next Section, I discuss at the example of Sokoban how environment-independent rules can be represented and how these rules can generate LFF to approximate conditions c_k.

5.4.2 Relational parameterization

GROUNDING

Generalization to arbitrary environments, for example, different Sokuban levels, requires to construct the experts and the conditionals of a given environment. This process is in other contexts often called SYMBOL GROUNDING (see Coradeschi et al., 2013, for an overview). In this section I assume a set of RELATIONAL RULES, that is, logical expressions based on predicates. Each predicate depends on one or more objects present in the environment, but the rules identify them merely by their CLASS. In Sokoban, for example, each object is either a player x, a box $b \in \mathcal{B}$ or a wall $w \in \mathcal{W}$. A rule composed of (predicates of) these classes may contain existence quantifiers $\exists(o_1 \in \mathcal{C}_1) \ldots \exists(o_n \in \mathcal{C}_n)$, which yield one *grounded* rule for each possible assignment of the variables $o_1, \ldots, o_n$ to objects in the environment with compatible classes. In relation reinforcement learning, this construction is called a DEICTIC RULE (Pasula et al., 2007; Lang and Toussaint, 2010).

DYNAMIC AND STATIC VARIABLES

The state space $\mathcal{X}$ is the collection all dynamic objects' variables. In Sokoban, $\mathcal{X}$ contains the positions of the player and all boxes. Note that one could increase the state space drastically by including the position of all walls. Although this probably renders the problem infeasible, the optimal policy would effectively solve all possible Sokoban levels at the same time. The designer has to decide[20] which objects are considered to be part of the state space, that is, whether they are DYNAMIC or STATIC. Furthermore, the objects' attributes are an important design decision. Sokoban has discrete positions, but all rules I define below can also be defined for continuous spaces. For example, Figure 5.14 on the next page shows that continuous

[20] It may be possible to frame this decision as a higher-order optimization problem by adding and removing objects. Note that this is a TASK-RELEVANT SUBSPACE (see Sections 2.1.4 and 4.1.1).

Sokoban rules without boxes can be used for navigation tasks. Knowing the classes of observed objects also allows more sample-efficient learning of "local" DBN P^α, as all samples from objects from the same class can be shared (see Section 5.5.3).

Modeling the RELATIONAL RULES of Sokoban requires a "collision" predicate $p(y, z, a) \in [0, 1]$, describing either the probability or the binary assumption that when action a is executed on object y, there will be a collision with object z. In the case of discrete Sokoban a neighborhood relationship suffices, that is, $p(y, z, \text{"left"}) = 1$ if and only if the object z is on the position left of object y. For continuous positions in stochastic worlds, the predicate should be probabilistic, though. The relational rules for Sokoban are (in the notation of Figure 5.13):

RELATIONAL RULES

$$\begin{aligned}
&\triangleleft\text{:}\ \exists x.\forall(k \in \mathcal{B} \cup \mathcal{W}).\neg p(x, k, a) && \rightarrow \text{move player } x \text{ only.}\\
&\square\text{:}\ \exists x.\exists(b \in \mathcal{B}).\big[p(x, b, a) \wedge \forall(k \in \mathcal{B} \cup \mathcal{W} \setminus \{b\}).\neg p(b, k, a)\big] && \rightarrow \text{move player } x \text{ and box } b.\\
&\bigcirc\text{:}\ \neg\triangleleft \ \wedge\ \neg\square && \rightarrow \text{movement blocked, no change.}
\end{aligned}$$

The corresponding transition models are trivially factored: the default DBN $P_\bigcirc$ does not change the state, player movement (in both $P^x_\triangleleft$ and $P^x_{\square}$) depends only on the player position and pushing box b in $P^b_{\square}$ depends only on b's position. The above conditions also resemble closely the "state context" of NOISY INDETERMINISTIC DEICTIC RULES[21]. NID rules can be learned inductively from interaction with the environment (Pasula et al., 2007) and are essential for relational planning with the PRADA algorithm (Lang and Toussaint, 2010). When the "context" of a rule holds, a discrete probability distribution selects an expert, which predicts the change in the relational state description. One "noisy" expert assumes a uniform transition to any state and represents the uncertainty of an imperfect model. Note the similarity to mixture-of-expert DBN: both have for each state a probability distribution over "experts", including a "default" expert when no other fits the state. One could see thus mixture-of-expert DBN as a *metric equivalent* of NID rules.

NID RULES

To estimate metric Q-values with mixture-of-expert DBN, relational rules have to be grounded and translated into conditions $c_k \in \mathcal{F}^m$. Given the predicates $p(\cdot, \cdot, a)$ for player x, all n_w walls $w_\alpha \in \mathcal{W}$ and all n_b boxes $b_\alpha \in \mathcal{B}$ as LFF, each Sokoban rule can be translated[22] into $n_w + n_b$ point-wise multiplications:

$$\begin{aligned}
c_\triangleleft(x, \boldsymbol{b}, \boldsymbol{w}, a) &= c\Big(\bigwedge_{y \in \mathcal{B} \cup \mathcal{W}} \neg p(x, y, a)\Big)\\
&= \prod_{\alpha=1}^{n_b} \big(1 - p(x, b_\alpha, a)\big) \cdot \prod_{\beta=1}^{n_w} \big(1 - p(x, w_\beta, a)\big)\,,\\
c_{\square}(x, \boldsymbol{b}, \boldsymbol{w}, a) &= c\Big(p(x, b_B, a) \bigwedge_{y \in \mathcal{B} \cup \mathcal{W} \setminus \{b_B\}} \neg p(b_B, y, a)\Big)\,, \qquad \forall B \in \{1, \ldots, n_b\}\\
&= p(x, b_B, a) \cdot \prod_{\alpha \neq B} \big(1 - p(b_B, b_\alpha, a)\big) \cdot \prod_{\beta=1}^{n_w} \big(1 - p(b_B, w, a)\big)\,.
\end{aligned} \tag{5.63}$$

In principle, one can therefore generate conditions from logical rules. However, it has yet to be seen how precise this can be achieved by point-wise LFF multiplication.

To demonstrate the concept, I used FAPI (Section 5.3.1) with a transition model $\mathbf{P}$, which I estimated from equidistant samples and continuous Sokoban rules without

[21] Pasula et al. (2007) called these rules originally NOISY DEICTIC RULES (NDR), but I use here the nomenclature from Lang and Toussaint (2010).

[22] I use here the transformations $c(\neg x) = 1 - c(x)$, $c(x \wedge y) = c(x)\,c(y)$ and $c(x \vee y) = c(x) + c(y) - c(x)\,c(y)$. The last transformation can be simplified to $c(x \vee y) = c(x) + c(y)$ when x and y are *disjunct*, that is, there are no state-action pairs for which both x and y holds.

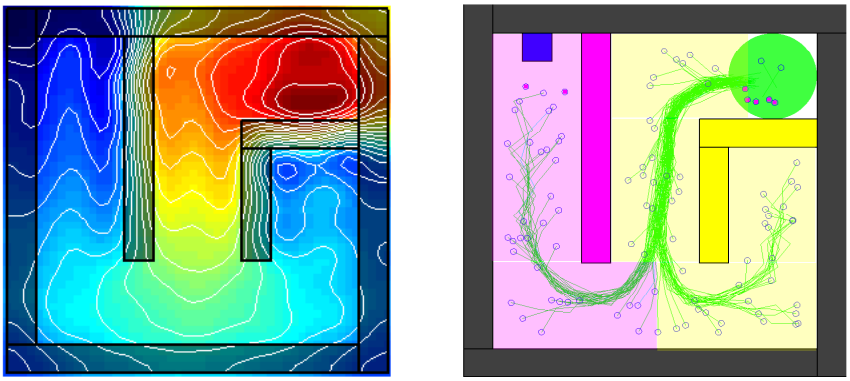


Figure 5.14: Estimated value function for a room with 7 walls (left), and test trajectories from the corresponding policy (right). Note that the policy can be approximated by two "relational actions": circle the magenta wall counter-clock-wise in the magenta area, and circle the yellow wall clock-wise in the yellow area.

boxes. Figure 5.14 shows on the left the learned value function in an environment with 7 walls and a goal area in the upper right corner. On the right, test-trajectories of the learned policy are plotted in green. A closer look onto these trajectories shows that the policy may also be expressible as RELATIONAL POLICY. These are decision trees or other sets of disjunct relational conditions $\{\bar{c}_k \in L^2(\mathcal{X}, \vartheta^x)\}_{k=1}^{\bar{K}}$. Analogous in mixture-of-expert DBN, these conditions select RELATIONAL ACTIONS, which are policies that usually depend only on the attributes of *few* objects. Note the similarity to *options* from hierarchical RL (see Section 2.3.1). In Figure 5.14, for example, the trajectories in the magenta area seem to circle the magenta wall (in some distance) counter-clock-wise. In the yellow area, on the other hand, the trajectories seem to circle the yellow wall clockwise. This could be interpreted as two out of four relational actions: two directions of movement for each wall. It is easy to see that these actions can solve navigation tasks in almost all imaginable mazes. Moreover, the relational actions are often *sparse*, that is, $\pi_k(d\boldsymbol{a}|\boldsymbol{x}) = \pi_k(d\boldsymbol{a}|\bar{\boldsymbol{s}}_k)$, $\bar{\boldsymbol{s}}_k \subset \boldsymbol{x} \in \mathcal{X}$, which allows to interpret a relational policy as a mixture-of-experts, too. However, as with relational rules, the number of grounded actions grows linear with the number of applicable objects, in this case walls.

RELATIONAL POLICIES

Given such a relational policy, for example, as a decision tree, the presented formalism may also be able to compute the corresponding metric Q-value. To apply relational actions to Algorithm 9, one only needs to replace the approximation of $\tilde{g} \approx \hat{\Gamma}_\pi[g]$. Similar to the transition model, one can interpret a relational policy as a mixture of expert relational actions, that is, sparse policies $\pi_k(d\boldsymbol{a}|\boldsymbol{x}) = \pi_k(d\boldsymbol{a}|\bar{\boldsymbol{s}}_k)$:

$$\begin{aligned} \hat{\Gamma}_\pi[g](\boldsymbol{x}) &= \hat{\Gamma}_{\pi_0}[g](\boldsymbol{x}) + \sum_{k=1}^{\bar{K}} \bar{c}_k(\boldsymbol{x}) \left(\hat{\Gamma}_{\pi_k}[g](\boldsymbol{x}) - \hat{\Gamma}_{\pi_0}[g](\boldsymbol{x})\right), \qquad \forall \boldsymbol{x} \in \mathcal{X}, \\ \hat{\Gamma}_{\pi_k}[g](\boldsymbol{x}) &= g^x(\boldsymbol{x}) \cdot \hat{\Gamma}_{\pi_k}[g^a](\bar{\boldsymbol{s}}_k), \quad \text{for } g(\boldsymbol{x}, \boldsymbol{a}) = g^x(\boldsymbol{x}) \cdot g^a(\boldsymbol{a}), \forall \boldsymbol{a} \in \mathcal{A}. \end{aligned} \tag{5.64}$$

Each policy π_k depends only on a small subset of variables $\bar{\boldsymbol{s}}_k$ from the participating objects, for example, the position and orientation of the agent and one specific wall. These policies can in principle be learned inductively as an LFF h^{π_k}, such that $\hat{\Gamma}_{\pi_k}[g^a] = \hat{\Gamma}_{\vartheta^a}[h^{\pi_k} \cdot g^a]$. RELATIONAL DECISION TREES yield also a natural order

DECISION TREES

for the point-wise multiplication of LFF, similar to DBN conditions in Equation 5.63. Here a node multiplies a given LFF to a predicate p_i and its negation $1 - p_i$, respectively. Each result is then passed to one of the nodes children. The conditions represented by the leafs of the tree are therefore always disjunct.

The above formalism allows in principle to perform planning in the abstraction of relational RL, but to evaluate the resulting relational policy in the underlying continuous or discrete state space $\mathcal{X}$. However, at the present time this is only conjecture, as I am not aware of any relational RL method that generates decision trees. Merging relational and metric reinforcement learning is a intriguing prospect, though, and calls for further study in future research.

5.5 Challenges to generalization

Many topics discussed in this chapter still await empirical verification and I expect some challenging surprises in the wake of these experiments. But there are still many unmentioned questions left, too. For example, it is still unclear how one can inductively learn a state- and action-space that *can* be modeled as a DBN, without running into the curse of insufficient samples (discussed in Section 4.5.3). In this section, I introduce some of the challenging problems and potential extrapolations, that had to be neglected in the rest of the chapter due to consistency and space.

5.5.1 Sparse regression with LFF

Inductive LFF algorithms like density estimation (Section 5.2.1), regression (Section 5.2.2) and value estimation (Section 5.2.3) have to regularize the cost functions $\hat{\mathrm{C}}[g]$ in the inner loop with virtual noise. Equation 5.32 on Page 88 demonstrates this for regression. Here one can separate the unregularized cost from the regularization:

$$\inf_{g\in\mathcal{F}} \underbrace{\|f+g-y\|_\zeta^2}_{\hat{\mathrm{C}}[g]} + \sum_{\alpha=1}^{d} \sigma_\alpha^2 \underbrace{\|\nabla_\alpha g + \nabla_\alpha f\|_\vartheta^2}_{\hat{\mathrm{R}}_\alpha[g]}, \qquad \text{s.t.} \quad \|g^\alpha\|_{\vartheta^\alpha} = 1\,, \quad \forall\alpha\,. \tag{5.65}$$

Note that there is one regularization term $\hat{\mathrm{R}}_\alpha[g]$ for each variable α, weighted by the virtual noise variance σ_α^2. In Section 5.2, these variances are hyper-parameters that have to be selected before training. As cross-validations over large dimensionalities d are very costly, the results reported there use the same parameter σ^2 for all input-variables. However, LFF are uniquely suited to represent SPARSE FUNCTIONS, that is, functions that depend on few input-dimensions, by setting all other factor functions to a constant 1. In practice, this can be achieved by increasing the parameter $\sigma_\alpha^2 \to \infty$ for unwanted "distractor" variables. In these input-dimensions, virtual samples dominate all differences between training samples, thereby enforcing a constant output. However, important variables must be known *before* training. It would therefore be of great help to select a sparse set of useful dimensions (low σ_α^2) during training. This is called SPARSE REGRESSION or AUTOMATIC RELEVANCE DETECTION (Li et al., 2002; Wipf and Nagarajan, 2008) and is usually performed using L_1 regularization (see an overview in Aravkin et al., 2014) or kernel selection (e.g. automatic relevance detection Rasmussen and Williams, 2006; Bishop, 2006).

SPARSE FUNCTIONS

One simple approach to select relevant dimensions during training is to penalize the Kullback-Leibler divergence D_{KL} of the regularization parameters w.r.t. another

prior distribution $\boldsymbol{\mu} \in \mathbb{R}^d_+, \sum_\alpha \mu_\alpha = 1$, for example the uniform distribution $\mu_\alpha = 1/d, \forall \alpha$. To translate variances $\mathbf{0} \preceq \boldsymbol{\sigma}^2 \prec \boldsymbol{\infty}$ into a probability distribution $\mathbf{0} \preceq \boldsymbol{\rho} \preceq \mathbf{1}$, I interpret ρ_α as the probability that variable $\mathcal{X}_\alpha$ is a "distractor", that is, not useful for estimating the target. Distractor variables are regularized with a large variance $\overline{\sigma}^2$, and useful variables with a low variance $\underline{\sigma}^2$. The actual regularization parameter σ_α^2 is thus the combination of both possibilities $\sigma_\alpha^2 := \rho_\alpha \overline{\sigma}^2 + (1-\rho_\alpha)\underline{\sigma}^2$. This yields the simultaneous optimization problem for basis g and parameters $\boldsymbol{\rho}$:

$$\begin{aligned} \inf_{g \in \mathcal{F},\, \boldsymbol{\rho} \in \mathbb{R}^d} \hat{\mathrm{C}}'[g, \boldsymbol{\rho}] \quad &:= \quad \hat{\mathrm{C}}[g] \;+\; \sum_{\alpha=1}^d \sigma_\alpha^2 \hat{\mathrm{R}}_\alpha[g] \;+\; \tfrac{1}{\eta} D_{\mathrm{KL}}(\boldsymbol{\rho}||\boldsymbol{\mu}) \\ \text{s.t.} \qquad \|g^\alpha\|_{\vartheta^\alpha} = 1\,, \quad \rho_\alpha \geq 0\,, \quad \forall \alpha\,, \qquad &\|\boldsymbol{\rho}\|_1 = 1\,. \end{aligned} \tag{5.66}$$

Optimizing the above cost function w.r.t. function g does not change the solutions derived in Section 5.2. Setting the derivative of the above cost function w.r.t. parameters $\boldsymbol{\rho}$ to zero, on the other hand, yields the closed form unconstrained solution $\boldsymbol{\rho}^{\mathrm{uc}}$, that is, using the shorthand $\bar{R}_\alpha := (\overline{\sigma}^2 - \underline{\sigma}^2)\,\hat{\mathrm{R}}_\alpha[g], \forall \alpha$:

$$\tfrac{\partial \hat{\mathrm{C}}'[g,\boldsymbol{\rho}]}{\partial \rho_\alpha} = \bar{R}_\alpha + \tfrac{1}{\eta}\big(\ln(\rho_\alpha) + 1 - \ln(\mu_\alpha)\big) \stackrel{!}{=} 0\,, \quad \Rightarrow \quad \rho_\alpha^{\mathrm{uc}} := \mu_\alpha \exp\left(-\eta \bar{R}_\alpha - 1\right). \tag{5.67}$$

The nonnegativity constraints are already fulfilled due to the exponential function, and $\|\boldsymbol{\rho}\|_1 = 1$ can be enforced by normalization:

$$\rho_\alpha \quad \stackrel{!}{=} \quad \frac{\rho_\alpha^{\mathrm{uc}}}{\sum_{\beta=1}^d \rho_\beta^{\mathrm{uc}}} \quad = \quad \frac{\mu_\alpha \exp(-\eta \bar{R}_\alpha)}{\sum_{\beta=1}^d \mu_\beta \exp(-\eta \bar{R}_\beta)}\,. \tag{5.68}$$

SOFTMIN Note that for the uniform prior $\mu_\alpha = \frac{1}{d}$, the solution for $\boldsymbol{\rho}$ is the SOFTMIN distribution (softmax with negative quantities) of the scaled regularization terms $\bar{\boldsymbol{R}}$. The inner loop of an inductive LFF algorithm can therefore optimize the next basis g and the regularization constants $\boldsymbol{\sigma}^2 := (\overline{\sigma}^2 - \underline{\sigma}^2)\boldsymbol{\rho} + \underline{\sigma}^2$ simultaneously.

Moreover, calculating $\bar{\boldsymbol{R}}$ can also help to adjust the INVERSE TEMPERATURE parameter η. The derivative of $D_{\mathrm{KL}}(\boldsymbol{\rho}||\boldsymbol{\mu})$ is complicated, but the negative gradient of the regularization term w.r.t. softmin parameter η can be calculated efficiently:

$$-\tfrac{\partial}{\partial \eta} \sum_{\alpha=1}^d \sigma_\alpha^2 \hat{\mathrm{R}}_\alpha[g] \quad = \quad \sum_{\alpha=1}^d \rho_\alpha \bar{R}_\alpha^2 - \Big(\sum_{\alpha=1}^d \rho_\alpha \bar{R}_\alpha\Big)^2. \tag{5.69}$$

The gradient of the cost function is therefore approximately the negative variance of the scaled regularization terms $\bar{\boldsymbol{R}} \in \mathbb{R}^d$ w.r.t. probability measure $\boldsymbol{\rho}$. One can thus perform a (small) gradient descend step each time $\boldsymbol{\rho}$ has been calculated to adjust η dynamically to the data. Practical experiments are very encouraging: η converged fast to some sensible level, which only changed when new bases were added to the estimated LFF.

The methods derived in this section work well to improve a given sparse (partial) solution, but do not solve the problem of sparse regression in general. This is because the order, in which the factor functions g^α are updated, has an unwanted effect on sparsity. Initially, all factor functions are constants, that is, all regularization terms $\bar{R}_\alpha = 0, \forall \alpha$. This changes when the first update of dimension α enforces $\|g^\alpha\|_{\vartheta^\alpha} = 1$. The new function g^α often induces $\bar{R}_\alpha > 0$ and minimizing the regularization parameters $\boldsymbol{\rho}$ yields a parameter ρ_α close to 0. The choice of almost no virtual sampling for the first updated dimension has a profound influence on the next update, and

the effect becomes more and more pronounced over time. Although I can not offer a solution yet, the potential for sparse regression is enormous. Future work in this direction could not only learn sparse DBN without any structural knowledge of the environment, but also address a vast variety of other research areas.

5.5.2 Sequential neural interpretation

Linear factored functions $f \in \mathcal{F}^m$ (see Section 5.1.2) are "shallow" neural networks (sketched in Figure 5.15): each input variable x_α has its own fist layer of m neurons $\psi_i^\alpha(x_\alpha) = \sum_{j=1}^{m_\alpha} B_{ji}^\alpha \, \phi_j^\alpha(x_\alpha)$, which are combined in a second layer of m multiplicative neurons $\psi_i(\boldsymbol{x}) = \prod_{\alpha=1}^{d} \psi_i^\alpha(x_\alpha)$. The last linear layer computes the output $f(\boldsymbol{x}) = \sum_{i=1}^{m} a_i \, \psi_i(\boldsymbol{x})$, which totals the entire network at $(d+1)m+1$ neurons. However, this architecture is biologically implausible as the number of objects, and thus associated variables, varies from situation to situation. An agent with a fixed neural architecture can simply not represent LFF with an arbitrary amount of variables. It also seems not very likely that so many neurons are dedicated to one function, given that a biological brain is highly multi-functional. Last but not least, all objects also have to be "perceived" at the same time. The narrow FOVEA CENTRALIS of human eyes does not allow simultaneous perception of multiple objects.

The neural network sketched in Figure 5.15 consists of d *parallel* sub-networks that compute the factor functions $\boldsymbol{\psi}^\alpha$. These differ only in their input and the weights $\mathbf{B}^\alpha \in \mathbb{R}^{m_\alpha \times m}$. Note that one can also interpret the linear weights in a connectionist neuron as inputs to multiplicative synapses. In this interpretation the parallel sub-networks can be computed sequentially using the same neuronal structure with "weights" $\mathbf{B}^\alpha$ set according to the currently observed variable x_α. Cycling between relevant objects/features is remarkably similar to SACCADES (rapid eye movements during vision, Buswell, 1935), observed in humans and many other animals (Land, 1999). Moreover, using parameters for weights allows to use the same neuronal structure to compute many different LFF. Figure 5.16 sketches such a recurrent neural network, which is calculating LFF f_i with only $2m+1$ neurons. The network needs to iterate in arbitrary order through all variables x_α of the environment. Variables that are not in the scope of f_i have constant factored basis functions $\boldsymbol{\psi}^\alpha = \mathbf{1}$ and do therefore not influence the computation. The architecture can be used for LFF with a maximum number of m basis functions, which allows to at least *approximate* any function. Note that the linear weights $\mathbf{B}^\alpha$and $\boldsymbol{a}$ are provided by a "database" of functions, which requires some kind of associative memory with an unknown amount of neurons.

SACCADES

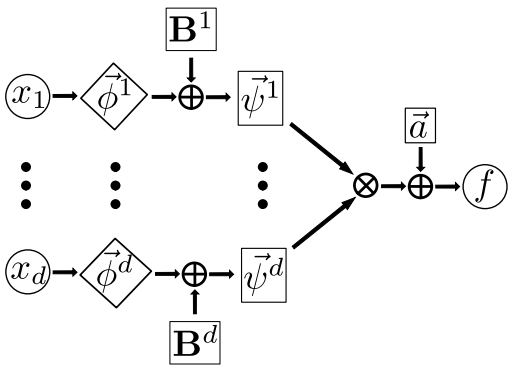


Figure 5.15: LFF $f \in \mathcal{F}^m$ as a neural network. $\oplus$-nodes indicate matrix multiplications (linear neurons), and $\otimes$-nodes point-wise multiplications (product synapses). Note that the path to each $\boldsymbol{\psi}^\alpha$ is structurally the same and varies only in the "weights" $\mathbf{B}^\alpha$. Figure 5.16 shows how this parallel structure can be translated into a compact and flexible neural network for sequential computation.

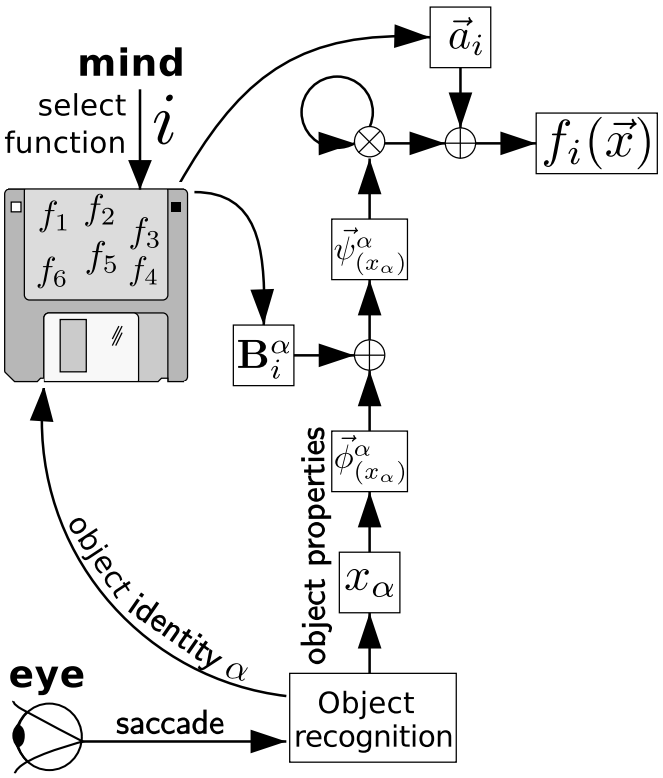


Figure 5.16: Neural architecture that computes LFF in environments with varying number of objects: the "eye" performs a "saccade" to an object and detects its identity α and properties x_α. A database of functions (associative memory) supplies the corresponding matrices $\mathbf{B}_i^\alpha$ and $\boldsymbol{a}_i$ for a function f_i, which may have been selected by another process (the "mind"). $\oplus$-nodes indicate matrix multiplications, and $\otimes$-nodes point-wise multiplications. Objects α outside of the scope of f_i have constant bases $\boldsymbol{\psi}^\alpha := \mathbf{1}$ and do therefore not influence the output. While traversing all required objects, the output converges to $f_i(\boldsymbol{x})$ of the current situation/state $\boldsymbol{x} \in \mathcal{X}$.

This architecture allows a resource-efficient neural implementation, which works in environments with an arbitrary number of variables d. However, the biological analogy also reveals some problems. For example, human saccades often "reexamine" previously visited spots. If one can not control precisely which variables have been observed yet, the network risks to process the same variable twice, which corrupts the output unrecoverably. Moreover, the OBJECT RECOGNITION system in Figure 5.16 also faces the though task to recognize objects/variables in the environment. Ensuring that the observed properties of an object always yield the same identifier α is a hard grounding problem. For example, two objects in the environment may look identical, but have different positions. To select the correct identifier for an observed property of one object, the recognition system would have to keep track of similar objects and their positions. This is too much for a simple feed-forward classification network, which can at most identify the objects' classes. A simplistic approach to object recognition may therefore *require* another, restricted class of LFF functions, which depend only on a variable's *class* and not on its object's *identity*.

OBJECT RECOGNITION

Even powerful object recognition systems may have a need for functions like that. In Section 5.4, Q-value function are generated from models, which are constructed using relational rules. This grounds all variables by assigning (essentially random) identifiers to the objects' properties. However, the learned functions can only be used once. As soon as the grounding is lost, models and value function have to be recomputed from the start. Using class identifiers instead of object identities yields a more general type of function, which can be applied to new situations. This may allow to construct some kind of SUBCONSCIOUSNESS, that is, a combination of all

SUBCONSCIOUS CONTROL

previously learned Q-value functions, that may be applicable to new situations. The "conscious" mind, that plans ahead using mental models, must only become active when the subconscious control fails or predicts failure. However, using variable classes instead of identities can not differentiate between "this" object and "that" object. It is not clear how to translate or integrate a Q-value function learned in a grounded environment into the subconscious Q-value function.

Repeating variables during the computation of LFF affects only the multiplicative neurons in Figure 5.16. To become invariant to such repetitions, one could replace the product with a maximum operation. However, I am not sure if one can give an equivalent to Corollary 5.1 with these functions and it is not possible to exploit structural knowledge in the same way (see Section 5.1). Future works may be able to exploit structure in L_∞ (see Section 2.2) to efficiently estimate Q-value functions, though. Given the computation costs of "conscious" planning and the benefits of a subconscious Q-value function, more research in this matter is certainly called for.

5.5.3 Learning mixture-of-expert DBN

Section 5.3.3 explains how to inductively learn DBN transition models from training data. Representing the DBN with d multivariate LFF $\hat{\boldsymbol{\varphi}}^\alpha$ allows to model arbitrary transition operators, but the inductive estimation (LFF regression in Section 5.2.2) requires the dependency structure of the DBN, that is, the parents of each variable must be known before training. Learning the structure of an unknown DBN from random-walks through the environment has been studied in literature (Degris et al., 2006; Chakraborty and Stone, 2011), but remains either heuristic or very expensive. However, there have recently been some breakthroughs in learning the structure of DEEP SUM-PRODUCT NETWORKS from data (Poon and Domingos, 2011; Gens and Domingos, 2013; Nath and Domingos, 2015). These model high-dimensional probability distributions for Bayesian inference as deep networks with sum- and product-nodes. The structure is very similar[23] to LFF and future work may extend the above techniques to learn "deep" DBN.

SUM-PRODUCT NETWORKS

However the DBN structure is learned, one can exploit another regularity in the environment to increase estimation quality of transition models: variables of the same class. In this chapter we assume that variables are properties of objects in the environment. Variables that denote the same property of the same class are drawn from the same transition model. Transition models P^α can therefore be estimated much more precisely by pooling the training data from all those variables together. Without knowledge of the class and object identities in the environment, one could try to classify the variables by CLUSTERING of their transitions[24] (for example with K-MEANS, see e.g. Bishop, 2006). This requires a similarity $k_{(\alpha,\beta)}$

CLUSTERING

[23] The computation $\hat{\mathrm{P}}[\bar{\psi}]$ (for DBN P estimated by LFF) in Equation 5.56 on Page 101 corresponds to a "flat" network, with one multiplicative node. Any "deep" network of point-wise addition and multiplication nodes can be computed in exactly the same way, though. These models are no longer *factorizing* and can therefore model correlated outcomes. This includes "simultaneously" active models, for example, Newtons first law with additional forces like gravity. One model describes the unaffected movement of an object and others the change induced by the additional forces. The models can be combined with a point-wise sum-node.

[24] Note that for a mixture-of-expert DBN the available transitions must also be *abducted* by the expert DBN most likely to explain them. This is a form of clustering, too.

between two variables' transitions, which could be calculated by measuring their correlation $c_{\alpha,\beta} = \frac{1}{n}\sum_{t=1}^{n}(x_{\alpha,t+1} - x_{\alpha,t})(x_{\beta,t+1} - x_{\beta,t})$ and defining $k_{(\alpha,\beta)} = \frac{c_{\alpha,\beta}}{\sqrt{c_{\alpha,\alpha}\, c_{\beta,\beta}}}$. It is easy to come up with non-Gaussian transition models that would fail this similarity measure, but correlation may nonetheless work in many applications.

Section 5.4.1 introduces the possibility to combine multiple expert DBN using LFF conditions and Section 5.4.2 describes how these conditions can be generated from relational rules. The above clustering solution should yield both *templates* for classes and a GROUNDING[25] of the corresponding objects. Given these classes, one must define predicates between them to construct relational rules. As with structure learning of DBN above, there exist attempts that LEARN PREDICATES to describe a relational state (Jetchev et al., 2013) or action space (Orthey et al., 2013). However, one of the authors[26] told me that in his experience one can only improve *either* state *or* action representation, and to my knowledge no completely autonomous approach to simultaneously learn both exists. A metric Q-value function, as derived in this chapter, may help to define or improve relational predicates, but future work is needed to verify this hypothesis.

PREDICATE LEARNING

Once a relational state (and action) description has been found, Pasula et al. (2007) have developed a greedy algorithm to learn NID-rules. The algorithm have since then been used in many publications (Lang and Toussaint, 2010; Lang et al., 2012; Höfer et al., 2014). An extension to learn relational rules for mixture-of-expert DBN should be straight forward. However, due to its greedy selection scheme, the algorithm by Pasula et al. (2007) can not learn higher order terms. For example, if $p_a(\boldsymbol{x}) \wedge p_b(\boldsymbol{x})$ describe the condition of a DBN perfectly, but neither $p_a(\boldsymbol{x})$ nor $p_b(\boldsymbol{x})$ alone raises the prediction quality, the algorithm can not learn the condition. To reliably learn higher order conditions, one would need to keep track of the BAYESIAN POSTERIOR over the set of all conditions. This is in general not feasible, due to the involved combinatorics. However, one could treat this as a SPARSE CODING PROBLEM (Olshausen and Field, 1996). Here one can use MCMC sampling to get the MAXIMUM-A-POSTERIORI estimate (MAP, Lee et al., 2007; Mairal et al., 2009), or attempt an approximation of the posterior (Seeger, 2008). Recent work has demonstrated that one can maintain most of the probability mass in sparse coding, while reducing the combinatorics to a minimum (Lücke and Eggert, 2010). The application of these techniques to learn relational rules appears promising and calls for further research.

SPARSE CODING

5.5.4 Deductive policy improvement

In Section 5.3.4 I derive a deductive Q-value estimation algorithm (Algorithm 9) that scales polynomially in the number of state-action dimensions $d + b$. To solve the complete control problem in polynomial time, one requires a policy improvement step that yields the PDF $h^{\pi_i} \in \mathcal{F}^{m'}$ of current policy π_i. As in FAPI (Section 5.3.1), policy improvement should maximize $\langle h^{\pi_i}, q^{\pi_{i-1}} \rangle_\vartheta$, where $q^{\pi_{i-1}} \in \mathcal{F}^m$ is the Q-value estimate of the previous policy π_{i-1}. However, this would require a point-wise maximum operation, which can only be computed in *exponential* time. My

[25] This does not allow to recognize a new (or reappearing old) object in the environment, (see Section 5.5.2). Generalization to new environments composed of objects from known classes would require an object recognition system, which must be based on additional sensor information.

[26] Marc Toussaint in a private conversation at the DFG PPR-1527 symposium in 2014.

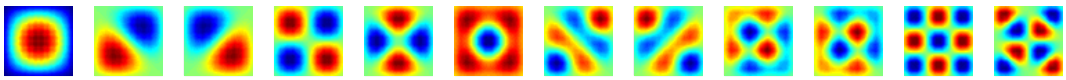


Figure 5.17: The action-invariant subspace $\mathbf{U}^\top \boldsymbol{\psi}'(\boldsymbol{x})$ of Gaussian bases $\boldsymbol{\psi}'$ (see Figure 5.2). Note the similarity to Fourier sine bases $\psi_{ij}(\boldsymbol{x}) = \sin(i\pi x_1) \cdot \sin(j\pi x_2)$.

autonomous deductive learning approach is therefore missing[27] a crucial component.

FAPI maximizes the policy with a linear program (LP, see Equation 5.50 on Page 97) to determine the linear coefficients $\boldsymbol{\varpi} \in \mathbb{R}^{m'}$ of a given set of basis functions $\{\psi'_k \in \mathcal{F}\}_{k=1}^{m'}$. These bases (i) must be able to express a near-greedy policy and (ii) must be non-negative. Selecting a universal non-negative factored function base, for example the Gaussian bases in Figure 5.2, requires a number of bases m' exponential in $d+b$. FAPI policy improvement is therefore not a viable approach for large state-action spaces. However, there are two modifications that are worth mentioning. First, one can reduce the number of non-negative basis functions by concentrating the variability of h^{π_i} on few "critical" states in $\mathcal{X}$. One could, for example, chose one basis that represent the current policy with "holes", and define for each hole a set of bases that allow to fill it for promising (or simply all possible) actions. How to select critical states and/or promising actions is an unresolved issue, though.

Secondly, one could try to loosen the requirement for non-negative bases and/or non-negative parameters $\boldsymbol{\varpi}$. In this context it is interesting to see how the LP changes by removing the equality constraint $\langle \hat{\Gamma}_{\vartheta^a}[\boldsymbol{\psi}'], \hat{\Gamma}_{\vartheta^a}[\boldsymbol{\psi}'^\top] \rangle_{\vartheta^x} = \langle \boldsymbol{\psi}', 1 \rangle_\vartheta$. Here one performs an eigenvalue decomposition[28] to project the solution h^π in an action-independent subspace $\mathbf{U}^\top \boldsymbol{\psi}'$ (with eigenvalues $\boldsymbol{\lambda}$) and an orthogonal policy-encoding subspace $\bar{\mathbf{U}}^\top \boldsymbol{\psi}'$ (with eigenvalues $\mathbf{0}$). By solving the optimization problem in both subspaces, that is, $\boldsymbol{\varpi} := \mathbf{U}\boldsymbol{w} + \bar{\mathbf{U}}\bar{\boldsymbol{w}}$, the linear equality constraint vanishes:

$$\begin{aligned} \sup_{\bar{\boldsymbol{w}} \in \mathbb{R}^{\bar{m}}} \quad & \bar{\boldsymbol{w}}^\top \langle \bar{\mathbf{U}}^\top \boldsymbol{\psi}', \boldsymbol{\psi} \rangle_\vartheta (\boldsymbol{r} + \gamma \mathbf{P}\boldsymbol{a}^*) \; + \; \overbrace{\boldsymbol{w}^\top \langle \mathbf{U}^\top \boldsymbol{\psi}', \boldsymbol{\psi} \rangle_\vartheta (\boldsymbol{r} + \gamma \mathbf{P}\boldsymbol{a}^*)}^{\text{constant w.r.t. } \bar{\boldsymbol{w}}}, \\ \text{s.t.} \quad & \bar{\mathbf{U}}\bar{\boldsymbol{w}} \succeq -\mathbf{U}\boldsymbol{w}, \qquad \boldsymbol{w} := \operatorname{diag}(\boldsymbol{\lambda})^{-1} \langle \mathbf{U}^\top \boldsymbol{\psi}', 1 \rangle_\vartheta \in \mathbb{R}^{m'-\bar{m}}. \end{aligned} \tag{5.70}$$

Figure 5.17 shows the action-invariant subspace $\mathbf{U}^\top \boldsymbol{\psi}'$ of Gaussian basis functions $\boldsymbol{\psi}'$ with two state and one action dimension. These are very similar to a Fourier sine basis, including the rotated versions of some bases. It appears therefore plausible to *first* define two orthogonal factored function spaces to solve the LP in, and *then* compute the rotations $\mathbf{U}$ and $\bar{\mathbf{U}}$ into an arbitrary non-negative function space. On the one hand, the inner products can be computed analytically and the increased freedom may allow a greedy optimization similar to LFF algorithms in this chapter. On the other hand, the inequality constraints still require one constraint for each equivalent nonnegative basis. Without restrictions on the representable policy, the number of constraints grows therefore exponential with the state-action dimension $d+b$. Although I am not familiar enough with INTERIOR POINT METHODS (Boyd

[27] In the context of combined deductive and inductive learning (Section 5.1.1) one could try a simplification: chose actions near the training samples according to a greedy policy and anywhere else with the uniform policy ϑ^a. Although this policy massively underestimates the value of unexplored state-actions, it should still yield an advantage over purely inductive Q-value estimation.

[28] The eigenvalue decomposition finds matrices $[\mathbf{U}, \bar{\mathbf{U}}] \operatorname{diag}([\boldsymbol{\lambda}, \mathbf{0}]) [\mathbf{U}, \bar{\mathbf{U}}]^\top := \langle \hat{\Gamma}_{\vartheta^a}[\boldsymbol{\psi}'], \hat{\Gamma}_{\vartheta^a}[\boldsymbol{\psi}'^\top] \rangle_{\vartheta^x}$. $\mathbf{U}$ and $\bar{\mathbf{U}}$ are unitary matrices, that is, $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$, $\bar{\mathbf{U}}^\top\bar{\mathbf{U}} = \mathbf{I}$, $\mathbf{U}^\top\bar{\mathbf{U}} = \mathbf{0}$, and $\boldsymbol{\lambda} > \mathbf{0} \in \mathbb{R}^{m'-\bar{m}}$.

and Vandenberghe, 2004) to say whether or not any regularity in the LP constraints can be exploited further, I have to conclude that policies in large state-action spaces cannot be improved using this method either.

Another potential way to perform policy improvement are relational policies, introduced in Section 5.4.2. Equation 5.64 on page 108 shows how the Q-value of relational actions can be computed from a given Q-value function. Relational policy improvement may therefore find grounded conditions for each action that maximize the Q-value estimated with Algorithm 9. These conditions can be translated into LFF and should allow to compute the improved policies PDF h^π. Success of this approach depends strongly on the available relational predicates and actions. However, it seems to be a long shot to rely on relational abstractions to solve an essentially metric maximization problem.

In summary, policy improvement in large state-action spaces remains an unsolved problem of utmost importance. Future research may follow lines presented in this section, but I am not too hopeful and expect the answer to lie somewhere else.

Chapter 6

Discussion

This thesis aims to find *representations that generalize continuous values in autonomous reinforcement learning*. First, I analyze the metric that generalizes values optimally for *inductive* learning and show that SFA learns a representation with this metric. Secondly, I investigate a representation that allows efficient *deductive* value estimation and define a new function class that can break the curse of dimensionality for autonomous control. In this chapter I summarize my results in Section 6.1, report my main conclusions in Section 6.2 and give a speculative outlook into both my own future work and the future of this field of research in Section 6.3.

6.1 Summary

I begin this thesis in Chapter 2 by introducing the foundation of AUTONOMOUS CONTROL, in particular the framework of REINFORCEMENT LEARNING (RL). RL can learn control policies autonomously, but computation is only feasible given the right representation of states and actions. The chapter discusses the various dimensions of PLANNING and LEARNING, but in this thesis I focus on the MDP framework (which excludes partial observability). In Chapter 3 I tackle *continuous* state and action spaces by introducing the concepts of MACHINE LEARNING and the notation of function analysis. I investigate in particular the generalization of values and the role that the METRIC of the state-action space plays for linear estimators used in *least-squares policy iteration* (LSPI). My main conclusions from Chapter 3 are:

AUTONOMOUS PLANNING AND LEARNING

- Linear functions (including kernel functions) are smooth w.r.t. the Euclidean metric in their feature spaces (see Lemmas 3.4 and 3.5 on Page 29), which constrains functions w.r.t. an induced metric $d(\cdot, \cdot)$ in the state(-action) space.
- Regularization schemes like weight decay in ridge regression and function norms in RKHS can be seen as approximations to VIRTUAL SAMPLING w.r.t. $d(\cdot, \cdot)$. I derive a novel regularization scheme based on a Taylor approximation of the regression cost in the limit of infinitely many virtual samples.
- LSPI is surprisingly *unstable* for large discount factors γ (see Figure 3.4 on Page 37). I identify greedy policy improvement as the most likely culprit and show how to stabilize LSPI using a softmax.

In Chapter 4 I ask what constitutes suitable REPRESENTATIONS for LSPI, and how one can learn them through interaction with the environment. I argue that all ISOMORPHIC STATE SPACES differ only in their metric $d(\cdot, \cdot)$. This metric generalizes the learned value function to unseen "nearby" states and actions by enforcing

REPRESENTATION

smoothness with constrains or regularization. Lemma 4.4 on Page 51 shows that value functions are always smooth w.r.t. a specific DIFFUSION METRIC $d^{\pi}_{\gamma}(\cdot,\cdot)$, and I prove with Lemma 4.5 and Proposition 4.6 that Euclidean distances in feature spaces learned by SLOW FEATURE ANALYSIS (SFA) approximate this metric. Moreover, I extend this unsupervised learning technique by minimizing a bound on all possible value functions which differ only in their reward (Proposition 4.11 on Page 56). In γ-SFA, the distribution over reward functions, and therefore the anticipated tasks, can be modeled by correlation kernels and the learned features reflect this choice (see Figure 4.6 for examples). To learn SFA representations in high-dimensional observation spaces, I developed a REGULARIZED SPARSE KERNEL SFA (RSK-SFA) algorithm with a novel sparse SUPPORT VECTOR SELECTION scheme in Section 4.3. In empirical evaluations, RSK-SFA requires less features and yields more robust LSPI policies than any other learned (in Figures 4.13 and 4.14) or predefined state representation (in Figure 4.3). This supports my theoretical claims that (γ-)SFA approximates a near-optimal representation for linear value estimation like in LSPI.

Irrespective of the representation, autonomous reinforcement learning faces a fundamental dilemma: value estimation in large state and action spaces suffers the CURSE OF INSUFFICIENT SAMPLES (and the curse of dimensionality in general). Here the number of training samples to reliably estimate the value function grows exponential with the dimensionality of the states and actions. GENERALIZATION techniques from machine learning developed in Chapter 4 do not scale well with the state dimensionality, and I propose instead DEDUCTIVE value estimation to overcome the curse. However, autonomous MODEL-BASED techniques must still (i) internally sample all state-actions for value estimation and (ii) estimate the transition model inductively from training samples. In Chapter 5 I define the novel class of LINEAR FACTORED FUNCTIONS (LFF) to address the first problem. LFF are in the limit equivalent to L^2 spaces of functions (over both continuous and finite input spaces, see Corollary 5.1), but also allow analytic inner products, marginalization and point-wise multiplication. Some of these operations can increase the number of bases, and I derived compression algorithms to address that. I also demonstrate inductive learning with LFF: density estimation, regression and value estimation. To establish that LFF are applicable to deductive value estimation, I derive the FACTORED APPROXIMATE PLANNING algorithm (FAPI in Section 5.3.1). However, the number of predefined basis functions required by the algorithm scales exponentially for a general basis, too. Therefore I develop a second deductive algorithm, which estimates the value directly in the space of LFF (Algorithm 9 on Page 103). The algorithm scales polynomial in the number of state and action dimensions, but requires a specific transition model represented by multivariate LFF. These models must be learned from interaction with the environment (see above) without running into the curse of insufficient samples themselves. Generalization on this level can only be tackled with STRUCTURAL ASSUMPTIONS on the given representation of states and actions. I first assume that the transitions follow a sparse DYNAMIC BAYESIAN NETWORK (DBN), which allows to estimate large transition models inductively with LFF regression (sketched in Figure 5.12). Realistic environments can rarely be described by DBN, and I derive a MIXTURE-OF-EXPERT DBN approach that can express more complex transition models. These can additionally be parameterized by RELATIONAL RULES, which allows to *generate* a transition model for every environment that is composed of known classes. Including this symbolic layer allows to generalize over an even larger class of autonomous control problems. The

GENERALIZATION

STRUCTURAL ASSUMPTIONS

only open questions are deductive POLICY IMPROVEMENT and relational STRUCTURE LEARNING.

To summarize, in this thesis I derive a framework that autonomously estimates value functions in large state and action spaces. My method breaks the curse of dimensionality by deductive value estimation with linear factored functions (LFF). These deductions are based on a transition model, which can be learned from realistic amounts of data with very few constraints on the representable distribution of successive states. A mixture-of-expert approach can additionally adjust the model to new environments by using relational conditions, that determine which "expert" model best predicts the transitions of specific states.

6.2 Conclusion

Most of my thoughts about representation leaning are summarized in Böhmer et al. (2015). In terms of theoretical analysis and realistic evaluation, I consider my work in Böhmer et al. (2013) to be the most advanced in the field. For linear value estimation, as in LSPI, the metric induced by the representation determines the generalization of values to unseen states and actions. I do therefore not expect future works to improve much over representations learned by (γ-)SFA (Böhmer et al., 2012). Here the only weak-points are the dependence on the sampling policy and the large number of features required to encode high dimensional state spaces (Böhmer et al., 2013). Non-linear value estimators, on the other hand, can compensate for imperfect metrics and do not require a functional basis over all states. Auto-encoder approaches yield thus more robust and compact spaces (Lange et al., 2012), which can be enhanced by additional objectives (Jonschkowski and Brock, 2014). Watter et al. (2015) have learned the, in my opinion, best representation for autonomous control so far. They train deep neural networks by optimizing predictions for a *stochastic locally optimal control* (SOC) planner. As in my work, the key insight is to use the definition of the autonomous control algorithm to optimize the representation.

However, all existing approaches to learn state representations are *inductive* estimators, that is, require samples from the entire state space (the curse of insufficient samples). Many of my colleagues assume that deep neural networks will eventually generalize learned representations to unseen regions of the state/observation space. I am not not convinced that is the case, though. It seems to me that the enormous success of deep networks in recent years relies primarily on convolutional layers and huge amounts of training samples. I argue that deep networks are very good to *transfer* important features of the data from one task to another, which reduces the amount of training samples to learn the new task. In highly combinatorial state spaces, these features must also be combinatorial and deep nets should therefore still fall prey to the curse of insufficient samples. I believe that the only remedy for this problem are *structural assumptions* (like conditional independence) that could be expressed in the neural networks' structure or as regularization in some layers. As an example, I discuss some sparse approaches to representation learning in Section 4.5.3. However the curse will be eventually broken, though, one needs to optimize the representation for some control algorithm that does not run into the same curse.

To exploit and transfer structure in state space, I focused in Chapter 5 on deductive value estimation under the assumption of conditional independent state transitions. Although most environments can only be modeled partially by DBN, this assumption allows varying degrees of deductive value estimation:

- describing *most* transitions as DBN and learn exceptions inductively,
- modeling regions with different transitions as mixture-of-expert DBN and
- generate a sitation-specific transition model from relational rules.

In difference to inductive learning, deductive estimation has to process entire functions f in single computational steps, that is, needs to compute $f' := \hat{\Gamma}_\pi[f]$ and $f'' := \hat{\mathrm{P}}[f']$ (Böhmer and Obermayer, 2013). For DBN P, these operators require marginalization, point-wise multiplication and addition between functions. Traditional non-linear function classes like kernel machines or neural networks miss efficient point-wise multiplication and marginalization operations. LFF, on the other hand, can perform these analytically (Böhmer and Obermayer, 2015). The possibilities discussed in this thesis do not exhausted all prospects for LFF, though. Instead of simply multiplying the predictions in a DBN, i.e. $\hat{\mathrm{P}}[\boldsymbol{\psi}] = \prod_{\alpha=1}^{d} \mathbf{B}^{\alpha\top} \hat{\mathrm{P}}^\alpha[\boldsymbol{\phi}^\alpha]$, deep networks of product, sum and marginalization nodes of arbitrary complexity are imaginable to express much more complex transition models than DBN (see Footnote 23 on Page 113). One can interpret mixture-of-expert DBN as "flat" instances of these networks. Here relational conditions allow generalization akin to symbolic deduction in classical AI, but for values in metric state-action spaces. As *grounded* relational rules are represented as LFF (i.e. conditions), such rules could also be integrated anywhere within a deep network. At this point in time, the largest drawback of LFF is the lack of efficient deductive policy improvement, which would require some[1] point-wise maximum operation. On the one hand, this may require to look in a very different direction, for example, to investigate ways to exploit functions from $L^\infty(\mathcal{X})$. On the other hand, policies exhibit often much more uniformity than value functions, which may eventually allow to find a suitable policy improvement in large state-action spaces by concentrating on few states where the policy switches.

In conclusion, the developed framework should allow unrivaled generalization abilities, but depends on a suitable state and action representations, whose transitions can be modeled by a small mixture-of-expert DBN. The representation closest to our needs has so far been learned by Watter et al. (2015). Including a factored DBN (or the above deep sum-product-marginalization network) in their auto-encoder framework should yield the required representation. Integrating a relational component would vastly improve the generalization abilities, but this also depends on a suitable set of relational predicates. Investigating how to learn (or improve) them autonomously appears to be another promising direction for future research.

[1] It does not need to be strictly a maximum operation. For example, with a point-wise inverse $f'(\cdot) \approx 1/f(\cdot)$, one can approximate a softmax that uses point-wise powers of the nonnegative Q-value f instead of the exponential function: $h^\pi \approx f^n \cdot (1/\hat{\Gamma}_{\vartheta^a}[f^n])$. Here $n \in \mathbb{N}$ denotes the number of point-wise multiplications of f, that is, the greediness of the resulting policy.

6.3 Outlook

Many claims in this thesis, in particular in Chapter 5, yet await empirical confirmation. To fully establish the presented ideas in the scientific community, I plan to perform the following experiments in the near future and to publish the results:

- Compare the predictions by the true sparse (non Gaussian) DBN P and the multivariate LFF $\{\hat{\boldsymbol{\varphi}}^\alpha\}_{\alpha=1}^d$, learned by LFF regression (Algorithm 8).
- Implement and evaluate deductive value estimation (Algorithm 9) with these models. Compare the results with inductive value estimation (Section 5.2.3).
- Optimize inductive and deductive cost simultaneously, as sketched in Section 5.1.1. Compare performance in the task of Figure 5.12 with LSTD and Alg. 9.
- Extend Algorithm 9 by mixture-of-experts DBN and compare with the original.
- Large scale evaluation of a simulated robotic arm. Here the position of each link depends only on its joint and the previous link's position, which is a DBN.

This thesis also discusses many unresolved challenges to autonomous control. Reaching its end, I want to summarize the, in my eyes, most important problems that have to be addressed by the community in order to achieve truly autonomous control:

- The most urgent challenge in the context of this thesis is to derive a deductive policy improvement scheme (see Section 5.5.4). This is not only necessary for the framework developed here, but also essential for *any* deductive planning in large state-action spaces, which are the norm in most areas of application. Note that this is related to finding task-relevant subspaces (see Section 4.5.3).
- Secondly, the community must embrace more realistic structural assumptions for generalization. This thesis attempts this with a mixture-of-expert DBN, but as I have discussed in Section 5.5.3, "deep" networks of nodes that compute point-wise products, matrix multiplications and/or marginalization may be able to model real environments much more compact and detailed.
- Directly related is the challenge of generalization in inductive representation learning. I have shown in Chapter 4 how representations can be learned, but generalization to unseen regions in combinatorial state spaces (Section 4.5.3) requires structural assumptions, too. It is still unclear which assumptions generalize well for sensor observations like camera images, though.
- Another essential stepping stone to move forward is the integration of *symbolic* relational representation and *metric* transition models in continuous state spaces. I address this in Sections 3.4.2 and 5.3.3. The challenge is to learn relational conditions, predicates and actions from metric observations.
- Autonomous control is often faced with combinatorial state spaces consisting of the attributes of objects. In Section 5.5.2, I propose a sequential neural architecture that computes LFF in face of varying *amounts* of objects. Creating architectures, that are independent of the symbol *grounding*, could reuse prior experiences furthermore by generalizing values to entirely new situations.

- Realistic observation rarely contain information on the entire state of the environment (see POMDP in Section 2.1.4). Together with the vast majority of literature, this thesis explicitly ignores partial observability to learn representations. When observation spaces are no longer *isomorphic* to the underlying state spaces (see Section 4.1.3), the representation must include short term memory using predictive state representations (PSR, Littman et al., 2001) or recurrent neural networks (e.g. LSTM, Hochreiter and Schmidhuber, 1997). However, another important challenge is to integrate the quickly changing policies/intentions of other agents into the representation (see Section 2.3.2).

- The last challenge I want to mention is an almost philosophical one: true autonomy is shaped by internal motivation, not external reward. I discuss in Section 2.3.3 how *imitation* may generate intrinsic reward to learn policies that maximize long-term extrinsic rewards. However, there is to my knowledge no approach that decides autonomously *which* behavior is worth imitating. In my eyes, the greates challenge to autonomous control is to construct a "playful" agent, that determines autonomously which short term intrinsic reward will minimize long-term extrinsic punishments, like hunger, pain and death.

In conclusion, working on autonomous control for as long as I did, has given me an appreciation for the framework of reinforcement learning (RL). It allows to theoretically study control problems with very few restrictions, and can be cast into flexible machine learning algorithms (ML). Nonetheless, this thesis is my attempt to question some of the assumptions in both RL and ML, like the curse of insufficient samples, that hold practical applications back but are almost never discussed. The framework I developed may or may not have an influence on the field, but I belief that we are at a point in time where the *quality* of autonomous control research is about to change. I am very confident that the issues I addressed in this thesis, and many mentioned above that go far beyond it, will become a major focus of research in the coming decades.

It is truly a great time to do science.

Appendix A

State Representation Publications

A.1 Böhmer et al., ML 89:67–86 (2012)

The article *Generating feature spaces for linear algorithms with regularized sparse kernel slow feature analysis* (Böhmer et al., 2012) has been published in the **Machine Learning** (ML) journal[1] on the 13th of June 2012. It is an extended version of the conference paper *Regularized sparse kernel slow feature analysis* (Böhmer et al., 2011), published in the proceedings of the *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databased 2011* in Athens, Greece.

The article derives a *regularized sparse kernel slow feature analysis* (RSK-SFA) algorithm to construct non-linear bases for subsequent applications of linear algorithms. Sparsity is achieved by selecting a representative subset of the training-data with the newly developed MP-MAH algorithm. The extracted features are evaluated at the example of a classification task that predicts vowels in spoken words. Results and algorithm are summarized in Section 4.3. The notation follows mostly Chapter 4, except for the training samples, which are named in line with standard literature:

- The training set is denoted $\{\boldsymbol{x}_t\}_{t=1}^n \subset \mathcal{X}$ instead of $\{\boldsymbol{z}_t\}_{t=1}^n \subset \mathcal{Z}$.
- The sparse support vector sets are denoted $\{\boldsymbol{z}_i\}_{i=1}^m \subset \mathcal{X}$ instead of $\{\boldsymbol{s}_k\}_{k=1}^m \subset \mathcal{Z}$.

The authors contributed in the following way:

- *Wendelin Böhmer* has written the entire article, has developed both the RSK-SFA and the MP-MAH algorithm, and has designed and performed the experiments.
- *Steffen Grünewälder* has contributed to the development of the RSK-SFA algorithm and has given comments to the article.
- *Hannes Nickisch* has developed a precursor algorithm to RSK-SFA without regularization.
- *Klaus Obermayer* supervised both development and experimentation, and gave comments to the article.

[1] The final publication is available at Springer via `http://dx.doi.org/10.1007/s10994-012-5300-0` .

Generating Feature Spaces for Linear Algorithms with Regularized Sparse Kernel Slow Feature Analysis

Wendelin Böhmer · Steffen Grünewälder · Hannes Nickisch · Klaus Obermayer

Received: 31 October 2011 / Accepted: 22 May 2012

Abstract Without non-linear basis functions many problems can not be solved by linear algorithms. This article proposes a method to automatically construct such basis functions with *slow feature analysis* (SFA). Non-linear optimization of this unsupervised learning method generates an orthogonal basis on the unknown latent space for a given time series. In contrast to methods like PCA, SFA is thus well suited for techniques that make direct use of the latent space. Real-world time series can be complex, and current SFA algorithms are either not powerful enough or tend to over-fit. We make use of the *kernel trick* in combination with *sparsification* to develop a kernelized SFA algorithm which provides a powerful function class for large data sets. Sparsity is achieved by a novel *matching pursuit* approach that can be applied to other tasks as well. For small data sets, however, the kernel SFA approach leads to over-fitting and numerical instabilities. To enforce a stable solution, we introduce *regularization* to the SFA objective. We hypothesize that *our algorithm generates a feature space that resembles a Fourier basis in the unknown space of latent variables underlying a given real-world time series.* We evaluate this hypothesis at the example of a *vowel classification* task in comparison to *sparse kernel PCA*. Our results show excellent classification accuracy and demonstrate the superiority of kernel SFA over kernel PCA in encoding latent variables.

Keywords Time Series, Latent Variables, Unsupervised Learning, Slow Feature Analysis, Sparse Kernel Methods, Linear Classification

Wendelin Böhmer · Klaus Obermayer
Neural Information Processing Group, Technische Universität Berlin, Germany.
E-mail: wendelin@ni.tu-berlin.de, klaus.obermayer@tu-berlin.de

Steffen Grünewälder
Centre for Computational Statistics and Machine Learning, University College London, United Kingdom.
E-mail: steffen@cs.ucl.ac.uk

Hannes Nickisch
Philips Research Laboratories, Hamburg, Germany. E-mail: hannes.nickisch@philips.com

1 Introduction

This article is concerned with the automatic construction of non-linear basis functions for linear algorithms. This is of particular importance if the original space of inputs $\mathcal{X} \subseteq \mathbb{R}^l$ can not support an adequate linear solution.

New algorithms are often initially formulated using a linear function class $\mathcal{F}_\mathcal{X} := \{f(\boldsymbol{x}) = \sum_{i=1}^{l} w_i\, x_i \,|\, \boldsymbol{w} \in \mathbb{R}^l\}$ of possible solutions $f : \mathcal{X} \to \mathbb{R}$. Examples include *linear regression* and the discrimination function for *linear classification*. We adopt the point of view that the desired solutions are actually defined on the domain of an unknown low dimensional space of latent variables Θ, which is embedded in the high dimensional space of sensor observations $\boldsymbol{x} \in \mathcal{X}$. The restriction to linear functions $f \in \mathcal{F}_\mathcal{X}$, however, can prevent a suitable solution because (a) Θ might be non-linearly embedded in $\mathcal{X}$ and (b) the true solution f^* might be non-linear in Θ. This can be compensated by the introduction of a feature space $\Phi := \{\boldsymbol{\phi}(\boldsymbol{x}) \,|\, \boldsymbol{x} \in \mathcal{X}\}$ with mappings $\phi_i : \mathcal{X} \to \mathbb{R}, \forall i \in \{1, \dots, p\}$, such that:

- $\phi_i(\cdot), \forall i \in \{1, \dots, p\}$, is non-linear in $\mathcal{X}$ to encode Θ rather than $\mathcal{X}$.
- $\{\phi_i\}_{i=1}^{p}$ constitutes a well behaving functional basis in Θ, e.g. a Fourier basis.
- Dimensionality p is as low as possible to reduce the number of training samples required to reliably estimate $f \in \mathcal{F}_\Phi := \{f(\boldsymbol{x}) = \sum_{i=1}^{p} w_i\, \phi_i(\boldsymbol{x}) \,|\, \boldsymbol{w} \in \mathbb{R}^p\}$.

This article outlines an approach to automatically construct such a feature space Φ for a given time series. The critical assumption is that there exist an unknown low dimensional space of latent variables Θ, that generated the observed data $\boldsymbol{x} \in \mathcal{X}$ by some unknown stochastic process $\Theta \to \mathcal{X}$. It is important to understand that we do not talk about the true underlying *cause* of the data, but a description Θ that suffices to generate the training samples $\{\boldsymbol{x}_t\}_{t=1}^{n} \subset \mathcal{X}$. Additionally, we restrict ourselves here to the most general encoding of Θ, i.e. we assume no additional information about labels or classes of the training samples[1]. This is achieved by an unsupervised learning principle called *slow feature analysis* (SFA, Wiskott and Sejnowski, 2002). SFA aims for temporally coherent features out of high dimensional and/or delayed sensor measurements. Given an infinite time series and an unrestricted function class, the learned features will be a Fourier basis[2] in Θ (Wiskott, 2003; Franzius et al., 2007). Although there have been numerous studies highlighting its resemblance to biological sensor processing (Wiskott and Sejnowski, 2002; Berkes and Wiskott, 2005; Franzius et al., 2007), the method has not yet found its way in the engineering community that focuses on the same problems. One of the reasons is undoubtedly the lack of an easily operated non-linear extension that is powerful enough to generate a Fourier basis.

The major contribution of this article is to provide such an extension in the form of a kernel SFA algorithm and to demonstrate its application at the example of linear classification. An approach on kernelizing SFA has previously been made by Bray and Martinez (2002) and is reported to work well with a large image data set. Small training sets, however, lead to numerical instabilities in any kernel SFA algorithm. Our goal is to provide an algorithm that can be applied to both of the cases above and to document the pitfalls that arise in its application on real-world time series. This algorithm has previously been presented in a conference paper, which has been the basis for this article (Böhmer et al., 2011). Beyond the scope of the original paper, we provide evidence for its main hypothesis and evaluate

[1] Extensions that include such information are discussed in Section 6.

[2] This holds only for *factorizing spaces* Θ, i.e. independent boundary conditions for each dimension. However, more general statements can be derived for any *ergodic* Markov chain.

linear classification algorithms in the feature spaces generated by kernel SFA and kernel PCA (Schölkopf et al., 1997).

Although formulated as a linear algorithm, SFA was originally intended to be applied on the space of polynomials, e.g. quadratic (Wiskott and Sejnowski, 2002) or cubic (Berkes and Wiskott, 2005). The polynomial expansion of potentially high dimensional data, however, spans an impractically large space of coefficients. Hierarchical application of quadratic SFA has been proposed to solve this problem (Wiskott and Sejnowski, 2002). Although proven to work in complex tasks (Franzius et al., 2007), this approach involves a multitude of hyper-parameters and no easy way to counteract inevitable over-fitting is known so far. It appears biologically plausible, but is definitely not easy to operate.

A powerful alternative to polynomial expansions are *kernel methods*. Here the considered feature maps $\phi : \mathcal{X} \to \mathbb{R}$ are elements of a *reproducing kernel Hilbert space* $\mathcal{H}$. For many optimization problems a *representer theorem* holds in line with Wahba (1990), if a regularization term is used. Such theorems guarantee that the optimal solution for a given training set exists within the span of *kernel functions*, which are parametrized by training samples. Depending on the kernel, continuous functions can be approximated arbitrarily well in $\mathcal{H}$ and a mapping $\phi \in \mathcal{H}$ is thus very powerful (Shawe-Taylor and Cristianini, 2004).

There are, however, fundamental drawbacks in a kernel approach to SFA. First, choosing feature mappings from a powerful Hilbert space is naturally prone to over-fitting. More to the point, kernel SFA shows *numerical instabilities* due to it's unit variance constraint (see Sections 3 and 5). This tendency has been analytically shown for the related *kernel canonical correlation analysis* (Fukumizu et al., 2007). We introduce a regularization term for the SFA objective to enforce a stable solution. Secondly, kernel SFA is based on a *kernel matrix* of size $\mathcal{O}(n^2)$, where n is the number of training samples. This is not feasible for large training sets. Our approach approximates the optimal solution by projecting into a sparse subset of the data. The choice of this subset, however, is a crucial decision.

The question how *many* samples should be selected can only be answered empirically. We compare two state-of-the-art sparse subset selection algorithms that approach this problem very differently: (1) A fast *online algorithm* (Csató and Opper, 2002) that must recompute the whole solution to change the subset's size. (2) A computational costly *matching pursuit approach* to sparse kernel PCA (Smola and Schölkopf, 2000) that incrementally augments the selected subset. To obtain a method that is *both* fast and incremental we derive a novel matching pursuit approach to the first algorithm.

Bray and Martinez (2002) have previously introduced a *kernel SFA* algorithm that incorporates a simplistic sparsity scheme. Instead of the well-established framework of Wiskott and Sejnowski (2002), they utilize the cost function of Stone (2001) based on long and short term variances without explicit constraints. Due to a high level of sparsity, their approach does not require function regularization. We show that the same holds for our algorithm if the sparse subset is only a small fraction of the training data. However, for larger fractions additional regularization becomes inevitable.

Theoretic predictions by Wiskott (2003) and Franzius et al. (2007) suggests that *regularized sparse kernel SFA features will resemble a Fourier basis in the space of latent variables Θ for a given real-world time series*. To verify this hypothesis, we perform a *vowel classification task* on spoken words. None of the tested *linear classification* algorithms (Bishop, 2006) were able to solve the task without non-linear basis functions. Our work extends Berkes (2005), who used non-linear SFA to classify images of hand written digits. His work was based on artificial time series rather that real-world data, though. Our results show excellent classification accuracy of more than 97% and superior encoding of latent variables by *kernel SFA* in comparison with

kernel PCA (Schölkopf et al., 1997). To the best of our knowledge, this work and its predecessor (Böhmer et al., 2011) are the first attempt to apply non-linear SFA as a pre-processing for audio data.

In the following section, we first review a variety of linear classification algorithms, which we will use to compare the constructed feature spaces Φ. In Section 3 we formulate the general SFA optimization problem and derive a *regularized sparse kernel SFA algorithm*. In Section 4 the sparse subset selection is introduced and a novel matching pursuit algorithm derived. Section 5 evaluates both the SFA algorithm and the generated feature space on a vowel classification task, followed by a discussion of our method and possible extensions in Section 6.

2 Linear Classification Algorithms

Classification aims to assign to each test sample $\boldsymbol{x}^* \in \mathcal{X}$ a class $c^* \in \mathcal{C}$, based on a given training set of samples $\{\boldsymbol{x}_t\}_{t=1}^n$ and their assigned classes $\{c_t\}_{t=1}^n$. In the case of two classes $\mathcal{C} = \{-1, +1\}$, *linear classification* aims for a *discrimination function* $f \in \mathcal{F}_\Phi = \{f(\boldsymbol{x}) = \sum_{i=1}^p w_i\, \phi_i(\boldsymbol{x}) \,|\, \boldsymbol{w} \in \mathbb{R}^p\}$ with $\operatorname{sign}\big(f(\boldsymbol{x}_t)\big) = c_t, \forall t \in \{1, \dots, n\}$. All of the below algorithms are excellently discussed by Bishop (2006).

2.1 Least Squares Classification / LDA / FDA

The most straight forward approach to classification is to encode each class label with a 1-of-$|\mathcal{C}|$ binary label vector[3] and to perform *least-squares* (LS) regression of $\boldsymbol{g} \in \mathcal{F}_\Phi^{|\mathcal{C}|}$ against this multivariate target (Bishop, 2006). The function $g_i \in \mathcal{F}_\Phi$ with the highest output on a test sample yields the corresponding class. For example, in the two-class case $\mathcal{C} = \{-1, +1\}$, the *discrimination function* $f \in \mathcal{F}_\Phi$ that separates the classes is given by $f(\boldsymbol{x}) = g_2(\boldsymbol{x}) - g_1(\boldsymbol{x})$. The LS solution is found by minimizing

$$\min_{\boldsymbol{g} \in \mathcal{F}_\Phi^{|\mathcal{C}|}} \sum_{i=1}^{|\mathcal{C}|} \sum_{t=1}^{n} \Big(g_i(\boldsymbol{x}_t) - T_{it}\Big)^2 . \tag{1}$$

As each linear function $g_i \in \mathcal{F}_\Phi$ is characterized by a vector $\boldsymbol{w}_i \in \mathbb{R}^p$, the above cost function has the unique[4] analytical solution

$$\boldsymbol{w}_i = \mathbf{C}^{-1} \boldsymbol{\mu}_i\,, \quad \text{where } C_{ij} := \frac{1}{n} \sum_{t=1}^{n} \phi_i(\boldsymbol{x}_t)\, \phi_j(\boldsymbol{x}_t) \text{ and } \boldsymbol{\mu}_i := \frac{1}{n} \sum_{t=1}^{n} T_{it}\, \boldsymbol{\phi}(\boldsymbol{x}_t)\,. \tag{2}$$

The terms $\boldsymbol{\mu}_i \in \mathbb{R}^p$ and $\mathbf{C} \in \mathbb{R}^{p \times p}$ are the empirical estimates of a weighted ith class mean and the second moment correlation matrix in feature space Φ. Notice that in *zero mean* and *unit variance* feature spaces, which we will compare in Section 5, the computation reduces to the estimation of the class means $\boldsymbol{\mu}_i$. In general, however, the algorithm has a computational complexity of $\mathcal{O}(p^2 n + p^3)$.

[3] Here the ith class label is a $|\mathcal{C}|$ dimensional vector which is zero everywhere except for the ith entry, which is one. The result is a matrix $\mathbf{T}$ with $T_{it} = 1$ if $c_t = i$ and 0 otherwise.

[4] The solution is unique if $\mathbf{C}$ is invertible. If not, a *Moore-Penrose pseudoinverse* of $\mathbf{C}$ can still yield acceptable results.

If Φ has zero mean w.r.t. all samples and the classes are equally distributed, $\mathbf{C}$ is the estimate of the covariance matrix and the solution is equivalent to a Gaussian estimate of the *class probability densities* with the restriction that all classes share the same covariance (Bishop, 2006). This approach to derive the discrimination function out of Gaussian density estimates is also called *linear discrimination analysis* (LDA). With small modifications to the target weights (Duda and Hart, 1973), the solution of Equation 2 becomes also equivalent to *Fisher discrimination analysis* (FDA, Fisher, 1936). As the empirical differences in our experiments were marginal, we restricted ourselves in Section 5 to LDA. Note, however, that due to the strong assumption of equal covariances, these algorithms are known to perform poorly if this assumption is not met.

2.2 Perceptron Classification

The algorithm that started the field of *artificial neural networks* (Rosenblatt, 1962) can classify two-class problems with $\mathcal{C} = \{-1, +1\}$ by assigning a signum function onto the output of the discrimination function $f \in \mathcal{F}_\Phi$, i.e. $y(\boldsymbol{x}_t) = \operatorname{sign}\big(f(\boldsymbol{x}_t)\big)$. At sample $\boldsymbol{x}_t$, the online version of this algorithm updates the weight vector $\boldsymbol{w} \in \mathbb{R}^p$ of $f(\boldsymbol{x}_t) = \sum_{i=1}^p w_i\, \phi_i(\boldsymbol{x}_t)$ by $\Delta w_i^{(t)} := -\,\phi_i(\boldsymbol{x}_t)\, c_t$, but *only* if the prediction $y(\boldsymbol{x}_t)$ was incorrect. The update $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha\, \Delta \boldsymbol{w}^{(t)}$ with learning rate $0 < \alpha \leq 1$ is repeated until all training samples are classified correctly. The algorithm is guaranteed to converge for linear separable problems (Rosenblatt, 1962).

We do not expect linearly separable problems and thus devised a batch algorithm with a decaying learning rate, i.e. $\alpha_i := 1/i$ at iteration i until convergence.

$$w_j^{(i+1)} := w_j^{(i)} - \frac{\alpha_i}{\sum_{t=1}^n d_t^{(i)}} \sum_{t=1}^n d_t^{(i)} c_t\, \phi_j(\boldsymbol{x}_t)\,, \quad \text{where } d_t^{(i)} := \begin{cases} 1\ , \text{if } y(\boldsymbol{x}_t) \neq c_t \\ 0\ , \text{otherwise} \end{cases} \tag{3}$$

Convergence criteria that stay comparable over multiple feature space sizes are hard to formulate and so we ran the algorithm for 100 iterations, which appeared to be sufficient for convergence in all feature spaces Φ. The computational complexity for i iterations of the above algorithm is $\mathcal{O}(pin)$.

2.3 Logistic Regression

Rather than a *signum* in the perceptron algorithm, the *logistic regression* algorithm applies the *logistic sigmoid* function $\sigma(a) := (1 + exp(-a))^{-1}$ onto the discrimination function $f \in \mathcal{F}_\Phi$ to approximate the probability $P(c_t = 1 \,|\, \boldsymbol{x}_t) \approx y(\boldsymbol{x}_t) := \sigma(f(\boldsymbol{x}_t))$. By encoding the classes $\mathcal{C} := \{0, 1\}$, one can express the likelihood function in the two-classes case (Bishop, 2006) for a given discriminant function $f(\cdot)$ as

$$p(c_1, \ldots, c_n \,|\, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \prod_{t=1}^n \Big(y(\boldsymbol{x}_t)\Big)^{c_t} \Big(1 - y(\boldsymbol{x}_t)\Big)^{1-c_t}\,. \tag{4}$$

This likelihood can be maximized by the *Newton-Raphson scheme* (Fletcher, 1987). The method requires the calculation of gradient and Hessian of the log-likelihood. As the sigmoid $\sigma(\cdot)$ is nonlinear, the Hessian $\mathbf{H}^{(i)}$ is no longer independent of the discrimination function $f^{(i)} \in \mathcal{F}_\Phi$

and has to be recomputed every iteration i. The resulting algorithm is called *iterative reweighted least squares* (Rubin, 1983):

$$\begin{aligned} w_j^{(i+1)} &:= w_j^{(i)} - \sum_{k=1}^{p}\sum_{t=1}^{n} (\mathbf{H}^{(i)})^{-1}_{jk}\ \phi_k(\boldsymbol{x}_t)\ \left(y^{(i)}(\boldsymbol{x}_t) - c_t\right), \\ \text{where}\quad H_{jk}^{(i)} &= \sum_{t=1}^{n} \phi_j(\boldsymbol{x}_t)\ y^{(i)}(\boldsymbol{x}_t)\ \left(1 - y^{(i)}(\boldsymbol{x}_t)\right)\ \phi_k(\boldsymbol{x}_t)\,. \end{aligned} \tag{5}$$

The algorithm showed clear signs of over-fitting in large feature spaces. Regularization[5] of the Hessian $\mathbf{H}^{(i)} \leftarrow \mathbf{H}^{(i)} + \eta\mathbf{I}$ has proven very effective in compensating for this problem (see Section 5.4). As the perceptron algorithm, we ran the algorithm for 100 iterations, which appeared to be sufficient for convergence. With i iterations the computational complexity is $\mathcal{O}(p^2 in + p^3 i)$.

3 Slow Feature Analysis

Let $\{\boldsymbol{x}_t\}_{t=1}^n \subset \mathcal{X}$ be a sequence of n observations. The goal of *slow feature analysis* (SFA) is to find a set of mappings $\phi_i : \mathcal{X} \to \mathbb{R}$, $i \in \{1, \dots, p\}$, such that the values $\phi_i(\boldsymbol{x}_t)$ change slowly over time (Wiskott and Sejnowski, 2002). A mapping ϕ_i's change over time is measured by the *discrete temporal derivative* $\dot{\phi}_i(\boldsymbol{x}_t) := \phi_i(\boldsymbol{x}_t) - \phi_i(\boldsymbol{x}_{t-1})$. The SFA objective (called *slowness*, Equation 6) is to minimize the squared mean of this derivative, where $\mathbb{E}_t[\cdot]$ is the sample mean[6] over all available indices t:

$$\min \quad s(\phi_i) := \mathbb{E}_t[\dot{\phi}_i^2(\boldsymbol{x}_t)] \quad \text{(Slowness)} \tag{6}$$

To avoid trivial solutions as well as to deal with mixed sensor observations in different scales, the mappings are forced to change uniformly, i.e. to exhibit *unit variance* (Equations 7 and 8). *Decorrelation* ensures every mapping to extract unique information (Equation 9). The last degree of freedom is eliminated by demanding *order* (Equation 10), leading to the following constraints:

$$\mathbb{E}_t[\phi_i(\boldsymbol{x}_t)] = 0 \qquad \text{(Zero Mean)} \tag{7}$$

$$\mathbb{E}_t[\phi_i^2(\boldsymbol{x}_t)] = 1 \qquad \text{(Unit Variance)} \tag{8}$$

$$\mathbb{E}_t[\phi_i(\boldsymbol{x}_t)\phi_j(\boldsymbol{x}_t)] = 0, \forall j \neq i \quad \text{(Decorrelation)} \tag{9}$$

$$\forall j > i : s(\phi_i) \leq s(\phi_j) \qquad \text{(Order)} \tag{10}$$

The principle of slowness, although not the above definition, has been used very early in the context of neural networks (Földiák, 1991; Becker and Hinton, 1992). Recent variations of SFA differ either in the objective (Bray and Martinez, 2002) or the constraints (Einhäuser et al., 2005; Wyss et al., 2006). For some simplified cases, given an infinite time series and unrestricted function class, it can be analytically shown that SFA solutions converge to trigonometric polynomials w.r.t. the underlying latent variables (Wiskott, 2003; Franzius et al., 2007). In reality those conditions are never met and one requires a function class that can be adjusted to the data set at hand.

[5] This regularization approach is equivalent to *weight decay*, i.e. an additional regularization term $\eta\|\boldsymbol{w}\|_2^2$ in the log-likelihood objective.

[6] The samples are not i.i.d and must be drawn by an *ergodic* Markov chain in order for the empirical mean to converge in the limit (Meyn and Tweedie, 1993).

3.1 Kernel SFA

Let the considered mappings $\phi_i : \mathcal{X} \to \mathbb{R}$ be elements of a *reproducing kernel Hilbert space* (RKHS) $\mathcal{H}$, with corresponding positive semi-definite kernel $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. The *reproducing property* of those kernels allows $\forall \phi \in \mathcal{H} : \phi(\boldsymbol{x}) = \langle \phi, \kappa(\cdot, \boldsymbol{x}) \rangle_{\mathcal{H}}$, in particular $\langle \kappa(\cdot, \boldsymbol{x}), \kappa(\cdot, \boldsymbol{x}') \rangle_{\mathcal{H}} = \kappa(\boldsymbol{x}, \boldsymbol{x}')$. The *representer theorem* ensures[7] the solution to be in the span[8] of *support functions*, parametrized by the training data (Wahba, 1990), i.e. $\phi = \sum_{t=1}^{n} a_t \kappa(\cdot, \boldsymbol{x}_t)$. Together those two relationships set the basis for the *kernel trick* (see e.g. Shawe-Taylor and Cristianini (2004)).

The zero mean constraint can be achieved by centring all involved *support functions* $\{\kappa(\cdot, \boldsymbol{x}_t)\}_{t=1}^{n}$ in $\mathcal{H}$ (for details see Section 3.2). Afterwards, the combined kernel SFA (K-SFA) optimization problem for all p mappings $\boldsymbol{\phi}(\cdot) \in \mathcal{H}^p$ is

$$\min_{\boldsymbol{\phi} \in \mathcal{H}^p} \sum_{i=1}^{p} \mathbb{E}_t \left[\dot{\phi}_i^2(\boldsymbol{x}_t) \right], \quad \text{s.t. } \mathbb{E}_t \left[\boldsymbol{\phi}(\boldsymbol{x}_t) \boldsymbol{\phi}(\boldsymbol{x}_t)^\top \right] = \mathbf{I}. \tag{11}$$

Through application of the kernel trick, the problem can be reformulated as

$$\begin{aligned} \min_{\mathbf{A} \in \mathbb{R}^{n \times p}} & \tfrac{1}{n-1} \mathbf{tr} \left(\mathbf{A}^\top \mathbf{K} \mathbf{D} \mathbf{D}^\top \mathbf{K}^\top \mathbf{A} \right) \\ \text{s.t. } & \tfrac{1}{n} \mathbf{A}^\top \mathbf{K} \mathbf{K}^\top \mathbf{A} = \mathbf{I}, \end{aligned} \tag{12}$$

where $K_{ij} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the *kernel matrix* and $\mathbf{D} \in \mathbb{R}^{n \times n-1}$ the *temporal derivation matrix*[9] with all zero entries except $\forall t \in \{1, \dots, n-1\} : D_{t,t} = -1$ and $D_{t+1,t} = 1$.

Sparse Kernel SFA. If one assumes the feature mappings within the span of another set of data $\{\boldsymbol{z}_i\}_{i=1}^{m} \subset \mathcal{X}$ (e.g. a sparse subset of the training data, often called *support vectors*), the *sparse kernel matrix* $\mathbf{K} \in \mathbb{R}^{m \times n}$ is defined as $K_{ij} = \kappa(\boldsymbol{z}_i, \boldsymbol{x}_j)$ instead. The resulting algorithm will be called *sparse kernel SFA*. Note that the representer theorem no longer applies and therefore the solution merely approximates the optimal mappings in $\mathcal{H}$. Both optimization problems have identical solutions if $\forall t \in \{1, \dots, n\} : \kappa(\cdot, \boldsymbol{x}_t) \in \mathbf{span}(\{\kappa(\cdot, \boldsymbol{z}_i)\}_{i=1}^{m})$, e.g. $\{\boldsymbol{z}_i\}_{i=1}^{n} = \{\boldsymbol{x}_t\}_{t=1}^{n}$.

Regularized Sparse Kernel SFA. The Hilbert spaces corresponding to some of the most popular kernels are equivalent to an infinite dimensional space of continuous functions (Shawe-Taylor and Cristianini, 2004). One example is the *Gaussian kernel* $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\frac{1}{2\sigma^2} \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2)$. Depending on hyper-parameter σ and data distribution, this can obviously lead to overfitting. Less obvious, however, is the tendency of kernel SFA to become numerically unstable for large σ, i.e. to violate the unit variance constraint. Fukumizu et al. (2007) have shown this analytically for the related *kernel canonical correlation analysis*. Note that both problems do not affect sufficiently sparse solutions, as sparsity reduces the function complexity and sparse kernel matrices $\mathbf{K}\mathbf{K}^\top$ are more robust w.r.t. eigenvalue decompositions.

One countermeasure is to introduce a *regularization term* to stabilize the sparse kernel SFA algorithm, which thereafter will be called *regularized sparse kernel SFA* (RSK-SFA). Our

[7] Technically this holds only when the solution is regularized in $\mathcal{H}$ - which we will do later.

[8] Note that the solutions $\phi_i(\cdot)$ are thus *linear functions* of *inner products* in $\mathcal{H}$.

[9] In Section 5 we will generate feature mappings from a collection of time series, i.e. multiple words. If all words are assembled into one large time series, the transition from one word to the next can be excluded from optimization by setting the respective entry in $\mathbf{D}$ to zero.

approach penalizes the squared Hilbert-norm of the selected functions $\|\phi_i\|^2_{\mathcal{H}}$ by a *regularization parameter* λ. Analogous to K-SFA the kernel trick can be utilized to obtain the new objective:

$$\min_{\mathbf{A}\in\mathbb{R}^{m\times p}} \tfrac{1}{n-1}\mathbf{tr}\left(\mathbf{A}^\top\mathbf{K}\mathbf{D}\mathbf{D}^\top\mathbf{K}^\top\mathbf{A}\right) + \lambda\,\mathbf{tr}(\mathbf{A}^\top\bar{\mathbf{K}}\mathbf{A}) \quad \text{s.t. } \tfrac{1}{n}\mathbf{A}^\top\mathbf{K}\mathbf{K}^\top\mathbf{A} = \mathbf{I}\,, \tag{13}$$

where $\bar{K}_{ij} = \kappa(\boldsymbol{z}_i, \boldsymbol{z}_j)$ is the kernel matrix of the support vectors.

3.2 The RSK-SFA Algorithm

The RSK-SFA algorithm (Algorithm 1) is closely related to the linear SFA algorithm of Wiskott and Sejnowski (2002). It consists of three phases: (1) fulfilling zero mean by centring, (2) fulfilling unit variance and decorrelation by sphering and (3) minimizing the objective by rotation.

Zero Mean. To fulfil the zero mean constraint, one centres the *support functions* $\{g_i\}_{i=1}^m \subset \mathcal{H}$ w.r.t. the data distribution, i.e. $g_i(\cdot) := \kappa(\cdot, \boldsymbol{z}_i) - \mathbb{E}_t[\kappa(\boldsymbol{x}_t, \boldsymbol{z}_i)]\,\mathbf{1}_{\mathcal{H}}(\cdot)$, where $\forall \boldsymbol{x} \in \mathcal{X} : \mathbf{1}_{\mathcal{H}}(\boldsymbol{x}) = \langle \mathbf{1}_{\mathcal{H}}, \kappa(\cdot, \boldsymbol{x})\rangle_{\mathcal{H}} := 1$ is the constant function[10] in Hilbert space $\mathcal{H}$. Although $\forall \phi \in \mathbf{span}(\{g_i\}_{i=1}^m) : \mathbb{E}_t[\phi(\boldsymbol{x}_t)] = 0$ already holds, it is of advantage to centre the support functions as well w.r.t. each other (Schölkopf et al., 1998), i.e. $\hat{g}_i := g_i - \mathbb{E}_j[g_j]$. The resulting transformation of support functions on the training data can be applied directly onto the kernel matrices $\mathbf{K}$ and $\bar{\mathbf{K}}$:

$$\hat{\mathbf{K}} := (\mathbf{I} - \tfrac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top)\,\mathbf{K}\,(\mathbf{I} - \tfrac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top) \tag{14}$$

$$\hat{\bar{\mathbf{K}}} := (\mathbf{I} - \tfrac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top)\,\bar{\mathbf{K}}\,(\mathbf{I} - \tfrac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top)\,, \tag{15}$$

where $\mathbf{1}_m$ and $\mathbf{1}_n$ are one-vectors of dimensionality m and n, respectively.

Unit Variance and Decorrelation. Analogue to linear SFA, we first project into the normalized eigenspace of $\frac{1}{n}\hat{\mathbf{K}}\hat{\mathbf{K}}^\top =: \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$. The procedure is called *sphering* or *whitening* and fulfils the constraint in Equation 13, invariant to further rotations $\mathbf{R} \in \mathbb{R}^{m\times p} : \mathbf{R}^\top\mathbf{R} = \mathbf{I}$.

$$\mathbf{A} := \mathbf{U}\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{R} \quad \Rightarrow \quad \tfrac{1}{n}\mathbf{A}^\top\hat{\mathbf{K}}\hat{\mathbf{K}}^\top\mathbf{A} = \mathbf{R}^\top\mathbf{R} = \mathbf{I} \tag{16}$$

Note that an inversion of the diagonal matrix $\mathbf{\Lambda}$ requires the removal of zero eigenvalues and corresponding eigenvectors first.

Minimization of the Objective. Application of Equation 16 allows us to solve Equation 13 with a second eigenvalue decomposition:

$$\min_{\mathbf{R}} \mathbf{tr}\left(\mathbf{R}^\top\left\{\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{U}^\top\mathbf{B}\mathbf{U}\mathbf{\Lambda}^{-\frac{1}{2}}\right\}\mathbf{R}\right) \quad \text{s.t.}\, \mathbf{R}^\top\mathbf{R} = \mathbf{I} \quad \text{with} \quad \mathbf{B} := \tfrac{1}{n-1}\hat{\mathbf{K}}\mathbf{D}\mathbf{D}^\top\hat{\mathbf{K}}^\top + \lambda\hat{\bar{\mathbf{K}}}\,. \tag{17}$$

Note that $\mathbf{R}$ is composed of the eigenvectors to the p *smallest* eigenvalues of the above expression.

[10] Not all Hilbert spaces $\mathcal{H}$ contain $\mathbf{1}_{\mathcal{H}}$. Technically we must optimize over $\mathcal{H}\cup\{\mathbf{1}_{\mathcal{H}}\}$. This can be achieved by allowing solutions of the form $\phi_i(\cdot) = \sum_{j=1}^m A_{ji}\,\kappa(\cdot, \boldsymbol{z}_j) - c_i$.

Algorithm 1 Regularized Sparse Kernel Slow Feature Analysis (RSK-SFA)

Input: $\mathbf{K} \in \mathbb{R}^{m\times n}$, $\bar{\mathbf{K}} \in \mathbb{R}^{m\times m}$, $p \in \mathbb{N}$, $\lambda \in \mathbb{R}^+$, $\mathbf{D} \in \mathbb{R}^{n\times n-1}$

$\hat{\mathbf{K}} = (\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top)\mathbf{K}(\mathbf{I} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top)$ (Eq. 14)

$\hat{\bar{\mathbf{K}}} = (\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top)\bar{\mathbf{K}}(\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top)$ (Eq. 15)

$\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top = \mathrm{eig}\left(\frac{1}{n}\hat{\mathbf{K}}\hat{\mathbf{K}}^\top\right)$

$(\mathbf{U}_r, \mathbf{\Lambda}_r) = \mathrm{remove\_zero\_eigenvalues}(\mathbf{U}, \mathbf{\Lambda})$

$\mathbf{B} = \frac{1}{n-1}\hat{\mathbf{K}}\mathbf{D}\mathbf{D}^\top\hat{\mathbf{K}}^\top + \lambda\hat{\bar{\mathbf{K}}}$ (Eq. 17)

$\mathbf{R}\mathbf{\Sigma}\mathbf{R}^\top = \mathrm{eig}\left(\mathbf{\Lambda}_r^{-1/2}\mathbf{U}_r^\top\mathbf{B}\mathbf{U}_r\mathbf{\Lambda}_r^{-1/2}\right)$ (Eq. 17)

$(\mathbf{R}_p, \mathbf{\Sigma}_p) = \mathrm{keep\_lowest\_p\_eigenvalues}(\mathbf{R}, \mathbf{\Sigma}, p)$

$\hat{\mathbf{A}} = \left(\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top\right)\mathbf{U}_r\mathbf{\Lambda}_r^{-1/2}\mathbf{R}_p$ (Eq. 16 + 19)

$\hat{\boldsymbol{c}} = \frac{1}{n}\hat{\mathbf{A}}^\top\mathbf{K}\mathbf{1}_n$ (Eq. 19)

Output: $\hat{\mathbf{A}}$, $\hat{\boldsymbol{c}}$

Solution. After the above calculations the i'th RSK-SFA solution is

$$\phi_i(\boldsymbol{x}) = \sum_{j=1}^{m} A_{ji}\langle \hat{g}_j, \kappa(\cdot, \boldsymbol{x})\rangle_{\mathcal{H}} \tag{18}$$

Grouping the kernel functions of all support vectors together in a column vector, i.e. $\boldsymbol{k}(\boldsymbol{x}) = [\kappa(\boldsymbol{z}_1, \boldsymbol{x}), \ldots, \kappa(\boldsymbol{z}_m, \boldsymbol{x})]^\top$, the combined solution $\boldsymbol{\phi}(\cdot) \in \mathcal{H}^p$ can be expressed more compactly:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \hat{\mathbf{A}}^\top\boldsymbol{k}(x) - \hat{\boldsymbol{c}} \tag{19}$$
$$\text{with } \hat{\mathbf{A}} := \left(\mathbf{I} - \tfrac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top\right)\mathbf{A} \text{ and } \hat{\boldsymbol{c}} := \tfrac{1}{n}\hat{\mathbf{A}}^\top\mathbf{K}\mathbf{1}_n\ .$$

The computational complexity is $\mathcal{O}(m^2 n)$. For illustrative purposes, Algorithm 1 exhibits a memory complexity of $\mathcal{O}(mn)$. An online calculation of $\hat{\mathbf{K}}\hat{\mathbf{K}}^\top$ and $\hat{\mathbf{K}}\mathbf{D}\mathbf{D}^\top\hat{\mathbf{K}}^\top$ reduces this to $\mathcal{O}(m^2)$.

4 Sparse Subset Selection

Representer theorems guarantee that the target function $\phi^* \in \mathcal{H}$ for training set $\{\boldsymbol{x}_t\}_{t=1}^n$ can be found within $\mathbf{span}(\{\kappa(\cdot, \boldsymbol{x}_t)\}_{t=1}^n)$. For sparse K-SFA, however, no such guarantee exists. The quality of such a sparse approximation depends exclusively on the set of support vectors $\{\boldsymbol{z}_i\}_{i=1}^m$.

Without restriction on ϕ^*, it is straight forward to select a subset of the training data, indicated by an *index vector*[11] $\boldsymbol{i} \in \mathbb{N}^m$ with $\{\boldsymbol{z}_j\}_{j=1}^m := \{\boldsymbol{x}_{\boldsymbol{i}_j}\}_{j=1}^m$, that minimizes the *approximation error*

$$\begin{aligned}\epsilon_t^{\boldsymbol{i}} &:= \min_{\boldsymbol{\alpha}\in\mathbb{R}^m} \left\|\kappa(\cdot, \boldsymbol{x}_t) - \sum_{j=1}^{m}\alpha_j\kappa(\cdot, \boldsymbol{x}_{\boldsymbol{i}_j})\right\|_{\mathcal{H}}^2 \\ &= K_{tt} - \boldsymbol{K}_{t\boldsymbol{i}}(\mathbf{K}_{\boldsymbol{i}\boldsymbol{i}})^{-1}\boldsymbol{K}_{\boldsymbol{i}t}\end{aligned}$$

for all training samples $\boldsymbol{x}_t$, where $K_{tj} = \kappa(\boldsymbol{x}_t, \boldsymbol{x}_j)$ is the full kernel matrix. Finding an optimal subset is an NP hard combinatorial problem, but there exist several greedy approximations to it.

[11] Let "**:**" denote the index vector of all available indices. See Algorithm 2.

Algorithm 2 Matching Pursuit Maximization of the Affine Hull (MP MAH)

Input: $\{\boldsymbol{x}_t\}_{t=1}^n \subset \mathcal{X},\ \kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R},\ m \in \mathbb{N}$
$\mathbf{K} = \emptyset; \quad \mathbf{K}_1^{-1} = \emptyset;$
$\forall t \in \{1, \dots, n\} : \epsilon_t^1 = \kappa(\boldsymbol{x}_t, \boldsymbol{x}_t)$ (Eq. 21)
$\boldsymbol{i}_1 = \underset{t}{\operatorname{argmax}}\{\epsilon_t^1\}_{t=1}^n$ (Eq. 20)
for $j \in \{1, \dots, m-1\}$ **do**
  $\boldsymbol{\alpha}^j = \left[\mathbf{K}_{(j,:)}\mathbf{K}_j^{-1}, -1\right]^\top$ (MIL)
  $\mathbf{K}_{j+1}^{-1} = \begin{bmatrix} \mathbf{K}_j^{-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} + \frac{\boldsymbol{\alpha}^j \boldsymbol{\alpha}^{j\top}}{\epsilon_{\boldsymbol{i}_j}^j}$ (MIL)
  for $t \in \{1, \dots, n\}$ **do**
    $\mathbf{K}_{(t,j)} = \kappa(\boldsymbol{x}_t, \boldsymbol{x}_{\boldsymbol{i}_j})$
    $\epsilon_t^{j+1} = \epsilon_t^j - \frac{1}{\epsilon_{\boldsymbol{i}_j}^j}\left(\mathbf{K}_{(t,:)}\boldsymbol{\alpha}^j\right)^2$ (Eq. 21)
  end for
  $\boldsymbol{i}_{j+1} = \underset{t}{\operatorname{argmax}}\{\epsilon_t^{j+1}\}_{t=1}^n$ (Eq. 20)
end for
Output: $\{\boldsymbol{i}_1, \dots, \boldsymbol{i}_m\}$

Online Maximization of the Affine Hull. A widely used algorithm (Csató and Opper, 2002), which we will call *online maximization of the affine hull* (online MAH) in the absence of a generally accepted name, iterates through the data in an online fashion. At time t, sample $\boldsymbol{x}_t$ is added to the selected subset if $\epsilon_t^{\boldsymbol{i}}$ is larger than some given threshold η. Exploitation of the *matrix inversion lemma* (MIL) allows an online algorithm with computational complexity $\mathcal{O}(m^2 n)$ and memory complexity $\mathcal{O}(m^2)$. The downside of this approach is the unpredictable dependence of the final subset size m on hyper-parameter η. Changing the subset size requires therefore a complete re-computation with larger η. The resulting subset size is not predictable, although monotonically dependent on η.

Matching Pursuit for Sparse Kernel PCA. This handicap is addressed by *matching pursuit* methods (Mallat and Zhang, 1993). Applied on kernels, some criterion selects the best fitting sample, followed by an orthogonalization of all remaining candidate support functions in Hilbert space $\mathcal{H}$. A resulting sequence of m selected samples therefore contains all sequences up to length m as well. The batch algorithm of Smola and Schölkopf (2000) chooses the sample $\boldsymbol{x}_j$ that minimizes[12] $\mathbb{E}_t\left[\epsilon_t^{\boldsymbol{i} \cup j}\right]$. It was shown later that this algorithm performs sparse PCA in $\mathcal{H}$ (Hussain and Shawe-Taylor, 2008). The algorithm, which we will call in the following *matching pursuit for sparse kernel PCA* (MP KPCA), has a computational complexity of $\mathcal{O}(n^2 m)$ and a memory complexity of $\mathcal{O}(n^2)$. In practice it is therefore not applicable to large data sets.

4.1 Matching Pursuit for Online MAH

The ability to change the size of the selected subset without re-computation is a powerful property of MP KPCA. Run-time and memory consumption, however, make this algorithm infeasible for most applications. To extend the desired property to the fast online algorithm, we derive a novel *matching pursuit for online MAH* algorithm (MP MAH). Online MAH selects samples with approximation errors that exceed the threshold η and therefore forces the

[12] $\epsilon_t^{\boldsymbol{i}}$ is non-negative and MP KPCA thus minimizes the L_1 norm of approximation errors.

supremum norm L_∞ of all samples below η. This is analogous to a successive selection of the *worst* approximated sample, until the approximation error of all samples drops below η. The matching pursuit approach therefore minimizes the supremum norm L_∞ of the approximation error[13]. At iteration j, given the current subset $\boldsymbol{i} \in \mathbb{R}^j$,

$$\boldsymbol{i}_{j+1} := \operatorname*{argmin}_t \|[\epsilon_1^{\boldsymbol{i}\cup t}, \ldots, \epsilon_n^{\boldsymbol{i}\cup t}]\|_\infty \approx \operatorname*{argmax}_t \epsilon_t^{\boldsymbol{i}} \tag{20}$$

Straight forward re-computation of the approximation error in every iteration is expensive. Using the matrix inversion lemma (MIL), this computation can be performed iteratively:

$$\epsilon_t^{\boldsymbol{i}\cup j} = \epsilon_t^{\boldsymbol{i}} - \frac{1}{\epsilon_j^{\boldsymbol{i}}}\Big(K_{tj} - \boldsymbol{K}_{t\boldsymbol{i}}(\mathbf{K}_{\boldsymbol{i}\boldsymbol{i}})^{-1}\boldsymbol{K}_{\boldsymbol{i}j}\Big)^2 . \tag{21}$$

Algorithm 2 iterates between sample selection (Equation 20) and error update (Equation 21). The complexity is $\mathcal{O}(m^2 n)$ in time and $\mathcal{O}(mn)$ in memory (to avoid re-computations of $\mathbf{K}_{(:,\boldsymbol{i})}$).

5 Empirical Validation

In this section we will evaluate the hypothesis that *RSK-SFA generates a feature space Φ which resembles a Fourier basis in the space of latent variables Θ for a given time series* at the example of a two-vowel classification task.

The true Θ is not known and we use the performance of different linear classification algorithms to measure how well RSK-SFA features encode Θ. For comparison, all algorithms are tested on feature spaces generated by RSK-SFA and *sparse kernel PCA* (Schölkopf et al., 1997). Results in Section 5.4 show that RSK-SFA features indeed encode Θ, whereas kernel PCA does not aim for such an encoding. We also demonstrate the importance of sparse subset selection to an efficient encoding. Last but not least, a classification accuracy of more than 97% in a task that is not linearly solvable demonstrates the excellent performance of our approach.

5.1 Benchmark Data Sets

The "north Texas vowel database"[14] contains uncompressed audio files with English words of the form H…D, spoken multiple times by multiple persons (Assmann et al., 2008). The natural task is to predict the central vowels of unseen instances of a trained word. To cover both cases of small and large training sets, we selected two data sets: (i) A small set with four training and four test instances for each of the words "heed" and "head", spoken by the same person. (ii) A large data set containing all instances of "heed" and "head" for all 18 adult subjects. To apply cross-validation, we performed 20 random splits (folds) into 12 training and 6 test subjects, leading to 245 ± 10 training instances and 116 ± 10 test instances.

The spoken words are provided as mono audio streams of varying length at 48kHz, i.e. as a series of amplitude readings $\{a_1, a_2, \ldots\}$. SFA requires the space of latent variables Θ to be *embedded* in the space of observations $\mathcal{X}$, which is not the case for one-dimensional data. The problem resides in the ambiguity of the observed amplitude readings a_t, i.e. mappings ϕ_i :

[13] An exact minimization of the L_∞ norm is as expensive as the MP KPCA algorithm. However, since $\epsilon_t^{\boldsymbol{i}\cup t} = 0$, selecting the worst approximated sample $\boldsymbol{x}_t$ effectively minimizes the supremum norm L_∞.

[14] `http://www.utdallas.edu/~assmann/KIDVOW1/North_Texas_vowel_database.html`

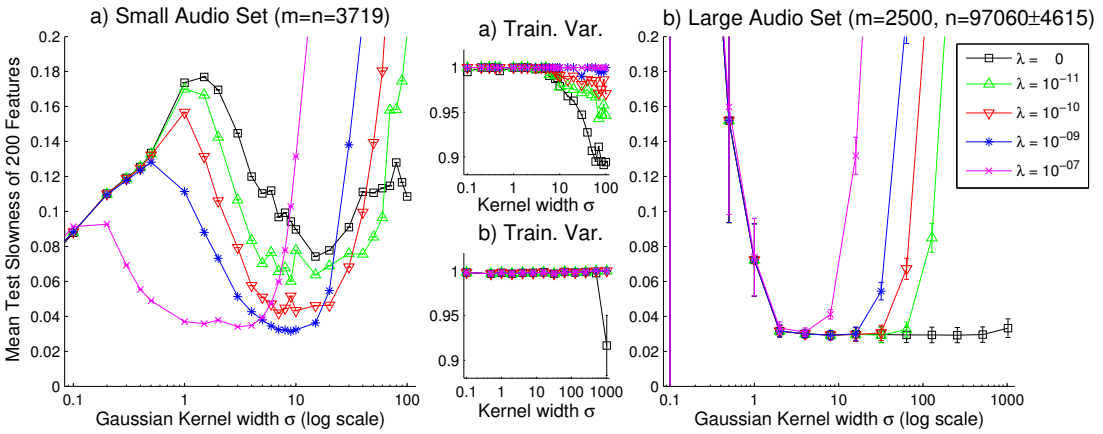


Fig. 1 Mean test slowness of 200 RSK-SFA features over varying kernel parameter σ for different regularization parameters λ: **(a)** *Small audio data set* with all $m = n = 3719$ training samples as support vectors, **(b)** *large audio data set* with $m = 2500$ support vectors selected out of $n = 97060 \pm 4615$ training samples by MP MAH. The small plots show the variance of the respective training output. Significant deviation from one violates the unit variance constraint and thus demonstrates numerical instability. The legend applies to all plots.

$\mathcal{X} \to \mathbb{R}^p$ can not distinguish between two latent states which both generate the same observed amplitude a_t. However, Takens Theorem (Takens, 1981; Huke, 2006) guarantees an embedding of Θ as a manifold in the space $\mathcal{X} \subset \mathbb{R}^l$ of sufficiently many *time-delayed* observations. Based on the observed pattern of l time-delayed amplitude readings, non-linear SFA and PCA can utilize the resulting one-to-one mapping of latent variables in Θ onto a manifold of observations in $\mathcal{X}$ to extract basis functions $\phi_i(\cdot)$ that encode Θ. We therefore defined our samples $\boldsymbol{x}_t := [a_{\delta t}, a_{\delta t+\epsilon}, a_{\delta t+2\epsilon}, \dots, a_{\delta t+(l-1)\epsilon}]^\top$, which is also called *sliding window*[15]. We evaluated the parameters δ, ϵ and l empirically[16] and chose $\delta = 50$, $\epsilon = 5$ and $l = 500$. All algorithms were trained and tested with the joint[9] time series $\{\boldsymbol{x}_t\}$ of all instances of all subjects in the respective fold. Note, however, that all samples of each word-instance were exclusively used for either training or testing. Additionally, in the large data set all instances of one subject were exclusively used, too. This ensures that we really test for generalization to unseen instances of the trained words as well as to previously unseen subjects. The above procedure provided us with 3719 (4108) trainings (test) samples $\boldsymbol{x}_t \in [-1, 1]^{500}$ for the small and 97060 ± 4615 (46142 ± 4614) trainings (test) samples for the large data set.

5.2 RSK-SFA Performance

We start our analysis with the RSK-SFA solution for different Gaussian kernels, i.e. $k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{1}{2\sigma^2}\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2\right)$ with varying σ. To ensure that meaningful information is extracted, we measure the *test slowness*, i.e. the slowness of the learned feature mappings applied on a previously unseen test sequence drawn from the same distribution. Small σ, however, grossly

[15] This violates the *i.i.d. assumption* of most classification algorithms, as two successive samples are no longer independent. The classification results were excellent, however, indicating that the violation did not influence the algorithms' performance too strongly.

[16] Although the choice of embedding parameters change the resulting slowness in a nontrivial fashion, we want to point out that this change appears to be smooth and the presented shapes similar over a wide range of embedding parameters.

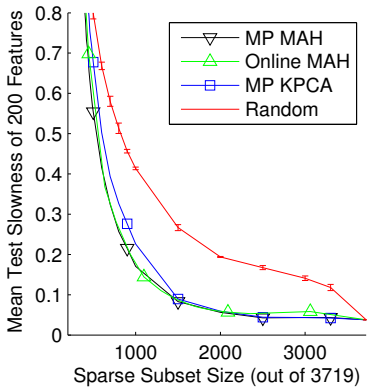


Fig. 2 Mean test slowness of 200 RSK-SFA features of the *small audio data set* ($\lambda = 10^{-7}, \sigma = 2$) over sparse subset size, selected by different algorithms. For *random selection* the mean of 10 trials is plotted with standard deviation error bars.

underestimate a feature's output on unseen test samples, as distances are strongly amplified. This changes the feature's slowness. For comparison we normalized all outputs to unit variance *on the test set* before measuring the test slowness.

Figure 1 shows the mean test slowness of 200 RSK-SFA features[17] on both sets for multiple kernel parameter σ and regularization parameter λ. The small data set (a) uses the complete training set as support vectors, whereas for the large data set (b) a full kernel approach is not feasible. Instead we selected a subset of size 2500 with the MP MAH algorithm (based on kernel parameter $\sigma = 5$) before training.

In the absence of significant sparseness (Figure 1a), unregularized kernel SFA ($\lambda = 0$, equivalent to K-SFA, Equations 11 and 12) shows both over-fitting and numerical instability. Over-fitting can be seen at small σ, where the features fulfil the unit variance constraint (small plot), but do not reach the minimal test slowness (main plot). The bad performance for larger σ, on the other hand, must be blamed on numerical instability, as indicated by a significantly violated unit variance constraint. Both can be counteracted by proper regularization. Although optimal regularization parameters λ are quite small and can reach computational precision, there is a wide range of kernel parameters σ for which the same minimal test slowness is reachable. E.g. in Figure 1a, a fitting λ can be found between $\sigma = 0.5$ and $\sigma = 20$.

The more common case, depicted in Figure 1b, is a large training set from which a small subset is selected by MP MAH. Here no regularization is necessary and unregularized sparse kernel SFA ($\lambda = 0$) learns mappings of minimal slowness in the range from $\sigma = 1$ to far beyond $\sigma = 500$. Notice that this is rendering a time consuming search for an optimal parameter σ obsolete.

5.3 Sparse Subset Selection

To evaluate the behaviour of RSK-SFA for sparse subsets of different size, Figure 2 plots the test slowness of the small data set for all discussed sparse subset selection algorithms in Section 4. As a baseline, we plotted mean and standard deviation of a random selection

[17] Comparison to linear SFA features is omitted due to scale, e.g. test slowness was slightly above 0.5 for both data sets. RSK-SFA can therefore outperform linear SFA by more than a factor of 10, a magnitude we observed in other experiments as well.

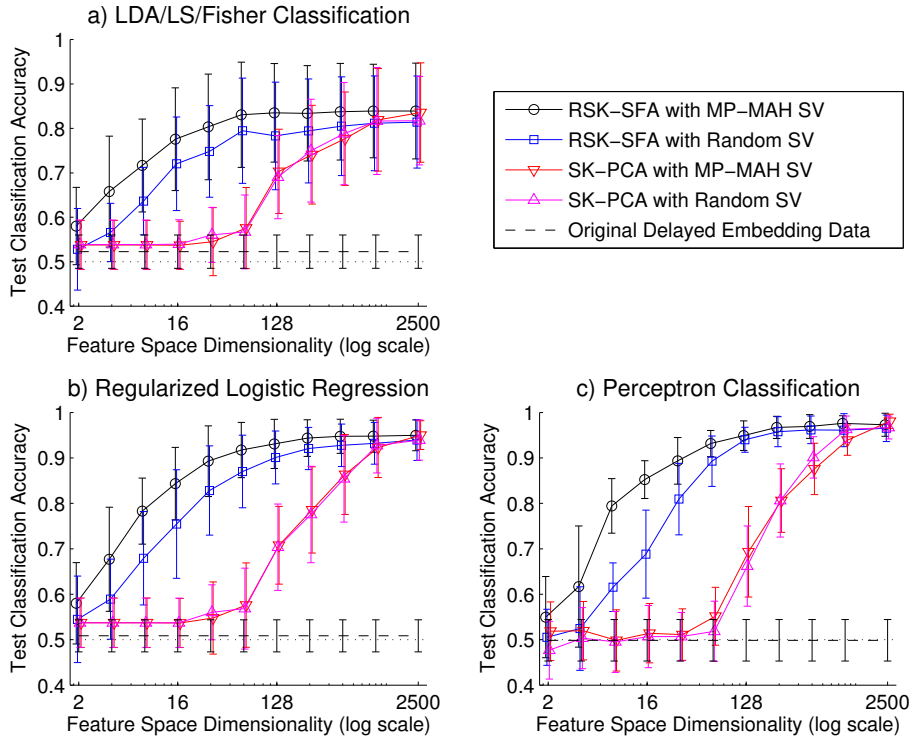


Fig. 3 Accuracy of vowel detection on unseen test set against feature space size for all discussed linear classification algorithms (a-c). Each algorithm was tested on feature spaces generated by RSK-SFA ($\lambda = 0, \sigma = 5$) and SK-PCA with 2500 support vectors selected at random or with MP MAH. As a comparison the accuracy on the original 500 dimensional space of delayed audio amplitudes is given as well (dashed line). All accuracies are plotted with mean and standard deviation for 20 random splits into training and test sets.

scheme. One can observe that all algorithms surpass the random selection significantly but do not differ much w.r.t. each other. As expected, the MP MAH and Online MAH algorithms perform virtually identical. The novel MP MAH, however, allows unproblematic and fast fine tuning of the selected subset's size.

5.4 Classification Performance

To demonstrate that Φ encodes Θ, we compared the accuracy of linear classification algorithms (see Section 2) in RSK-SFA and SK-PCA feature spaces of different sizes. Sparse kernel PCA optimizes the PCA objective (maximal variance) on the same function class as RSK-SFA, i.e. with the same kernel and support vectors. Because some iterative algorithms are sensitive to scaling, we scaled the SK-PCA features to unit variance *on the test set*, to make them fully comparable with RSK-SFA features.

In the evaluated two-class problem ("head" vs. "heed"), class labels are only available for whole words. Individual samples $\boldsymbol{x}_t$, however, do not necessarily belong to a vowel, but might contain one of the consonants in the beginning or end of the word. Many samples will therefore be ambiguously labelled, which the classifier must interpret as noise. When all samples of a word are classified, the final judgement about the spoken vowel is determined by the sum over all *discrimination function* outputs $\sum_{t=1}^{n} f(\boldsymbol{x}_t)$, rather than over the individual

class predictions. This is supposed to take the classifiers certainty into account and yields much better results. It is also equivalent to the empirical *temporal mean* of the discriminant function output over the whole word[18].

This shift from training each sample to evaluating only the mean over all samples might look like a completely different problem at first. One could rightfully demand to train the classifiers on the mean feature vector $\bar{\boldsymbol{\phi}} := \mathbb{E}_t[\boldsymbol{\phi}(\boldsymbol{x}_t)]$ of each word instead of all individual samples thereof. Nonetheless, the authors chose the presented methodology to compare SFA and PCA feature spaces because

1. a training set of 245 ± 10 words must over-fit in high dimensional feature spaces.
2. classification of individual samples $\boldsymbol{x}_t$ must always result in very poor accuracy as most of them do not encode the vowel and will thus yield only noise.
3. $\bar{\boldsymbol{\phi}}$ lives in the *same feature space* as the individual samples and should be affected by the *same insufficiencies* of that feature space Φ.

Figure 3 shows mean and standard deviation of the *test accuracy*[19] over 20 random splits of the large data set into training and test set (see Section 5.1) against the number of RSK-SFA or SK-PCA features. Both feature spaces are generated using the same support vectors selected at random or by the novel MP MAH algorithm. To demonstrate that this is not a trivial problem, the algorithms' performance on the original 500 dimensional delayed amplitude space is given as a dashed line.

It is apparent that in all cases the RSK-SFA feature space is superior to SK-PCA for all but 2500 features. At this point both feature spaces are identical up to rotation and any linear algorithm must therefore perform identical. Moreover, RSK-SFA features based on support vectors selected by MP MAH clearly outperform those based on randomly selected SV in all cases. In a comparison between algorithms, perceptron classification (Figure 3c) yields the highest accuracy with the least features, i.e. more that 97% with 256 RSK-SFA features. More features do not significantly raise the performance. This is particularly interesting because the training does not require a matrix inverse and is therefore the fastest of the evaluated algorithms in large feature spaces. It is also worth mentioning that 8 (32) RSK-SFA features reach an accuracy of 80% (90%), whereas the first 64 SK-PCA features apparently do not encode any information about Θ. Regularized logistic regression[20] (Figure 3b) reaches comparable accuracy (around 95% with 256 or more features) but exhibits higher variance between folds. The poor performance of LS/LDA/FDA classification (Figure 3a) must stem from the violated assumption of equal class covariances (see Section 2). Note, however, that all above comparisons between feature spaces and SV selection schemes still hold.

These results demonstrate that, in difference to SK-PCA, RSK-SFA feature spaces Φ aim to encode Θ. The high accuracy of over 97% establishes that either (i) the problem is almost linearly separable in Θ or (ii) Φ resembles a functional basis in Θ, presumably[21] a Fourier basis. At this point in time it is not possible to distinguish between (i) and (ii). However, the large number of required features could be interpreted as evidence for (ii).

[18] Note that this is not the same as the discriminant output of the temporal mean over the whole word, i.e. $\mathbb{E}_t[f(\boldsymbol{x}_t)] = \sum_{i=1}^{p} w_i \, \mathbb{E}_t[\phi_i(\boldsymbol{x}_t)] \neq f(\mathbb{E}_t[\boldsymbol{x}_t])$, as $\phi_i(\cdot)$ are non-linear.

[19] *Test accuracy* is the fraction of correct classifications for an previously unseen test set.

[20] The unregularized algorithm showed clear signs of over-fitting in large feature spaces, i.e. the accuracy dropped for large p. We repeated the training and evaluation procedure with slowly increasing regularization parameter η. The (presented) results stabilized at $\eta = 500$.

[21] The functional form in Θ can not be derived from the presented results. However, theoretical works by Wiskott (2003) and Franzius et al. (2007) suggest that Φ approximates a *Fourier basis* in Θ, as discussed in Sections 1 and 3.

We therefore conclude that RSK-SFA features encode the space of latent variables Θ of a given time series, presumably by resembling a Fourier basis of Θ.

6 Discussion

This article investigates the hypothesis that sufficiently powerful *non-linear slow feature analysis features resemble a Fourier basis in the space of latent variables Θ*, which underlies complex real-world time series. To perform a powerful but easily operated non-linear SFA, we derived a kernelized SFA algorithm (RSK-SFA). The novel algorithm is capable of handling small data sets by *regularization* and large data sets through *sparsity*. To select a sparse subset for the latter, we developed a matching pursuit approach to a widely used algorithm (MP MAH). In combination with linear classification algorithms, particularly the perceptron algorithm, our results support the hypothesis' validity and demonstrate excellent performance.

6.1 Comparison to Previous Works

The major advancement of our approach over the kernel SFA algorithm of Bray and Martinez (2002) is the ability to obtain features that generalize well for *small data sets*. If one is forced to use a large proportion of the training set as support vectors, e.g. for small training sets of complex data, the solution can violate the unit variance constraint.

As suggested by Bray and Martinez (2002), our experiments show that for large data sets no explicit regularization is needed. The implicit regularization introduced by sparsity is sufficient to generate features that generalize well over a wide range of Gaussian kernels. However, our work documents the dependence of the algorithms' performance on the sparse subset for the first time. In this setting, the subset size m takes the role of regularization parameter λ. It is therefore imperative to control m with minimal computational overhead.

We compared two state-of-the-art algorithms that select sparse subsets in polynomial time. *Online MAH* is well suited to process large data sets, but selects unpredictably large subsets. A change of m therefore requires a full re-computation without the ability to target a specific size. *Matching pursuit for sparse kernel PCA* (MP KPCA), on the other hand, returns an ordered list of selected samples. Increasing the subsets size requires simply another loop of the algorithm. The downside is a quadratic dependency on the training set size, both in time and memory. Both algorithms exhibited similar performance and significantly outperformed a random selection scheme. The subsets selected by the novel *matching pursuit to online MAH* (MP MAH) algorithm yielded virtually the same performance as those selected by Online MAH. There is no difference in computation time, but the memory complexity of MP MAH is linearly dependent on the training sets size. However, increasing m works just as with MP KPCA, which makes this algorithm the better choice if one can afford the memory. If not, Online MAH can be applied several times with slowly decreasing hyper-parameter η. Although a subset of suitable size will eventually be found, this approach will take much more time than MP MAH.

6.2 Hyper-Parameter Selection

For a first assessment of the feature space generated by RSK-SFA, selecting the support vectors randomly is sufficient. For optimal performance, however, one should select the sparse

subset with MP MAH or Online MAH, depending on available resources and time (see last section). As shown in Figure 1b, selecting the Gaussian kernel parameter σ is not an issue in face of sufficient sparsity. Empirically, setting σ such that more than half of all kernel outputs are above 0.5 has yielded sufficient performance in most cases.

Before raising the regularization parameter λ above 0, it suggests itself to check the unit variance constraint on the training data. Particularly if the kernel can not be adjusted, a violation can be compensated by slowly rising λ until the variance is sufficiently close to one. A numerically stable solution is no guarantee for optimal slowness, though. It is therefore always recommendable to track the test slowness on an independent test set. When in doubt of over-fitting, try to shrink the sparse subset size m, raise λ or increase kernel width σ, in this order.

6.3 Limitations

The methodology developed in this article will work in most standard machine learning scenarios involving time series. Although not unique to our method, there are certain limitations that are worth mentioning.

1. Our method of generating feature spaces Φ is based on the assumption of an underlying space of latent variables Θ. This implies that the *generative process* $\Theta \to \mathcal{X}$ is *stationary* over time. Preliminary experiments on EEG data (not shown), which are known to be non-stationary, yielded no generalizing features.
2. The generative mapping must be unique. Two elements of Θ which are reliably mapped onto the same element of $\mathcal{X}$ are not distinguishable in Φ.
3. As the exact nature of Θ is ambiguous, there can be arbitrary many "unwanted" latent variables, which raises three problems: (i) If the slowest variables are not relevant for the task, they will appear as noise. (ii) The size of a Fourier basis Φ of Θ grows exponential in the dimensionality of Θ. (iii) Generating a reliable feature space Φ requires training samples from all regions of Θ, which could require infeasible amounts of training samples if Θ is high dimensional.

6.4 Including Label or Importance Information

Applied on real-world time series, RSK-SFA features resemble a Fourier basis in Θ. We want to discuss two possible modifications to generate feature spaces.

1. SFA exploits the temporal succession of samples and thus can only be applied on time series. It is imaginable, however, to use other available information to modify the approach without much change to the methodology. For example, if one is faced with labelled *iid* data, Algorithm 1 can be modified with a different *objective* to minimize the distance between *all* samples of *each class*:

$$\min_{\phi \in \mathcal{H}^p} \sum_{k=1}^{p} \mathbb{E}_i \left[\mathbb{E}_j \left[\delta_{c_i c_j} \left(\phi_k(\boldsymbol{x}_i) - \phi_k(\boldsymbol{x}_j) \right)^2 \right] \right], \tag{22}$$

where $\delta_{c_i c_j}$ is the Kronecker delta, which is 1 if $c_i = c_j$ and 0 otherwise. The two nested expectations induce a quadratic dependency on the training set size n, but can be approximated by randomly drawing sample pairs from the same class. The resulting feature

space Φ will not resemble a functional basis in Θ, but reflect a between-classes metric by mapping same class samples close to each other. Preliminary experiments (not shown) have generated promising feature spaces for linear classification. This approach has also been taken by Berkes (2005), who used SFA on 2 step "time series" of same class samples of hand written digits.

2. The individual *components* (or dimensions) of Θ are encoded according to their slowness. This can result in an inefficient encoding if the slowest changing components of Θ are not of any use to the task. It is therefore of interest if one can use additional information about the samples to restrict Φ to encode only a relevant *subspace* of Θ, at least in the first p features. If one can define an *importance measure* $p(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t) > 0$ with high values for *transitions within the desired subspace*, the slowness objective in Equation 6 can be modified to

$$\min \quad s'(\phi_i) := \frac{1}{n-1} \sum_{t=1}^{n-1} \frac{\big(\phi_i(\boldsymbol{x}_{t+1}) - \phi_i(\boldsymbol{x}_t)\big)^2}{p(\boldsymbol{x}_{t+1}, \boldsymbol{x}_t)} \,. \tag{23}$$

   As a result, the relative slowness of important transitions is reduced and the respective subspace of Θ will be encoded earlier. Increasing the importance will eventually encode the *subspace spanned by the important transitions only* within the first p features extracted by the modified algorithm.

3. If only arbitrary subsets of Θ are of consequence to the task, another modification can increase encoding by including additional information. If the *samples* can be labelled according to another *importance measure* $q(\boldsymbol{x}) \geq 0$, which marks the important ($q(\boldsymbol{x}) \gg 0$) and unimportant samples ($q(\boldsymbol{x}) \approx 0$), the *unit variance* and *decorrelation constraints* of Equation 11 can be modified, i.e.

$$\mathbb{E}_t\Big[\boldsymbol{\phi}(\boldsymbol{x}_t)\, q(\boldsymbol{x}_t)\, \boldsymbol{\phi}(\boldsymbol{x}_t)^\top\Big] \quad = \quad \mathbf{I} \,, \tag{24}$$

   which would enforce unit variance *on the important samples only*. Slowness still applies to all samples equally and the resulting feature space Φ would map samples of low importance onto[22] each other. Given a powerful enough function class, the modified algorithm should construct a Fourier basis in Θ on the subset of important samples only[23]. For example, if we would know which samples $\boldsymbol{x}_t$ contain a vowel (but not which vowel it is), the resulting feature space Φ should exclusively encode the subset of Θ that represents the vowels. When this subset is relatively small, the resulting feature space Φ will approximate the same task with much less basis functions.

6.5 Relationship to Deep Belief Networks

The sound performance of the perceptron algorithm in combination with non-linear SFA suggests an interesting connection to *deep belief networks* (Hinton and Salakhutdinov, 2006). It is long known that *multilayer perceptrons* (MLP) are so prone to initialization, that it is virtually impossible to learn a *deep architecture* (i.e. many layers) with any random initialization thereof

[22] Samples of low (or no) importance would be ideally mapped onto the feature output of the *last* important sample seen in the trajectory. However, due to the structure of Hilbert space $\mathcal{H}$ the modified algorithm can produce mappings that generalize well in Θ.

[23] The feature output of the unimportant part of Θ should be either constant or change as smooth as possible between *adjacent* important samples.

(Haykin, 1999). Hinton and Osindero (2006) came up with the idea[24] to train the first layer as an *auto-encoder*, i.e. to predict its own inputs via a hidden layer of perceptrons. The trained auto-encoder is then fixed and the hidden layer used as the input of the next layer, which is trained in the same way. The resulting deep architecture is a generative model of the data and has proven to be a suitable initialization for many problems.

In line with this article's hypothesis, however, we assume that the target function is defined over a space of latent variables Θ. Layer-wise training of an MLP version of SFA might therefore yield even better initialization, as it already spans a suitable *function space* rather than simply encoding the data (see Franzius et al. (2007) for an example of a hierarchical SFA). However, repeated application of non-linear SFA, even with the limited function class of one perceptron per feature, bears sooner or later the problem of over-fitting. Preliminary experiments on repeated application of quadratic SFA lost any generalisation ability within a few layers. Without an automatic way to regularize the solution in each layer, this approach will thus not yield a reliable initialization for classical MLP backpropagation (Haykin, 1999). Future research must show whether a properly regularized MLP-SFA algorithm can be used to initialize an MLP and whether or not this approach can compete with deep belief networks based on auto-encoders.

Acknowledgements This work has been supported by the Integrated Graduate Program on Human-Centric Communication at Technische Universität Berlin, the German Research Foundation (DFG SPP 1527 *autonomous learning*), the German Federal Ministry of Education and Research (grant 01GQ0850) and EPSRC grant #EP/H017402/1 (CARDyAL). We want to thank Matthias Franzius, who gave us a sound introduction into non-linear SFA, and Roland Vollgraf for his contribution to an earlier version of RSK-SFA.

References

P.F. Assmann, T.M. Nearey, and S. Bharadwaj. Analysis and classification of a vowel database. In *Canadian Acoustics*, number 36(3), pages 148–149, 2008.

S. Becker and G.E. Hinton. A self-organizing neural network that discovers surfaces in randomdot stereograms. *Nature*, 355(6356):161–163, 1992.

P. Berkes and L. Wiskott. Slow feature analysis yields a rich repertoire of complex cell properties. *Journal of Vision*, 5:579–602, 2005.

Pietro Berkes. Pattern recognition with slow feature analysis. *Cognitive Sciences EPrint Archive (CogPrint)*, (4104), 2005.

C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer Verlag, 2006. ISBN 978-0-387-31073-2.

W. Böhmer, S. Grünewälder, H. Nickisch, and K. Obermayer. Regularizes sparse kernel slow feature analysis. In *ECML / PKDD 2011*, volume I, pages 235–248, 2011.

A. Bray and D. Martinez. Kernel-based extraction of Slow features: Complex cells learn disparity and translation invariance from natural images. *Neural Information Processing Systems*, 15:253–260, 2002.

L. Csató and M. Opper. Sparse on-line gaussian processes. *Neural Computation*, 14(3):641 – 668, 2002.

R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

W. Einhäuser, J. Hipp, J. Eggert, E. Körner, and P. König. Learning viewpoint invariant object representations using temporal coherence principle. *Biological Cybernetics*, 93(1):79–90, 2005.

R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

R. Fletcher. *Practical Methods of Optimization*. Wiley, 2nd edition, 1987.

P. Földiák. Learning invariance from transformation sequences. *Neural Computation*, 3(2):194–200, 1991.

M. Franzius, H. Sprekeler, and L. Wiskott. Slowness and sparseness leads to place, head-direction, and spatial-view cells. *PLoS Computational Biology*, 3(8):e166, 2007.

[24] Hinton and Osindero (2006) showed this for *restricted Boltzmann machines*, but the principle holds for MLP as well.

K. Fukumizu, F.R. Bach, and A. Gretton. Statistical consistency of kernel canonical correlation analysis. *Journal of Machine Learning Research*, 8:361–383, 2007.
S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice-Hall, 2nd edition, 1999.
G.E. Hinton and S. Osindero. A fast learning algorithm for deep belief nets. *Neural Computation*, 18: 1527–1554, 2006.
G.E. Hinton and R.R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.
J.P. Huke. Embedding nonlinear dynamical systems: A guide to takens' theorem. Technical report, University of Manchester, 2006.
Z. Hussain and J. Shawe-Taylor. Theory of matching pursuit. In *Advances in Neural Information Processing Systems 21*, pages 721–728, 2008.
S. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions On Signal Processing*, 41:3397–3415, 1993.
S.P. Meyn and R.L. Tweedie. *Markov chains and stochastic stability*. Springer-Verlag, London, 1993.
F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan, 1962.
D.B. Rubin. Iteratively reweighted least squares. *Encyclopedia of Statistical Sciences*, 4:272–275, 1983.
B. Schölkopf, A. Smola, and K.R. Müller. Kernel principal component analysis. In *Artificial Neural Networks ICANN*, 1997.
B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.
J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.
A.J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings to the 17th International Conference Machine Learning*, pages 911–918, 2000.
J.V. Stone. Blind source separation using temporal predictability. *Neural Computation*, 13(7):1559–1574, 2001.
F. Takens. Detecting strange attractors in turbulence. *Dynamical Systems and Turbulence*, pages 366–381, 1981.
G. Wahba. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, 1990.
L. Wiskott. Slow feature analysis: A theoretical analysis of optimal free responses. *Neural Computation*, 15(9):2147–2177, 2003.
L. Wiskott and T. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, 2002.
R. Wyss, P. König, and P.F.M.J. Verschure. A model of the ventral visual system based on temporal stability and local memory. *PLoS Biology*, 4(5):e120, 2006.

A.2 Böhmer et al., JMLR 14:2067–2118 (2013a)

The article *Construction of Approximation Spaces for Reinforcement Learning* (Böhmer et al., 2013) has been published in the **Journal of Machine Learning Research**[2] (JMLR) in July 2013.

The article investigates the theoretical properties of *slow feature analysis* (SFA) to learn state representations for linear *reinforcement learning* (RL) algorithms. We derive a connection between SFA features and bounds on the *approximation error* of linear value functions in RL. The analysis allows insights into *optimal* state representations and extends SFA to compensate for shortcomings of the original method. Our predictions are validated at discrete benchmark tasks and in a real robotic visual navigation task, which is also extensively evaluated in a realistic simulation.

The theoretical results of this article are discussed and extended in Section 4.2, whereas the empirical evaluation is summarized in Section 4.4. The notation follows mostly Chapter 4, with some notable differences in the interpretation of state and observation spaces:

- The sets $\mathcal{X}$ and $\mathcal{Z}$ are in the article *isomorphic* and primarily used to indicate the different metrics in observation space $\mathcal{Z}$ and an ideal state space $\mathcal{X}$.
- The steady state distribution in observation space $\mathcal{Z}$ is denoted ξ, not ζ as in Chapter 4.

The authors contributed in the following way:

- *Wendelin Böhmer* has written the entire article, has derived all theoretical results, has designed and performed all simulated experiments, and carried out the majority of the work for the robotic experiment.
- *Steffen Grünewälder* has designed the robotic experiment and gave comments to the article.
- *Yun Shen* constructed the robot-tracking system, used in the robotic experiment.
- *Marek Musial* constructed the low-level control system for the robotic platform used in the robotic experiment.
- *Klaus Obermayer* proposed the initial research question, supervised both development and experimentation, and gave comments to the article.

[2] The original article can be downloaded from `http://jmlr.org/papers/v14/boehmer13a.html` .

Journal of Machine Learning Research 14 (2013) 2067-2118 Submitted 7/12; Revised 4/13; Published 7/13

Construction of Approximation Spaces for Reinforcement Learning

Wendelin Böhmer WENDELIN@NI.TU-BERLIN.DE
Neural Information Processing Group
Technische Universität Berlin
Marchstrasse 23, Berlin 10587, Germany

Steffen Grünewälder STEFFEN@CS.UCL.AC.UK
Centre for Computational Statistics and Machine Learning
University College London
London WC1E 6BT, United Kingdom

Yun Shen YUN@NI.TU-BERLIN.DE
Neural Information Processing Group
Technische Universität Berlin
Marchstrasse 23, Berlin 10587, Germany

Marek Musial MUSIAL@CS.TU-BERLIN.DE
Robotics Group
Technische Universität Berlin
Einsteinufer17, Berlin 10587, Germany

Klaus Obermayer OBY@NI.TU-BERLIN.DE
Neural Information Processing Group
Technische Universität Berlin
Marchstrasse 23, Berlin 10587, Germany

Editor: Sridhar Mahadevan

Abstract

Linear *reinforcement learning* (RL) algorithms like *least-squares temporal difference learning* (LSTD) require *basis functions* that span *approximation spaces* of potential value functions. This article investigates methods to construct these bases from samples. We hypothesize that an ideal approximation spaces should encode *diffusion distances* and that *slow feature analysis* (SFA) constructs such spaces. To validate our hypothesis we provide theoretical statements about the LSTD value approximation error and induced metric of approximation spaces constructed by SFA and the state-of-the-art methods *Krylov bases* and *proto-value functions* (PVF). In particular, we prove that SFA minimizes the average (over all tasks in the same environment) bound on the above approximation error. Compared to other methods, SFA is very sensitive to sampling and can sometimes fail to encode the whole state space. We derive a novel *importance sampling* modification to compensate for this effect. Finally, the LSTD and *least squares policy iteration* (LSPI) performance of approximation spaces constructed by Krylov bases, PVF, SFA and PCA is compared in benchmark tasks and a visual robot navigation experiment (both in a realistic simulation and with a robot). The results support our hypothesis and suggest that (i) SFA provides *subspace-invariant* features for MDPs with *self-adjoint* transition operators, which allows strong guarantees on the approximation error, (ii) the modified SFA algorithm is best suited for LSPI in both discrete and continuous state spaces and (iii) approximation spaces encoding diffusion distances facilitate LSPI performance.

Keywords: reinforcement learning, diffusion distance, proto value functions, slow feature analysis, least-squares policy iteration, visual robot navigation

1. Introduction

Reinforcement learning (RL, Sutton and Barto, 1998; Bertsekas and Tsitsiklis, 1996) provides a framework to autonomously learn *control policies* in stochastic environments and has become popular in recent years for controlling robots (e.g., Abbeel et al., 2007; Kober and Peters, 2009). The goal of RL is to compute a policy which selects *actions* that maximize the *expected future reward* (called *value*). An agent has to make these decisions based on the *state* $\boldsymbol{x} \in \mathcal{X}$ of the system. The state space $\mathcal{X}$ may be finite or continuous, but is in many practical cases too large to be represented directly. *Approximated RL* addresses this by choosing a function from *function set* $\mathcal{F}$ that resembles the true value function. Many function sets $\mathcal{F}$ have been proposed (see, e.g., Sutton and Barto, 1998; Kaelbling et al., 1996, for an overview). This article will focus on the space of *linear functions* with p non-linear *basis functions* $\{\phi_i(\cdot)\}_{i=1}^p$ (Bertsekas, 2007), which we call *approximation space* $\mathcal{F}_\phi$.

The required basis functions $\phi_i(\cdot)$ are usually defined by hand (e.g., Sutton, 1996; Konidaris et al., 2011) and a bad choice can critically impede the accuracy of both the value estimate and the resulting control policy (see, e.g., Thrun and Schwartz, 1993). To address this issue, a growing body of literature has been devoted to the *construction* of basis functions and their theoretical properties (Mahadevan and Maggioni, 2007; Petrik, 2007; Parr et al., 2007; Mahadevan and Liu, 2010; Sun et al., 2011). Recently, the unsupervised method *slow feature analysis* (SFA, Wiskott and Sejnowski, 2002) has been proposed in this context (Legenstein et al., 2010; Luciw and Schmidhuber, 2012). This article presents a theoretical analysis of this technique and compares it with state-of-the-art methods. We provide theoretical statements for two major classes of automatically constructed basis functions (*reward-based* and *subspace-invariant* features, Parr et al., 2008) with respect to the induced Euclidean metric and the approximation error of *least-squares temporal difference learning* (LSTD, Bradtke and Barto, 1996). We also prove that under some assumptions SFA minimizes an average bound on the approximation error of all tasks in the same environment and argue that no better solution based on a single training sequence exists.

In practical applications (such as robotics) the state can not always be observed directly, but may be deducted from *observations*[1] $\boldsymbol{z} \in \mathcal{Z}$ of the environment. *Partial observable Markov decision processes* (POMDPs, Kaelbling et al., 1998) deal with the necessary inference of hidden states from observations. POMDPs are theoretically better suited, but become quickly infeasible for robotics. In contrast, this article focuses on another obstacle to value estimation: the *metric* associated with observation space $\mathcal{Z}$ can influence basis function construction. We assume for this purpose a unique one-to-one correspondence[2] between states $\boldsymbol{x} \in \mathcal{X}$ and observations $\boldsymbol{z} \in \mathcal{Z}$. To demonstrate the predicted effect we evaluate construction methods on a robotic visual navigation task. The observations are first-person perspective images, which exhibit a very different Euclidean metric than the underlying state of robot position and orientation. We hypothesize that *continuous SFA* (RSK-SFA, Böhmer et al., 2012) is not severely impeded by the change in observation metric and substantiate this in comparison to *continuous proto value functions* (PVF, Mahadevan and Maggioni, 2007) and *kernel PCA* (Schölkopf et al., 1998). We also confirm theoretical predictions that SFA is sensitive to the sampling policy (Franzius et al., 2007) and derive an *importance sampling* modification to compensate for these imbalances.

1. In this article the set of all possible observations $\mathcal{Z}$ is assumed to be a manifold in vector space $\mathbb{R}^d$.
2. This makes $\mathcal{Z}$ an isomorphism of $\mathcal{X}$, embedded in $\mathbb{R}^d$. The only difference is the associated *metric*. In the following we will continue to discriminate between $\mathcal{X}$ and $\mathcal{Z}$ for illustrative purposes.

1.1 Approximation Spaces

State spaces $\mathcal{X}$ can be finite or continuous.[3] We define the corresponding observation space $\mathcal{Z}$ to be isomorphic, but it may be governed by a very different *metric*. For example: finite $\mathcal{X}$ are usually equipped with a *discrete metric*, in which all states have equal distance to each other. Isomorphic observations $\boldsymbol{z} \in \mathcal{Z} \subset \mathbb{R}^d$, on the other hand, might be equipped with an Euclidean metric in $\mathbb{R}^d$ instead. To approximate the *value function* $V(\cdot) : \mathcal{Z} \to \mathbb{R}$, one aims for a function $f(\cdot) \in \mathcal{F}$ that minimizes the *approximation error* w.r.t. some norm $\|V - f\|$. This article is focusing on the Euclidean L_2 norm[4] (see Section 2 for details), which depends on the metric's distance function. Besides different approximation errors, *generalization* to unseen states will also be very different in these spaces. This raises the question which metric is best suited to approximate value functions. Values are defined as the expected sum of *future* rewards. States with similar futures will therefore have similar values and are thus close by under an ideal metric. *Diffusion distances* compare the probabilities to end up in the same states (see, e.g., Coifman et al., 2005, and Section 4.1). It sands therefore to reason that a diffusion metric facilitates value approximation.

This article is using the term *approximation space* $\mathcal{F}_{\boldsymbol{\phi}}$ for the set of linear functions with p non-linear basis functions $\phi_i : \mathcal{Z} \to \mathbb{R}$, $\mathcal{F}_{\boldsymbol{\phi}} := \left\{ f(\cdot) = \boldsymbol{w}^\top \boldsymbol{\phi}(\cdot) \,|\, \boldsymbol{w} \in \mathbb{R}^p \right\}$. Function approximation can be essentially performed by an inverse of the covariance matrix (see Section 2.3) and value estimation can be guaranteed to converge (Bertsekas, 2007). Nonetheless, the choice of basis functions $\boldsymbol{\phi} : \mathcal{Z} \to \mathbb{R}^p$ and thus approximation space $\mathcal{F}_{\boldsymbol{\phi}}$ will strongly affect approximation quality and generalization to unseen samples. An ideal approximation space should therefore (i) be able to approximate the value function well and (ii) be equipped with a Euclidean metric in $\{\boldsymbol{\phi}(\boldsymbol{z}) \,|\, \boldsymbol{z} \in \mathcal{Z}\}$ that resembles a diffusion metric. Approximation theory provides us with general functional bases that allow arbitrarily close approximation of continuous functions and thus fulfill (i), for example polynomials or a Fourier basis (Konidaris et al., 2011). However, those bases can usually not be defined on high-dimensional observation spaces $\mathcal{Z}$, as they are prone to the *curse of dimensionality*.

A straightforward approach to basis construction would extract a low-dimensional manifold of $\mathcal{Z}$ and construct a general function base on top of it. This can be achieved by manifold extraction (Tenenbaum et al., 2000; Jenkins and Mataric, 2004) or by computer vision techniques (e.g., Visual SLAM, Smith et al., 1990; Davison, 2003), which require extensive knowledge of the latent state space $\mathcal{X}$. Some approaches construct basis functions $\boldsymbol{\phi}(\cdot)$ directly on the observations $\boldsymbol{z} \in \mathcal{Z}$, but are either restricted to linear maps $\boldsymbol{\phi}(\cdot)$ (PP, Sprague, 2009) or do not generalize to unseen samples (ARE, Bowling et al., 2005). None of the above methods extracts $\mathcal{X}$ in a representation that encodes a diffusion metric.

Recent analysis of the approximation error has revealed two opposing approaches to basis construction: *reward-based* and *subspace-invariant* features (Parr et al., 2008). The former encode the propagated reward function and the latter aim for eigenvectors of the transition matrix. Section 3.1 provides an overview of the reward-based *Krylov bases* (Petrik, 2007), *Bellman error basis functions* (BEBF, Parr et al., 2007), *Bellman average reward bases* (BARB, Mahadevan and Liu, 2010) and *Value-function of the Bellman error* bases (V-BEBF, Sun et al., 2011). All of these algorithms are defined exclusively for finite state spaces. The encoded metric is investigated in Section 4.1. *Proto-value functions* (PVF, Mahadevan and Maggioni, 2007, and Section 3.3) are the state-of-the-

3. This article does not discuss discrete countable infinite state spaces, which are isomorphisms to $\mathbb{N}$.

4. Other approaches are based on the L_∞ norm (Guestrin et al., 2001; Petrik and Zilberstein, 2011) or the L_1 norm (de Farias and Roy, 2003). However, all norms eventually depend on the metric of $\mathcal{X}$ or $\mathcal{Z}$.

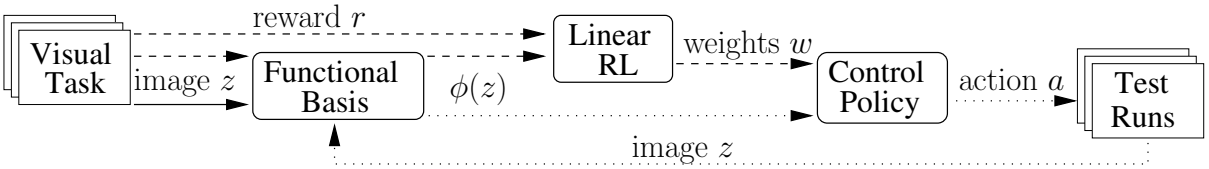


Figure 1: Scheme of a general RL architecture for visual tasks. First (solid arrow) one or many visual tasks generate images $\boldsymbol{z}$ to train a *representation* $\boldsymbol{\phi} : \mathcal{Z} \to \mathbb{R}^p$, which is a *functional basis* in the true state space $\mathcal{X}$. Given such a basis (dashed) one task is used to train a *control policy* with a *linear RL* algorithm. In the verification phase (dotted) the trained control policy generates multiple test trajectories.

art subspace-invariant feature construction method. In finite state spaces PVF are the eigenvectors to the smallest eigenvalues of the normalized graph Laplacian of an undirected graph representing the *transition possibility* (not *probability*) between states. As the calculation requires no knowledge of the reward, this technique has proven useful to transfer knowledge between different tasks in the same environment (Ferguson and Mahadevan, 2006; Ferrante et al., 2008). In Section 4.3 we will explain this observation by defining the class of learning problems for which this transfer is nearly optimal. To cope with continuous state or observation spaces, there also exists an extension based on the PVF of a k-nearest neighbors graph and Nyström approximation between the graph nodes (Mahadevan and Maggioni, 2007). However, as this approach is based on neighborhood relationships in $\mathcal{Z}$, the solution will not preserve diffusion distances.

An extension preserving these distances are Laplacian eigenmaps (Belkin and Niyogi, 2003) of the *transition operator*. Recently Sprekeler (2011) has shown that *slow feature analysis* (SFA, Wiskott and Sejnowski, 2002, and Section 3.4) approximates Laplacian eigenmaps. In the limit of an infinite training sequence, it can be shown (under mild assumptions) that the resulting non-linear SFA features span a Fourier basis in the unknown state space $\mathcal{X}$ (Wiskott, 2003). Franzius et al. (2007) show additionally that the *order* in which the basis functions are encoded is strongly dependent on the *relative velocities* in different state dimensions. This can lead to an insufficient approximation for *low dimensional*, but has little effect on *high dimensional* approximation spaces. Section 3.5 addresses this problem with an *importance sampling* modification to SFA.

1.2 Visual Tasks

Most benchmark tasks in RL have either a finite or a continuous state space with a well behaving Euclidean metric.[5] Theoretical statements in Section 4 predict that SFA encodes diffusion distances, which are supposed to facilitate generalization. Testing this hypothesis requires a task that can be solved either with a well behaving true state $\boldsymbol{x} \in \mathcal{X}$ or based on observations $\boldsymbol{z} \in \mathcal{Z} \subset \mathbb{R}^d$ with a disadvantageous metric. Value functions approximated w.r.t. a diffusion metric, that is, with SFA features, should provide comparable performance in both spaces. Based on a method that encodes only Euclidean distances in $\mathcal{Z}$ (e.g., PCA), on the other hand, the performance of the approximated value functions should differ.

5. An example for finite state spaces is the *50-state chain* (Section 5.2). The *puddle-world task* (Section 5.3) and the *mountain-car task* (not evaluated) have been defined with continuous and with discrete state spaces (Boyan and Moore, 1995; Sutton, 1996). Both continuous tasks are defined on well-scaled two dimensional state spaces. Euclidean distances in these spaces resemble diffusion distances closely.

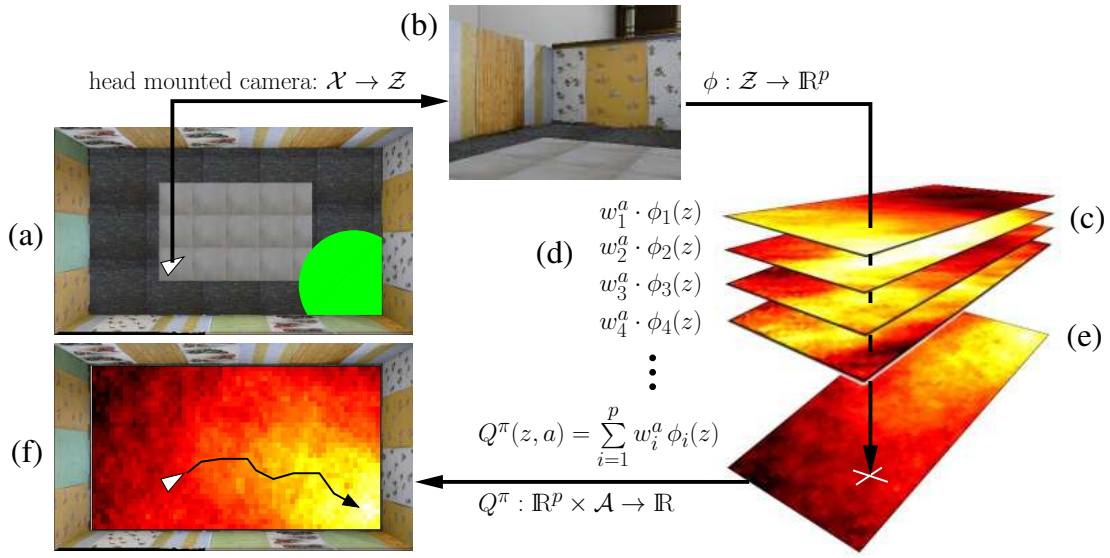


Figure 2: The figure shows the visual control process to guide a robot into the circular goal area. At its position (a), the robot observes an image with its head mounted camera (b). A function $\boldsymbol{\phi}(\cdot)$, generated by one of the discussed unsupervised methods, maps the image into a p-dimensional feature space (c). For each action $a \in \mathcal{A}$, these features are weighted by the LSPI parameter vector $\boldsymbol{w}^a \in \mathbb{R}^p$ (d), giving rise to the Q-value function $Q^{\boldsymbol{\pi}}(\cdot,\cdot)$ (e). The control always chooses the action a with the highest Q-value (f).

Applied to visual input, for example camera images, this class of problems is called *visual tasks*. Setting problems of *partial observability* aside, the true state $\boldsymbol{x}$ is usually assumed to be sufficiently *represented* by a set of hand-crafted features of $\boldsymbol{z}$. However, there is no straightforward way to extract the state reliably out of visual data without introducing artificial markers to the environment. Current approaches to visual tasks aim thus to learn a feature mapping $\boldsymbol{\phi} : \mathcal{Z} \to \mathbb{R}^p$ from observations $\boldsymbol{z}$, without loosing too much information about the true state $\boldsymbol{x}$ (see Jodogne and Piater, 2007, for an overview). Figure 1 shows a sketch of a general RL architecture to solve visual tasks with linear RL methods. Here we first learn an *image representation* $\boldsymbol{\phi} : \mathcal{Z} \to \mathbb{R}^p$ from experience (solid arrow), collected in one or more visual tasks within the same environment. To learn a *control policy* (dashed arrows) the agent treats the representation $\boldsymbol{\phi}(\boldsymbol{z}) \in \mathbb{R}^p$ of each observed image $\boldsymbol{z} \in \mathcal{Z}$ as the representation of the corresponding state $\boldsymbol{x} \in \mathcal{X}$. A linear RL algorithm can estimate future *rewards* $r \in \mathbb{R}$ by approximating the linear *Q-value function* $Q^{\boldsymbol{\pi}} : \mathbb{R}^p \times \mathcal{A} \to \mathbb{R}$ with *weight vector* $\boldsymbol{w} \in \mathbb{R}^{p|\mathcal{A}|}$. The control policy always chooses the *action* $a \in \mathcal{A}$ with the highest Q-value predicted by $Q^{\boldsymbol{\pi}}$ and can be verified by independent test runs from random start positions (dotted arrows). For example, in the context of navigation, Lange and Riedmiller (2010) employed a *deep auto-encoder* (Hinton and Osindero, 2006), Legenstein et al. (2010) hierarchical nonlinear *slow feature analysis* (SFA, Wiskott and Sejnowski, 2002) and Luciw and Schmidhuber (2012) *incremental SFA* (Kompella et al., 2012) to represent the underlying state space. The control problem was subsequently solved by different *approximate RL* algorithms. All above works verified

their approaches on a very regular observation space $\mathcal{Z}$ by providing the agent with a bird's eye view of an artificial world, in which a set of pixels determines the agents position uniquely.

To yield a less ideal observation space $\mathcal{Z}$, the visual navigation task in Section 5.4 observes first-person perspective images[6] instead. Figure 2 shows the control loop of the robot. The true state $\boldsymbol{x}$ is the robot's position at which an image $\boldsymbol{z}$ is taken by a head-mounted camera. $\mathcal{X}$ is continuous and in principle the actions $a \in \mathcal{A}$ should be continuous too. However, selecting continuous actions is not trivial and for the sake of simplicity we restricted the agent to three discrete actions: move forward and turn left or right.

1.3 Contributions

The contributions of this article are threefold:

1. We provide theoretical statements about the encoded *diffusion metric* (Section 4.1) and the LSTD value *approximation error* (Section 4.2) of both reward-based (Krylov bases) and subspace-invariant (SFA) features. We also prove that SFA minimizes an average bound on the approximation error of a particular set of tasks (Section 4.3). We conclude that *SFA can construct better approximation spaces for LSTD than PVF* and demonstrate this on multiple discrete benchmark tasks (Sections 5.1 to 5.3).

2. We investigate the role of the metric in approximation space $\mathcal{F}_\phi$ on a visual robot navigation experiment, both in a realistic simulation and on a robot (Sections 5.4 to 5.7). We demonstrate than SFA can sometimes fail to encode the whole state space due to its dependence on the sampling policy and address this problem with a novel *importance sampling* modification to the SFA algorithm.

3. We compare the performance of approximation spaces constructed by Krylov bases, PVF, SFA and PCA for *least-squares policy iteration* (LSPI, Lagoudakis and Parr, 2003). Results suggest that (i) the modified SFA algorithm is best suited for LSPI in both discrete and continuous state spaces and (ii) approximation spaces that encode a diffusion metric facilitate LSPI performance.

Both theoretical and empirical results leave room for interpretation and unresolved issues for future works. Section 6 discusses open questions as well as potential solutions. Finally, the main results and conclusions of this article are summarized in Section 7.

2. Reinforcement Learning

In this section we review *reinforcement learning* in potentially continuous state spaces $\mathcal{X}$, which require a slightly more complicated formalism than used in standard text books (e.g., Sutton and Barto, 1998). The introduced notation is necessary for Section 4 and the corresponding proofs in Appendix A. However, casual readers familiar with the RL problem can skip this section and still comprehend the more practical aspects of the article.

There exist many linear RL algorithms one could apply to our experiment, like *temporal difference learning* (TD(λ), Sutton and Barto, 1998) or *Q-learning* (Watkins and Dayan, 1992). We

6. Rotating a camera by some degrees represents only a minor change in its orientation and therefore in $\mathcal{X}$, but shifts all pixels and can lead to very large Euclidean distances in $\mathcal{Z}$. Moving slightly forward, on the other hand, changes only the pixels of objects close by and thus yields a much smaller distance in $\mathcal{Z}$.

chose here the *least-squares policy iteration* algorithm (LSPI, Lagoudakis and Parr, 2003, see Section 2.4), because the underlying *least-squares temporal difference* algorithm (LSTD, Bradtke and Barto, 1996, see Section 2.3) is the most sample effective unbiased value estimator (Grünewälder and Obermayer, 2011). For practical implementation we consider *sparse kernel methods*, which are introduced in Section 2.5.

2.1 The Reinforcement Learning Problem

We start with the definition of a *Markov decision process* (MDP). Let $\mathcal{B}(\mathcal{X})$ denote the collection of all *Borel sets* of set $\mathcal{X}$. A *Markov decision process* is a tuple $(\mathcal{X},\mathcal{A},P,R)$. In our setup, $\mathcal{X}$ is a *finite* or *compact continuous*[7] state space and $\mathcal{A}$ the finite[8] action space. The *transition kernel*[9] $P : \mathcal{X}\times\mathcal{A}\times\mathcal{B}(\mathcal{X}) \to [0,1]$ represents the probability $P(A|x,a)$ to end up in set $A \in \mathcal{B}(\mathcal{X})$ after executing action a in state x. $R : \mathcal{X}\times\mathcal{A}\times\mathcal{X}\times\mathcal{B}(\mathbb{R}) \to [0,1]$ is a distribution over *rewards*: $R(B|x,a,y)$ is the probability to receive a reward within set $B \in \mathcal{B}(\mathbb{R})$ after a transition from state x to state y, executing action a. In our context, however, we will be content with the mean *reward function* $r : \mathcal{X}\times\mathcal{A} \to \mathbb{R}$, defined as $r(x,a) = \int_{\mathbb{R}}\int_{\mathcal{X}} rR(dr|x,a,y)P(dy|x,a), \forall x \in \mathcal{X}, a \in \mathcal{A}$. A control *policy* $\pi : \mathcal{X}\times\mathcal{B}(\mathcal{A}) \to [0,1]$ is a conditional distribution of actions given states. The goal of *reinforcement learning* is to find a policy that maximizes the *value* $V^\pi(x)$ at each state x, that is the expected sum of discounted future rewards

$$V^\pi(x_0) := \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t\, r(x_t,a_t) \;\middle|\; \begin{array}{rcl} a_t & \sim & \pi(\cdot|x_t) \\ x_{t+1} & \sim & P(\cdot|x_t,a_t)\end{array}\right], \quad \forall x_0 \in \mathcal{X}.$$

Here the *discount factor* $\gamma \in [0,1)$ determines the relative importance of short term to long term rewards.[10] The value function can also be expressed *recursively*:

$$V^\pi(x) = \int r(x,a)\,\pi(da|x) + \gamma \iint V^\pi(y)\,P(dy|x,a)\,\pi(da|x), \quad \forall x \in \mathcal{X}.$$

In finite state (and action) spaces this equation can be solved by dynamic programming. Note that for fixed $V^\pi(\cdot)$ the equation is linear in the policy $\pi(\cdot|\cdot)$ and vice versa, allowing an *expectation maximization* type algorithm called *policy iteration* (PI, Sutton and Barto, 1998) to find the best policy. To allow for continuous state spaces, however, we need to translate this formalism into a Hilbert space.

2.2 MDP in Hilbert Spaces

For the sake of feasibility we will restrict our discussion to value functions $v^\pi \in L^2(\mathcal{X},\xi)$ from the space of *square-integrable functions* on $\mathcal{X}$, endowed with probability measure $\xi : \mathcal{B}(\mathcal{X}) \to [0,1]$, $\int\xi(dx) = 1$. This Hilbert space contains ξ-measurable functions and should suffice for all

7. Compact state spaces $\mathcal{X}$ are necessary for *ergodicity*, see Footnote 12 on Page 2074. All finite $\mathcal{X}$ are compact.
8. For generality we maintain the notation of continuous compact action spaces as long as possible.
9. Following probability theory, a *kernel* denotes here a *conditional measure* over some set, in this article $\mathcal{X}\times\mathcal{A}$ or just $\mathcal{X}$. If this measure over the whole set is always one then it is called a *transition* or *Markov kernel*. Note that the *Radon-Nikodym derivative* of a kernel w.r.t. the uniform measure is called a *kernel function* in integral calculus. Note also the difference to *positive semi-definite kernels* in the context of RKHS (see Section 3.2).
10. In classical decision theory, γ can be interpreted as the continuous version of a maximal search depth in the decision tree. Alternatively, one can see γ^t as shrinking certainty about predicted rewards.

continuous setups. The induced *inner product* and *norm* are

$$\langle f,g\rangle_\xi = \int f(x)\,g(x)\,\xi(dx) \quad \text{and} \quad \|f\|_\xi = \langle f,f\rangle_\xi^{1/2}, \quad \forall f,g \in L^2(\mathcal{X},\xi).$$

For a fixed policy π, this yields the *transition operator*[11] $\hat{P}^\pi : L^2(\mathcal{X},\xi) \to L^2(\mathcal{X},\xi)$,

$$\hat{P}^\pi[f](x) := \iint f(y)\,P(dy|x,a)\,\pi(da|x), \quad \forall x \in \mathcal{X}, \quad \forall f \in L^2(\mathcal{X},\xi).$$

The operator is called *ergodic* if every Markov chain sampled by the underlying transition kernel P and policy π is ergodic.[12] This is a convenient assumption as it implies the existence of a *steady state distribution* ξ, which we will use as measure of $L^2(\mathcal{X},\xi)$. This also implies

$$\xi(B) = \iint P(B|x,a)\,\pi(da|x)\,\xi(dx), \quad \forall B \in \mathcal{B}(\mathcal{X}).$$

Under the assumption that all rewards are *bounded*, that is, $|r(x,a)|^2 < \infty \Rightarrow \exists r^\pi \in L^2(\mathcal{X},\xi) : r^\pi(x) := \int r(x,a)\,\pi(da|x), \forall x \in \mathcal{X}, \forall a \in \mathcal{A}$, we can define the *Bellman operator* in $L^2(\mathcal{X},\xi)$

$$\hat{B}^\pi[f](x) := r^\pi(x) + \gamma \hat{P}^\pi[f](x), \quad \forall x \in \mathcal{X}, \quad \forall f \in L^2(\mathcal{X},\xi),$$

which performs *recursive value propagation*. This is of particular interest as one can show[13] that $\hat{B}^\pi[f]$ is a contract mapping in $\|\cdot\|_\xi$ and an infinite application starting from any function $f \in L^2(\mathcal{X},\xi)$ converges to the *true* value function $v^\pi \in L^2(\mathcal{X},\xi)$.

2.3 Least-squares Temporal Difference Learning

Infinitely many applications of the Bellman operator $\hat{B}^\pi[\cdot]$ are not feasible in practice. However, there exist an efficient solution if one restricts oneself to an approximation from $\mathcal{F}_\phi = \{f(\cdot) = w^\top \phi(\cdot) \,|\, w \in \mathbb{R}^p\} \subset L^2(\mathcal{X},\xi)$. For linearly independent basis functions $\phi_i \in L^2(\mathcal{X},\xi)$ the projection of any function $f \in L^2(\mathcal{X},\xi)$ into $\mathcal{F}_\phi$ w.r.t. norm $\|\cdot\|_\xi$ can be calculated by the linear *projection operator* $\hat{\Pi}_\xi^\phi : L^2(\mathcal{X},\xi) \to L^2(\mathcal{X},\xi)$,

$$\hat{\Pi}_\xi^\phi[f](x) := \sum_{j=1}^p \overbrace{\sum_{i=1}^p \langle f,\phi_i\rangle_\xi (\mathbf{C}^{-1})_{ij}}^{w_j \in \mathbf{R}} \phi_j(x), \quad C_{ij} := \langle \phi_i,\phi_j\rangle_\xi, \quad \forall x \in \mathcal{X}, \quad \forall f \in L^2(\mathcal{X},\xi).$$

Instead of infinitely many alternating applications of $\hat{B}^\pi$ and $\hat{\Pi}_\xi^\phi$, one can directly calculate the fixed point $f^\pi \in \mathcal{F}_\phi$ of the combined operator

$$f^\pi \overset{!}{=} \hat{\Pi}_\xi^\phi\left[\hat{B}^\pi[f^\pi]\right] \qquad \Rightarrow \qquad w^\pi = \Big(\underbrace{\langle \phi, \phi - \gamma \hat{P}^\pi[\phi]\rangle_\xi}_{\mathbf{A}^\pi \in \mathbf{R}^{p\times p}}\Big)^\dagger \underbrace{\langle \phi, r^\pi\rangle_\xi}_{b^\pi \in \mathbf{R}^p},$$

11. Every *kernel* $A : \mathcal{X} \times \mathcal{B}(\mathcal{X}) \to [0,\infty)$ induces a linear operator $\hat{A} : L^2(\mathcal{X},\xi) \to L^2(\mathcal{X},\xi)$, $\hat{A}[f](x) := \int A(dy|x)\,f(y), \forall x \in \mathcal{X}, \forall f \in L^2(\mathcal{X},\xi)$, which in this article bears the same name with a hat.

12. A Markov chain is called *ergodic* if it is *aperiodic* and *positive recurrent*: if there is a nonzero probability to break any periodic cycle and if any infinite sequence eventually must come arbitrarily close to every state $x \in \mathcal{X}$. This is a property of the transition kernel rather than the policy. If *one* policy that assigns a nonzero probability to each action yields ergodic Markov chains, then *every* such policy does. Of course this does not hold for deterministic policies.

13. In a straightforward extension of the argument for finite state spaces (Bertsekas, 2007, Chapter 6).

where $\boldsymbol{w}^{\boldsymbol{\pi}} \in \mathbb{R}^p$ denotes the corresponding parameter vector of fixed point $f^{\boldsymbol{\pi}}(x) = (\boldsymbol{w}^{\boldsymbol{\pi}})^\top \boldsymbol{\phi}(x)$, $\forall x \in \mathcal{X}$, $(\mathbf{A}^{\boldsymbol{\pi}})^\dagger$ denotes the *Moore-Penrose pseudo-inverse* of matrix $\mathbf{A}^{\boldsymbol{\pi}}$ and we wrote $(\langle \boldsymbol{\phi}, \boldsymbol{\phi} \rangle_\xi)_{ij} = \langle \phi_i, \phi_j \rangle_\xi$ for convenience.

The stochastic matrices $\mathbf{A}^{\boldsymbol{\pi}}$ and $\boldsymbol{b}^{\boldsymbol{\pi}}$ can be bias-free estimated given a set of transitions $\left\{ \boldsymbol{\phi}(x_t) \xrightarrow{r_t} \boldsymbol{\phi}(x'_t) \right\}_{t=1}^n$ of start states $x_t \sim \xi(\cdot)$, executed actions $a_t \sim \boldsymbol{\pi}(\cdot|x_t)$, corresponding successive states $x'_t \sim P(\cdot|x_t, a_t)$ and received rewards $r_t \sim R(\cdot|x_t, a_t, x'_t)$. The resulting algorithm is known as *least-squares temporal difference learning* (LSTD, Bradtke and Barto, 1996). It can be shown that it converges[14] in $\|\cdot\|_\xi$ norm (Bertsekas, 2007). Note that for this property the samples must be drawn from steady state distribution ξ and policy $\boldsymbol{\pi}$, usually by a long Markov chain executing $\boldsymbol{\pi}$.

Moreover, Tsitsiklis and Van Roy (1997) have proven that in $\|\cdot\|_\xi$ norm the error between true value function $v^{\boldsymbol{\pi}} \in L^2(\mathcal{X}, \xi)$ and approximation $f^{\boldsymbol{\pi}} \in \mathcal{F}_{\boldsymbol{\phi}}$ is bounded[15] by

$$\left\| v^{\boldsymbol{\pi}} - f^{\boldsymbol{\pi}} \right\|_\xi \leq \frac{1}{\sqrt{1-\gamma^2}} \left\| v^{\boldsymbol{\pi}} - \hat{\Pi}_\xi^{\boldsymbol{\phi}} \left[v^{\boldsymbol{\pi}} \right] \right\|_\xi .$$

In Section 4 we will improve upon this bound significantly for a special case of SFA features. We will also show that for a specific class of tasks the basis functions $\phi_i(\cdot)$ extracted by SFA minimize a mean bound on the right hand side of this equation, in other words minimize the mean approximation error over all considered tasks.

2.4 Least-squares Policy Iteration

Estimating the value function does not directly yield a control policy. This problem is tackled by *least-squares policy iteration* (LSPI, Lagoudakis and Parr, 2003), which alternates between *Q-value estimation* (the expectation step) and *policy improvement* (the maximization step). At iteration i with current policy $\boldsymbol{\pi}_i$, the *Q-value function* $Q^{\boldsymbol{\pi}_i} : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is defined as the value of state $x \in \mathcal{X}$ conditioned on the next action $a \in \mathcal{A}$:

$$Q^{\boldsymbol{\pi}_i}(x,a) := r(x,a) + \gamma \int V^{\boldsymbol{\pi}_i}(y) P(dy|x,a) = r(x,a) + \gamma \iint Q^{\boldsymbol{\pi}_i}(y,b) \, \boldsymbol{\pi}_i(db|y) P(dy|x,a) .$$

Note that *Q-value estimation* is equivalent to *value estimation* in the space of twice integrable functions over the space of state-action pairs $\mathcal{X} \times \mathcal{A}$ endowed with probability measure $\mu(B,A) := \int_B \boldsymbol{\pi}_i(A|x) \, \xi(dx)$, $\forall (B,A) \in \mathcal{B}(\mathcal{X}) \times \mathcal{B}(\mathcal{A})$, that is, $L^2(\mathcal{X} \times \mathcal{A}, \mu)$. The corresponding transition operator $\hat{P}_Q^{\boldsymbol{\pi}_i} : L^2(\mathcal{X} \times \mathcal{A}, \mu) \to L^2(\mathcal{X} \times \mathcal{A}, \mu)$ is

$$\hat{P}_Q^{\boldsymbol{\pi}_i}[f](x,a) := \iint f(y,b) \, \boldsymbol{\pi}_i(db|y) P(dy|x,a) , \quad \forall f \in L^2(\mathcal{X} \times \mathcal{A}, \mu) , \quad \forall x \in \mathcal{X} , \quad \forall a \in \mathcal{A} .$$

The greedy policy $\boldsymbol{\pi}_{i+1}$ in the i'th *policy improvement* step will for each state x draw one of the actions with the highest Q-value, that is, $a^{\boldsymbol{\pi}_i}(x) \sim \arg\max_{a \in \mathcal{A}} Q^{\boldsymbol{\pi}_i}(x,a)$, and stick with it:

$$\boldsymbol{\pi}_{i+1}(a|x) := \begin{cases} 1 & ,\text{if} \quad a = a^{\boldsymbol{\pi}_i}(x) \\ 0 & ,\text{else} \end{cases} , \quad \forall x \in \mathcal{X} , \quad \forall a \in \mathcal{A} .$$

14. In a straightforward extension of the argument for finite state spaces (Bertsekas, 2007, Chapter 6).
15. Besides this bound in the weighted L_2 norm there exists a multitude of bounds in L_∞ and sometimes L_1 norm. See Petrik and Zilberstein (2011) for a recent overview.

In finite state-action spaces, this procedure will provably converge to a policy that maximizes the value for all states (Kaelbling et al., 1996).

To cope with continuous state and/or action spaces, LSPI employs the LSTD algorithm to estimate approximated Q-value functions $f^{\boldsymbol{\pi}_i} \in \mathcal{F}_{\boldsymbol{\phi}}$, where the basis functions $\boldsymbol{\phi}_i \in L^2(\mathcal{X} \times \mathcal{A}, \boldsymbol{\mu})$ are defined over state-action pairs rather than states alone. In difference to value estimation, any experienced set of transitions $\{(x_t, a_t) \xrightarrow{r_t} x'_t\}_{t=1}^n$ yields the necessary information for Q-value estimation with arbitrary policies $\boldsymbol{\pi}_i$, in other words the LSTD training set

$$\left\{ \boldsymbol{\phi}(x_t, a_t) \xrightarrow{r_t} \int \boldsymbol{\phi}(x'_t, a)\, \boldsymbol{\pi}_i(da|x'_t) \right\}_{t=1}^n .$$

However, convergence guarantees hold *only* when $\boldsymbol{\mu}$ is the steady state distribution of $P_Q^{\boldsymbol{\pi}_i}$, which usually only holds in the first iteration. Although it can thus not be guaranteed, empirically LSPI fails only for large function spaces $\mathcal{F}_{\boldsymbol{\phi}}$ and $\boldsymbol{\gamma}$ close to 1. In Section 5, Figure 8, we demonstrate this at the example of well and poorly constructed basis functions.

The easiest way to encode p state-action pairs for finite action spaces $\mathcal{A}$ is to use an arbitrary $q := p/|\mathcal{A}|$ dimensional state encoding $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^q$ and to extend it by $\bar{\boldsymbol{\phi}}(x,a) := \boldsymbol{\phi}(x)\boldsymbol{e}_a^\top$, $\forall x \in \mathcal{X}, \forall a \in \mathcal{A}$, where $\boldsymbol{e}_a \in \mathbb{R}^{|\mathcal{A}|}$ is a column vector of length 1 which is 0 everywhere except in one dimension uniquely associated with action a. The resulting $q \times |\mathcal{A}|$ matrix $\bar{\boldsymbol{\phi}}(x,a)$ can be treated as set of p state-action basis functions.

2.5 Reproducing Kernel Hilbert Spaces

Although $L^2(\mathcal{X}, \boldsymbol{\xi})$ is a very powerful tool for analysis, it has proven problematic in machine learning (Wahba, 1990; Schölkopf and Smola, 2002). Many algorithms employ instead the well behaving *reproducing kernel Hilbert spaces* $\mathcal{H}_{\boldsymbol{\kappa}} \subset L^2(\mathcal{Z}, \boldsymbol{\xi})$ (RKHS, see, e.g., Schölkopf and Smola, 2002). A RKHS is induced by a *positive semi-definite kernel function* $\boldsymbol{\kappa} : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$; the set $\{\boldsymbol{\kappa}(\cdot, \boldsymbol{x}) \mid \boldsymbol{x} \in \mathcal{Z}\}$ is a full (but not orthonormal) basis of $\mathcal{H}_{\boldsymbol{\kappa}}$. The inner product of two kernel functions in $\mathcal{H}_{\boldsymbol{\kappa}}$ can be expressed as a kernel function itself. Take the example of the *Gaussian kernel* used in this article:

$$\langle \boldsymbol{\kappa}(\cdot, \boldsymbol{x}), \boldsymbol{\kappa}(\cdot, \boldsymbol{y}) \rangle_{\mathcal{H}_{\boldsymbol{\kappa}}} \quad = \quad \boldsymbol{\kappa}(\boldsymbol{x}, \boldsymbol{y}) \quad := \quad \exp\left(-\tfrac{1}{2\sigma^2} \|\boldsymbol{x} - \boldsymbol{y}\|_2^2\right), \qquad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{Z}.$$

Due to compactness of $\mathcal{Z}$, all continuous functions f in $L^2(\mathcal{Z}, \boldsymbol{\xi})$ can be approximated arbitrarily well in L_∞ (supremum) norm by functions from $\mathcal{H}_{\boldsymbol{\kappa}}$.

Naive implementation of the *kernel trick* with n observed samples $\{\boldsymbol{z}_t\}_{t=1}^n$ induces a computational complexity of $\mathcal{O}(n^3)$ and a memory complexity of $\mathcal{O}(n^2)$. For large n it can thus be necessary to look for approximate solutions in the subspace spanned by some *sparse subset* $\{\boldsymbol{s}_i\}_{i=1}^m \subset \{\boldsymbol{z}_t\}_{t=1}^n$, $m \ll n$, and thus $f(\cdot) = \sum_{i=1}^m \alpha_i \boldsymbol{\kappa}(\cdot, \boldsymbol{s}_i) \in \mathcal{H}_{\boldsymbol{\kappa}}$, $\boldsymbol{\alpha} \in \mathbb{R}^m$ (*projected process* matrix sparsification, Rasmussen and Williams, 2006). If subset and approximation space are chosen well, the LSTD solution $f^{\boldsymbol{\pi}} \in \mathcal{F}_{\boldsymbol{\phi}}$ can be approximated well too:

$$f^{\boldsymbol{\pi}} \quad \in \quad \underbrace{\mathcal{F}_{\boldsymbol{\phi}} \quad \subset \quad \mathcal{F}_{\{\boldsymbol{\kappa}(\cdot, \boldsymbol{s}_i)\}_{i=1}^m}}_{\text{approximation space}} \quad \underset{\substack{\uparrow \\ \text{subset selection}}}{\subset} \quad \underbrace{\mathcal{H}_{\boldsymbol{\kappa}} \quad \subset \quad L^2(\mathcal{Z}, \boldsymbol{\xi})}_{\text{all continuous functions}} .$$

However, finding a suitable subset is not trivial (Smola and Schölkopf, 2000; Csató and Opper, 2002). We employ the *matching pursuit for maximization of the affine hull* algorithm (MP-MAH,

Böhmer et al., 2012) to find a uniformly distributed subset. Section 5.5 empirically evaluates the effect of this choice.

A MDP with *discrete state space* $\mathcal{Z} := \{x_i\}_{i=1}^d$ can also be embedded as a RKHS. The kernel $\kappa(x_i, x_j) = \delta_{ij}$ induces the *discrete metric*, where δ_{ij} is the *Kronecker delta*. In this metric every state is an *open set* and thus Borel sets, integrals, ergodicity and all other concepts in this section can be extended to discrete state spaces. The discrete metric does not allow generalization to neighboring states, though. In this case the "sparse" subsets of the kernel algorithms discussed in Section 3 must contain *all* states, that is, $\{\boldsymbol{s}_i\}_{i=1}^m = \{x_i\}_{i=1}^d$, which restricts the formalism to *finite* or *compact continuous* state spaces.

3. Basis Function Construction

Solving an MDP with the methods discussed in Section 2 requires the projection into an approximation space $\mathcal{F}_\phi = \{f(\cdot) = \boldsymbol{w}^\top \boldsymbol{\phi}(\cdot) \,|\, \boldsymbol{w} \in \mathbb{R}^p\} \subset L^2(\mathcal{Z}, \xi)$. The discussed algorithms make it necessary to specify the involved basis functions $\phi_i(\cdot) \in L^2(\mathcal{Z}, \xi), \forall i \in \{1, \dots, p\}$, *before* training, though. As the true value function $v^\pi \in L^2(\mathcal{Z}, \xi)$ is initially unknown, it is not obvious how to pick a basis that will eventually approximate it well. Classical choices (like Fourier bases, Konidaris et al., 2011) are known to approximate *any* continuous function arbitrarily well in the limit case. However, if applied on high-dimensional observations, for example, $\boldsymbol{z} \in \mathbb{R}^d$, the number of required functions p scales exponentially with d. It would therefore be highly advantageous to exploit knowledge of task or observation space and *construct* a low dimensional basis.

In this context, recent works have revealed two diametrically opposed concepts (Parr et al., 2008). Expressed in the notation of Section 2, the *Bellman error* of the fixed point solution $f^\pi(\cdot) = (\boldsymbol{w}^\pi)^\top \boldsymbol{\phi}(\cdot) \overset{!}{=} \hat{\Pi}_\xi^\phi [\hat{B}^\pi [f^\pi]] \in \mathcal{F}_\phi$ can be separated into two types of error functions,

$$\hat{B}^\pi[f^\pi] - f^\pi \quad = \quad \underbrace{\left(\hat{I} - \hat{\Pi}_\xi^\phi\right)[r^\pi]}_{\Delta^r \in L^2(\mathcal{Z},\xi)} \quad + \quad \gamma \sum_{i=1}^p w_i^\pi \underbrace{\left(\hat{I} - \hat{\Pi}_\xi^\phi\right)\left[\hat{P}^\pi[\phi_i]\right]}_{\Delta_i^\phi \in L^2(\mathcal{Z},\xi)},$$

the *reward error* $\Delta^r \in L^2(\mathcal{Z}, \xi)$ and the *per-feature errors* $\Delta_i^\phi \in L^2(\mathcal{Z}, \xi)$. Correspondingly, there have been two opposing approaches to basis function construction in literature:

1. *Reward-based features* encode the reward function and how it propagates in time. Δ^r is thus zero everywhere, but Δ_p^ϕ can still induce Bellman errors.

2. *Subspace-invariant features* aim for eigenfunctions of transition operator $\hat{P}^\pi$ to achieve no per-feature errors. Δ^r, however, can still induce Bellman errors.

This article focuses on subspace-invariant feature sets, but reward-based features are introduced for comparison in Section 3.1. As a baseline which encodes distances in $\mathcal{Z}$ but does not attempt subspace invariance, we introduce *principal component analysis* (PCA, Section 3.2). We continue with *proto value functions* (PVF, Section 3.3), which are the current state of the art in subspace-invariant features. *Slow feature analysis* (SFA, Section 3.4) has only recently been proposed to generate basis functions for RL. Section 4 analyzes the properties of both reward-based and subspace-invariant features in detail and Section 5 empirically compares all discussed algorithms in various experiments.

3.1 Reward-based Basis Functions

The true value function $v^\pi \in L^2(\mathcal{Z},\xi)$ is defined as $v^\pi(z) = \sum_{t=0}^\infty \gamma^t (\hat{P}^\pi)^t[r](z), \forall z \in \mathcal{Z}$, where $(\hat{P}^\pi)^t$ refers to t consecutive applications of operator $\hat{P}^\pi$ and $(\hat{P}^\pi)^0 := \hat{I}$. This illustrates the intuition of *Krylov bases* (Petrik, 2007):

$$\phi_i^K := (\hat{P}^\pi)^{i-1}[r] \in L^2(\mathcal{Z},\xi), \qquad \Phi_k^K := \{\phi_1^K, \ldots, \phi_k^K\}.$$

These bases are natural for *value iteration*, as the value functions of all iterations can be exactly represented (see also Corollary 10, Page 2087). However, the transition operator must be approximated from observations and the resulting basis Φ_k^K is not orthonormal. If employed, a projection operator must thus compute an expensive inverse (see Section 2.3). *Bellman error basis functions* (BEBF, Parr et al., 2007) rectify this by defining the $(k+1)$'th feature as the Bellman error of the fixed point solution $f^k \in \mathcal{F}_{\Phi_k^B}$ with k features:

$$\phi_{k+1}^B := \hat{B}^\pi[f^k] - f^k \in L^2(\mathcal{Z},\xi), \quad f^k \overset{!}{=} \hat{\Pi}_\xi^{\Phi_k^B}\left[\hat{B}^\pi[f^k]\right] \in \mathcal{F}_{\Phi_k^B}, \quad \Phi_k^B := \{\phi_1^B, \ldots, \phi_k^B\}.$$

BEBF are orthogonal, that is, $\langle \phi_i^B, \phi_j^B \rangle_\xi = \varepsilon \delta_{ij}, \varepsilon > 0$, and scaling ε to 1 yields an orthonormal basis. Parr et al. (2007) have shown that Krylov bases and BEBF span the same approximation space $\mathcal{F}_{\Phi_k^K} = \mathcal{F}_{\Phi_k^B}$. Both approaches require many features if $\gamma \to 1$.

Mahadevan and Liu (2010) have extended BEBF to *Bellman average reward bases* (BARB) by including the *average reward* ρ as the first feature. This is motivated by *Drazin bases* and has been reported to reduce the number of required features for large γ. Recently, Sun et al. (2011) have pointed out that given some basis Φ_k, the *best* $k+1$'th basis function is always the fixed point solution with the current Bellman error, that is, the next BEBF ϕ_{k+1}^B, as reward. Adding the resulting *Value function of the Bellman error* (V-BEBF) to the current basis can represent the true value exactly. However, the approach has to be combined with some feature selection strategy, as finding the V-BEBF fixed point is just as hard.

All above algorithms are exclusively defined on discrete MDPs. Although an extension to general RKHSs seems possible, it is not the focus of this article. However, to give readers a comparison of available methods we will evaluate orthogonalized Krylov bases (which are equivalent to BEBF) on discrete Benchmark tasks in Sections 5.2 and 5.3.

3.2 Principal Component Analysis

To provide a baseline for comparison, we introduce *principal component analysis* (PCA, Pearson, 1901). As PCA does not take any transitions into account, the extracted features must therefore encode Euclidean distances in $\mathcal{Z}$. PCA aims to find subspaces of maximal variance, which are spanned by the eigenvectors to the *p largest* eigenvalues of the data covariance matrix. One interpretation of PCA features $\phi : \mathcal{Z} \to \mathbb{R}^p$ is an optimal encoding of the centered data $\{z_t\}_{t=1}^n \subset \mathcal{Z} \subset \mathbb{R}^d$ w.r.t. linear *least-squares* reconstruction, that is the optimization problem

$$\inf_{\phi \in (\mathcal{F}_{\text{lin}})^p} \underbrace{\inf_{f \in (\mathcal{F}_\phi)^d} \tilde{\mathbb{E}}_t\left[\left\|z_t - f(z_t)\right\|_2^2\right]}_{\text{least-squares reconstruction in } \mathcal{F}_\phi},$$

where $\tilde{\mathbb{E}}_t[\cdot]$ is the empirical expectation operator w.r.t. all indices t and $\mathcal{F}_{\text{lin}}$ the set of linear functions in $\mathbb{R}^d$.

In the interest of a more efficient encoding one can extend the function set $\mathcal{F}_{\text{lin}}$. A popular example are *reproducing kernel Hilbert spaces*, introduced in Section 2.5. The resulting algorithm is called *kernel PCA* (Schölkopf et al., 1998) and performs an eigenvalue decomposition of the centered *kernel matrix* $K_{ij} = \kappa(z_i, z_j)$. The eigenvectors $v^i \in \mathbb{R}^n$ to the p largest eigenvalues are the coefficients to the feature maps:

$$\phi_i(z) := \sum_{t=1}^{n} v_t^i \, \kappa(z, z_t), \quad \forall z \in \mathcal{Z}.$$

The classical algorithm (Schölkopf et al., 1998) is severely limited by a *computational complexity* of $\mathcal{O}(n^3)$ and a *memory complexity* of $\mathcal{O}(n^2)$. It can thus be necessary to approximate the solution by using a *sparse kernel matrix* of a subset of the data (*projected process*, Rasmussen and Williams, 2006), that is, $K_{it} = \kappa(s_i, z_t)$, with $\{s_i\}_{i=1}^m \subset \{z_t\}_{t=1}^n$, $m \ll n$. The eigenvectors $v^i \in \mathbb{R}^m$ of $\frac{1}{n}\mathbf{K}\mathbf{K}^\top$ determine the coefficients of the *sparse kernel PCA* features. If a large enough subset is distributed uniformly in $\mathcal{Z}$, the approximation is usually very good.

3.3 Proto-value Functions

In finite state spaces $\mathcal{Z}$, $|\mathcal{Z}| < \infty$, proto-value functions (PVF, Mahadevan and Maggioni, 2007) are motivated by *diffusion maps* on graphs (Coifman et al., 2005). For this purpose a *connection graph* is constructed out of a Markov chain $\{z_t\}_{t=1}^n \subset \mathcal{Z}$: for the first observed transition from state x to y, the corresponding entry of connection matrix $\mathbf{W}$ is set $W_{xy} := 1$. All entries of non-observed transitions are set to zero. As diffusion maps require undirected graphs, this matrix must be symmetrized by setting $\mathbf{W} \leftarrow \frac{1}{2}(\mathbf{W} + \mathbf{W}^\top)$. PVF are the eigenvectors to the p smallest eigenvalues of the *normalized graph Laplacian* $\mathbf{L} := \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{W})\,\mathbf{D}^{-1/2}$, where $D_{xy} = \delta_{xy} \sum_{z=1}^{|\mathcal{Z}|} W_{xz}, \forall x, y \in \mathcal{Z}$, and δ_{xy} is the Kronecker delta. Section 4.1 shows that this approach, also known as *spectral encoding* (Belkin and Niyogi, 2003), yields approximation spaces in which Euclidean distances are equivalent to diffusion distances on the connection graph. Note, however, that these are not exactly the diffusion distances of the transition kernel, as the transition *possibility* rather than *probability* is encoded in matrix $\mathbf{W}$. Section 4.2 discusses this difference.

For infinite observation spaces $\mathcal{Z}$ PVF are also defined by connection graphs. However, in difference to the finite case, the construction of this graph is not straightforward. Mahadevan and Maggioni (2007) proposed a symmetrized *k-nearest neighbors graph* $\mathbf{W}$ out of a random[16] set $\{s_j\}_{j=1}^m \subset \{z_t\}_{t=1}^n$, $m \ll n$. Each node s_i is only connected with the k nearest nodes $\{s'_j\}_{j=1}^k \subset \{s_j\}_{j=1}^m$ (w.r.t. the Euclidean norm in $\mathcal{Z}$), with weights determined by a Gaussian kernel $\kappa(\cdot,\cdot)$ with width-parameter σ,

$$W_{ij} := \kappa(s_i, s_j) = \exp\left(-\tfrac{1}{2\sigma^2}\|s_i - s_j\|_2^2\right).$$

After symmetrization the PVF $\hat{\phi}_i$ at the nodes s_j are calculated. A Nyström extension approximates the PVF for all samples z by calculating the mean over the weighted PVF of the k nodes $\{s'_j\}_{j=1}^k \subset \{s_j\}_{j=1}^m$ closest to z,

$$\phi_i(z) := \sum_{j=1}^{k} \frac{\kappa(z, s'_j)}{\sum_{l=1}^{k} \kappa(z, s'_l)} \, \hat{\phi}_i(s'_j), \quad \forall z \in \mathcal{Z}.$$

16. Ideally the nodes are uniformly drawn w.r.t. the true diffusion metric, in other words uniformly in $\mathcal{X}$. If nodes are drawn randomly or uniformly in $\mathcal{Z}$, this difference can lead to a significant deviation in the number of transitions between nodes and the resulting diffusion distances thus deviate as well.

Note that these features are no longer based on the *transitions* of the observed Markov chain, but on Euclidean distances in $\mathcal{Z}$.

3.4 Slow Feature Analysis

The unsupervised learning method *slow feature analysis* (SFA, Wiskott and Sejnowski, 2002) aims for a set of mappings $\boldsymbol{\phi} : \mathcal{Z} \to \mathbb{R}^p$ such that the values $\boldsymbol{\phi}_i(\boldsymbol{z}_t)$ change slowly over an observed Markov chain $\{\boldsymbol{z}_t\}_{t=1}^n \subset \mathcal{Z}$. The objective (called *slowness* $\mathcal{S}$) is defined as the *expectation of the squared discrete temporal derivative*:

$$\inf_{\boldsymbol{\phi}\in(\mathcal{F})^p} \quad \sum_{i=1}^p \mathcal{S}(\boldsymbol{\phi}_i) \quad := \quad \sum_{i=1}^p \tilde{\mathbb{E}}_t[\dot{\boldsymbol{\phi}}_i^2(\boldsymbol{z}_t)] \quad \text{(slowness)}\,.$$

To ensure each slow feature encodes unique information and can be calculated in an iterative fashion, the following constraints must hold $\forall i \in \{1,\dots,p\}$:

$$\begin{aligned} \tilde{\mathbb{E}}_t[\boldsymbol{\phi}_i(\boldsymbol{z}_t)] &= 0 && \text{(zero mean)},\\ \tilde{\mathbb{E}}_t[\boldsymbol{\phi}_i^2(\boldsymbol{z}_t)] &= 1 && \text{(unit variance)},\\ \forall j \neq i : \tilde{\mathbb{E}}_t[\boldsymbol{\phi}_i(\boldsymbol{z}_t)\boldsymbol{\phi}_j(\boldsymbol{z}_t)] &= 0 && \text{(decorrelation)},\\ \forall j > i : \mathcal{S}(\boldsymbol{\phi}_i) &\leq \mathcal{S}(\boldsymbol{\phi}_j) && \text{(order)}\,. \end{aligned}$$

The principle of slowness has been used for a long time in the context of neural networks (Földiák, 1991). Kompella et al. (2012) have proposed an incremental online SFA algorithm. Recently several groups have attempted to use SFA on a random walk of observations to generate basis functions for RL (Legenstein et al., 2010; Luciw and Schmidhuber, 2012).

Although formulated as a linear algorithm, SFA was originally intended to be applied on the space of polynomials like quadratic (Wiskott and Sejnowski, 2002) or cubic (Berkes and Wiskott, 2005). The polynomial expansion of potentially high dimensional data, however, spans an impractically large space of coefficients. Hierarchical application of quadratic SFA has been proposed to solve this problem (Wiskott and Sejnowski, 2002; Legenstein et al., 2010). Although proven to work in complex tasks (Franzius et al., 2007), this approach involves a multitude of hyper-parameters and no easy way to counteract inevitable over-fitting is known so far.

An alternative to polynomial expansions are *sparse kernel methods* (see Section 2.5). We summarize in the following the *regularized sparse kernel SFA* (RSK-SFA, Böhmer et al., 2012) which we have used in our experiments. For a given sparse subset $\{\boldsymbol{s}_i\}_{i=1}^m \subset \{\boldsymbol{z}_t\}_{t=1}^n$, the algorithm determines the mapping $\boldsymbol{\phi} : \mathcal{Z} \to \mathbb{R}^p$ in 3 steps:

i Fulfilling the zero mean constraint directly on sparse kernel matrix $K_{it} := \boldsymbol{\kappa}(\boldsymbol{s}_i, \boldsymbol{z}_t)$, that is, $\mathbf{K}' := (\mathbf{I} - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^\top)\mathbf{K}(\mathbf{I} - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top)$, where $\mathbf{1}_k \in \mathbb{R}^k$ is a column vector of ones.

ii Fulfilling unit variance and decorrelation constraints by performing an eigenvalue decomposition $\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top := \frac{1}{n}\mathbf{K}'\mathbf{K}'^\top$ and projecting $\mathbf{K}'' := \boldsymbol{\Lambda}^{-\frac{1}{2}}\mathbf{U}^\top\mathbf{K}'$.

iii Minimize the objective by another eigenvalue decomposition $\mathbf{R}\boldsymbol{\Sigma}\mathbf{R}^\top := \frac{1}{n-1}\dot{\mathbf{K}}''\dot{\mathbf{K}}''^\top$, where $\dot{K}''_{it} := K''_{it+1} - K''_{it}$. Grouping the kernel functions of the sparse subset into one multivariate function $\boldsymbol{k} : \mathcal{Z} \to \mathbb{R}^m$ with $k_i(\boldsymbol{z}) := \boldsymbol{\kappa}(\boldsymbol{z}, \boldsymbol{s}_i)$, $\forall \boldsymbol{z} \in \mathcal{Z}$, the solution is given by

$$\boldsymbol{\phi}(\boldsymbol{z}) = \mathbf{A}^\top\boldsymbol{k}(\boldsymbol{z}) - \boldsymbol{c}, \quad \forall \boldsymbol{z} \in \mathcal{Z}$$
$$\text{with } \mathbf{A} := (\mathbf{I} - \mathbf{1}_m\mathbf{1}_m^\top)\mathbf{U}\boldsymbol{\Lambda}^{-\frac{1}{2}}\mathbf{R}, \text{ and } \boldsymbol{c} := \tfrac{1}{n}\mathbf{A}\mathbf{K}\mathbf{1}_n\,.$$

Böhmer et al. (2012) have demonstrated the numerical instability of this algorithm in face of insufficient sparseness and introduced a *regularization term* $\|\phi_i\|_{\mathcal{H}_\kappa}$ to the objective to stabilize the solution. In our experiments we did not face this problem, and regularization is thus omitted here.

3.5 Relative Velocities and Modified SFA

In the limit of an infinite Markov chain in $\mathcal{Z}$ and some mild assumptions[17] on the transition kernel in $\mathcal{X}$, the slowest possible mappings can be calculated analytically (Wiskott, 2003; Franzius et al., 2007). As the discrete temporal derivative is specified by the transition kernel in $\mathcal{X}$, the analytical solutions have domain $\mathcal{X}$ as well. Note, however, that the same transition kernel yields the same feature maps in $\mathcal{X}$, independent[18] of the actual observation space $\mathcal{Z}$. Section 4.1 demonstrates that these solutions are endowed with the diffusion metric of the symmetrized transition kernel, not unlike PVF in finite state spaces. Sprekeler (2011) has recently shown that in this case SFA solutions are (almost) equivalent to PVF. Note also that SFA encodes the actual transition *probabilities*, which requires more samples to converge than the transition *possibilities* encoded by PVF.

The analytically derived SFA features of Franzius et al. (2007) are of particular interest to the visual navigation experiment (Section 5.4 and Figure 2, Page 2071), as they assume the same setup. The solution is a *Fourier basis* on domain $\mathcal{X} := [0, L_x] \times [0, L_y] \times [0, 2\pi)$,

$$\phi_{\iota(i,j,l)}(x,y,\theta) = \begin{cases} \sqrt[3]{2}\cos(\frac{i\pi}{L_x}x)\cos(\frac{j\pi}{L_y}y)\sin(\frac{l+1}{2}\theta), & l \text{ odd} \\ \sqrt[3]{2}\cos(\frac{i\pi}{L_x}x)\cos(\frac{j\pi}{L_y}y)\cos(\frac{l}{2}\theta), & l \text{ even} \end{cases} \quad , \quad \forall (x,y,\theta) \in \mathcal{X} \,,$$

where $\iota : (\mathbb{N} \times \mathbb{N} \times \mathbb{N} \setminus \{(0,0,0)\}) \to \mathbb{N}^+$ is an index function, which depends on the *relative velocities* in two spatial dimensions x and y, and the robot's orientation θ. It can occur that SFA features have the same slowness, in which case the solution is no longer unique. For example, if $\phi_{\iota(1,0,0)}$ and $\phi_{\iota(0,1,0)}$ have the same slowness, then $\mathcal{S}(\phi_{\iota(1,0,0)}) = \mathcal{S}(\phi_{\iota(0,1,0)}) = \mathcal{S}(a\phi_{\iota(1,0,0)} + b\phi_{\iota(0,1,0)})$ holds as long as $a^2 + b^2 = 1$. This corresponds to an arbitrary rotation in the subspace of equally slow features. However, if we are interested in the space spanned by all features *up to a certain slowness*, every rotated solution spans the same approximation space $\mathcal{F}_\phi$.

The order $\iota(\cdot,\cdot,\cdot)$ of the analytical SFA features derived by Franzius et al. (2007, see above) depend strongly on the *relative velocities* in the state dimensions. For example, crossing the room in our experiment in Section 5.4 requires 10 movements, during which feature $\phi_{\iota(1,0,0)}$ will run through half a cosine wave. In as little as 4 rotations, on the other hand, feature $\phi_{\iota(0,0,1)}$ registers the same amount of change. Sampled evenly by a random policy, the first SFA features will therefore *not* encode the robot's orientation, which can critically impair the value representation in low dimensional approximation spaces. This article proposes a simple modification to the RSK-SFA algorithm to adjust the relative velocities by means of *importance sampling*.

Let $\{(z_t, a_t)\}_{t=0}^n$ denote a training sequence sampled by policy π with a steady state distribution ξ, which induces the joint distribution $\mu(B,A) = \int_B \pi(A|z)\,\xi(dz), \forall B \in \mathcal{B}(\mathcal{Z}), \forall A \in \mathcal{B}(\mathcal{A})$. To switch to another policy τ and state distribution ζ, that is, the joint distribution $\eta(B,A) = \int_B \tau(A|z)\,\zeta(dz), \forall B \in \mathcal{B}(\mathcal{Z}), \forall A \in \mathcal{B}(\mathcal{A})$, one can weight each transition with the *Radon-Nikodym*

17. In this case decorrelated Brownian motion in a multivariate state space $\mathcal{X}$ with independent boundary conditions for each dimension. Examples are rectangles, cubes, tori or spheres of real coordinates.

18. In line with SFA literature, this article does not discuss *partial observability* of the state. In other words, we assume there exist an *unknown* one-to-one mapping of states $x \in \mathcal{X}$ to observations $z \in \mathcal{Z}$.

derivative $\frac{d\eta}{d\mu}$. This yields the modified SFA optimization problem

$$\begin{aligned} \inf_{\phi\in(\mathcal{F})^p} \quad & \sum_{i=1}^{p} \hat{\mathbb{S}}(\phi_i,\eta) \quad := \quad \sum_{i=1}^{p} \tilde{\mathbb{E}}_t\left[\tfrac{d\eta}{d\mu}(z_t,a_t)\,\dot{\phi}_i^2(z_t)\right] \\ \text{s.t.} \quad & \begin{aligned} \tilde{\mathbb{E}}_t\left[\tfrac{d\eta}{d\mu}(z_t,a_t)\,\phi_i(z_t)\right] &= 0 \\ \tilde{\mathbb{E}}_t\left[\tfrac{d\eta}{d\mu}(z_t,a_t)\,\phi_i(z_t)\,\phi_j(z_t)\right] &= \delta_{ij} \end{aligned} \quad , \quad \forall i,j\in\{1,\ldots,p\}\,. \end{aligned}$$

However, there is no indication which distribution ζ and policy τ ensure a balanced encoding.

We propose here a simple heuristic for situations in which the actions affect only mutually independent subspaces of $\mathcal{X}$. In our robot navigation experiment, for example, rotations do not influence the robot's spatial position nor do the movements influence it's orientation. As optimal SFA features in the spatial subspace are significantly slower (see above), the first features will encode this subspace exclusively. This can be counteracted by setting $\zeta := \xi$ and defining $\frac{d\tau}{d\pi}(z,a) := \vartheta(a), \forall z\in\mathcal{Z}, \forall a\in\mathcal{A}$, where $\vartheta : \mathcal{A}\to\mathbb{R}^+$ weights each action independent of the current state. In practice, weights $\vartheta(a)$ need to be adjusted by hand for each action $a\in\mathcal{A}$: the *higher* $\vartheta(a)$, the *weaker* the influence of the corresponding subspace of $\mathcal{X}$ onto the first features. Only the last step (iii) of RSK-SFA has to be modified by redefining $\dot{K}''_{it} \leftarrow \vartheta^{\frac{1}{2}}(a_t)\,\dot{K}''_{it}$. Figure 9, Page 2101, demonstrates the effect of this modification.

4. Theoretical Analysis

This section analyzes the theoretical properties of *reward-based* and *subspace-invariant* features w.r.t. value function approximation. The employed formalism is introduced in Section 2. If not stated otherwise, the features are assumed to be optimized over $L^2(\mathcal{Z},\xi)$ and based on an infinite ergodic Markov chain. Proofs to all given lemmas and theorems can be found in Appendix A.

At the heart of function approximation lies the concept of *similarity*. Similar states will have similar function values. Usually this similarity is given by a metric on the observation space. Deviations of the function output from this metric must be compensated by the optimization algorithm. However, value function approximation allows for *explicit* specification of the required similarity. The definition of the value assigns similar function output to states with (i) similar immediate rewards and (ii) similar futures. As discussed in Section 3, *reward-based features* focus on encoding (i), whereas *subspace-invariant features* focus on (ii). Section 4.1 analyzes how SFA encodes similar futures as *diffusion distances* and shows some restrictions imposed onto the class of subspace-invariant features. The ramifications of these restrictions onto value function approximation are discussed in Section 4.2.

This article also presents a second, novel perspective onto value function approximation. The MDP one will face is usually not known before learning and the construction of a suitable basis is very expensive. Instead of approximating a particular MDP at hand, one could focus on a complete set $\mathcal{M}$ of *anticipated* MDPs. An *optimal* approximation space should be able to approximate any MDP $m\in\mathcal{M}$ if encountered. In difference to the classical analysis put forward by Petrik (2007), Parr et al. (2007) and Mahadevan and Maggioni (2007), this point of view puts emphasis on the reuse of prior experience, as investigated in *transfer learning* (Taylor and Stone, 2009; Ferguson and Mahadevan, 2006; Ferrante et al., 2008). Section 4.3 defines a criterion of *optimal features* for some anticipated set $\mathcal{M}$. Under some assumptions on $\mathcal{M}$, we prove that SFA optimizes a bound on

this criterion and argue that there can be no better bound based on a single Markov chain. Section 4.4 provides a summarizing conclusion and further implications can be found in Section 6.

4.1 Diffusion Metric

Values of observed states $x, y \in \mathcal{Z}$ depend less on their Euclidean distance in $\mathcal{Z}$ than on common *future states*. PVF are thus based on *diffusion distances* $d_t(x,y)$ of a graph representing the symmetrized *transition possibilities* $T_{xy} := W_{xy}/(\sum_{z\in\mathcal{Z}} W_{xz})$ between discrete states (see Section 3.3 or Mahadevan and Maggioni, 2007):

$$d_t^2(x,y) = \sum_{z\in\mathcal{Z}} \xi_z \left((\mathbf{T}^t)_{xz} - (\mathbf{T}^t)_{yz} \right)^2 ,$$

where $\xi \in \mathbb{R}^{|\mathcal{Z}|}$ are arbitrary[19] non-negative weights and $\mathbf{T}^t$ denotes the t'th power of matrix $\mathbf{T}$. These diffusion distances are equal to Euclidean distances in a space spanned by the eigenvectors $\phi_i \in \mathbb{R}^{|\mathcal{Z}|}$ and eigenvalues $\lambda_i \in \mathbb{R}$ of connectivity matrix $\mathbf{T}$ (e.g., for general similarity matrices see Coifman et al., 2005):

$$d_t(x,y) = \left\| \psi^t(x) - \psi^t(y) \right\|_2 , \qquad \psi_i^t(x) := \lambda_i^t \phi_{ix} , \qquad \forall t \in \mathbb{N} .$$

An extension to potentially continuous observation (or state) spaces $\mathcal{Z}$ with ergodic transition kernels $P^\pi : \mathcal{Z} \times \mathcal{B}(\mathcal{Z}) \to [0,1]$ is not trivial. Mean-squared differences between distributions are not directly possible, but one can calculate the difference between *Radon-Nikodym derivatives*.[20] Due to ergodicity the derivative always exists for finite sets $\mathcal{Z}$, but for continuous $\mathcal{Z}$ one must exclude transition kernels that are not absolutely continuous.[21]

Assumption 1 *If $\mathcal{Z}$ is continuous, the transition kernel knows no finite set of future states.*

$$P(B|z,a) \quad = \quad 0 , \qquad \forall B \in \left\{ B \in \mathcal{B}(\mathcal{Z}) \,\middle|\, |B| < \infty \right\} , \quad \forall z \in \mathcal{Z} , \quad \forall a \in \mathcal{A} .$$

This can always be fulfilled by adding a small amount of *continuous noise* (e.g., Gaussian) to each transition. Let in the following $(P^\pi)^t(\cdot,x) : \mathcal{B}(\mathcal{Z}) \to [0,1]$ denote the state distribution after t transitions, starting at state $x \in \mathcal{Z}$. Note that under Assumption 1 the Radon-Nikodym derivative w.r.t. steady state distribution ξ is[22] $\frac{d(P^\pi)^t(\cdot|x)}{d\xi} \in L^2(\mathcal{Z},\xi), \forall t \in \mathbb{N}\setminus\{0\}$.

Definition 1 *The diffusion distance $d_t : \mathcal{Z}\times\mathcal{Z} \to \mathbb{R}^+$ based on ergodic transition kernel P^π with steady state distribution ξ is defined as*

$$d_t(x,y) \;:=\; \left\| \mu_x^t - \mu_y^t \right\|_\xi , \qquad \mu_x^t \;:=\; \tfrac{d(P^\pi)^t(\cdot|x)}{d\xi} \in L^2(\mathcal{Z},\xi) , \qquad \forall x,y \in \mathcal{Z} , \quad \forall t \in \mathbb{N}\setminus\{0\} .$$

19. In our context these weights are the equivalent to the steady state distribution and thus named the same.
20. If Radon-Nikodym derivative $\frac{d\zeta}{d\xi}$ exists then $\int \xi(dz) \frac{d\zeta}{d\xi}(z) f(z) \;=\; \int \zeta(dz) f(z)$, $\forall f \in L^2(\mathcal{Z},\xi)$.
21. The Radon-Nikodym derivative $\frac{d\zeta}{d\xi}$ exists if distribution ζ is *absolutely continuous* w.r.t. steady state distribution ξ, that is if $\xi(B) = 0 \;\Rightarrow\; \zeta(B) = 0$, $\forall B \in \mathcal{B}(\mathcal{Z})$. If there exists a finite Borel set $B \in \mathcal{B}(\mathcal{Z})$ with $\zeta(B) > 0$, however, the derivative must not exist as $\xi(B) = 0$ can hold for ergodic Markov chains.
22. Assumption 1 guarantees the Radon-Nikodym derivative exists in the space of integrable functions $L^1(\mathcal{Z},\xi)$, but by compactness of $\mathcal{Z}$ the derivative is also in $L^2(\mathcal{Z},\xi) \subset L^1(\mathcal{Z},\xi)$ (Reed and Simon, 1980).

The projection methods discussed in Section 2 are based on *Euclidean distances* in the approximation space $\mathcal{F}_\phi$. These spaces are invariant to scaling of the basis functions. Given a particular scaling, however, diffusion distances can equal Euclidean distances in $\mathcal{F}_\phi$. In this case we say that the basis functions *encode* this distance.

Definition 2 *Basis functions $\phi_i \in L^2(\mathcal{Z},\xi), i \in \{1,\dots,p\}$, are said to "encode" a distance function $d : \mathcal{Z}\times\mathcal{Z} \to \mathbb{R}^+$ if there exists a scaling vector $\varrho \in (\mathbb{R}^+)^p$ such that*

$$d(x,y) \quad = \quad \sqrt{\sum_{i=1}^{p} \varrho_i \Big(\phi_i(x) - \phi_i(y)\Big)^2} \quad = \quad \Big\|\phi(x) - \phi(y)\Big\|_\varrho , \qquad \forall x,y \in \mathcal{Z}.$$

If the analogy between value function generalization and diffusion distances is correct, one should aim for a set of basis functions that at least approximates an encoding of diffusion distances $d_t(\cdot,\cdot)$, if possible for all forecast parameters $t \in \mathbb{N}\setminus\{0\}$ at once.

Lemma 3 *Let ξ denote the steady state distribution of ergodic transition kernel P^π, which has a self-adjoint transition operator $\hat{P}^\pi = (\hat{P}^\pi)^* : L^2(\mathcal{Z},\xi) \to L^2(\mathcal{Z},\xi)$. The corresponding diffusion distance equals the Euclidean distance in the space spanned by $\psi_i^t(\cdot) := \lambda_i^t \phi_i(\cdot), \forall i \in \mathbb{N}$, where $\lambda_i \in \mathbb{R}$ and $\phi_i \in L^2(\mathcal{Z},\xi)$ are the eigenvalues and eigenfunctions of $\hat{P}^\pi$, that is*

$$d_t(x,y) = \|\psi^t(x) - \psi^t(y)\|_2 , \quad \forall x,y \in \mathcal{Z}, \forall t \in \mathbb{N}\setminus\{0\}.$$

Proof *see Appendix A, Page 2108.* ■

Note that the full set of eigenfunctions ϕ_i encodes *all* diffusion distances $d_t(\cdot,\cdot), \forall t \in \mathbb{N}\setminus\{0\}$. Lemma 3 shows that the above relationship between diffusion and Euclidean distances in the eigenspace of the transition operator $\hat{P}^\pi : L^2(\mathcal{Z},\xi) \to L^2(\mathcal{Z},\xi)$ also holds, but only if this operator is *self-adjoint*.[23] This does not hold for most transition operators, however. Their eigenfunctions do not have to be orthogonal or even be real-valued functions, analogous to complex eigenvectors of asymmetric matrices. Using these eigenfunctions, on the other hand, is the declared intent of *subspace-invariant features* (see Section 3). Constructing real-valued basis functions with zero *per-feature error* thus does not seem generally possible.

In this light one can interpret the *symmetrized transition possibilities* encoded by PVF as a self-adjoint approximation of the *transition probabilities* of P^π. This raises the question whether better approximations exist.

Lemma 4 *Let P^π be an ergodic transition kernel in $\mathcal{Z}$ with steady state distribution ξ. The kernel induced by adjoint transition operator $(\hat{P}^\pi)^*$ in $L^2(\mathcal{Z},\xi)$ is ξ-almost-everywhere an ergodic transition kernel with steady state distribution ξ.*
Proof *see Appendix A, Page 2109.* ■

To obtain a self-adjoint transition kernel, Lemma 4 shows that the kernel of the adjoint operator $(\hat{P}^\pi)^*$ is a transition kernel as well. Intuitively, when $\hat{P}^\pi$ causes all water to flow downhill, $(\hat{P}^\pi)^*$ would cause it to flow the same way uphill. Note the difference to an *inverse* transition kernel, which could find new ways for the water to flow uphill. Although this changes the transition dynamics,

23. Each linear operator $\hat{A} : L^2(\mathcal{Z},\xi) \to L^2(\mathcal{Z},\xi)$ has a unique *adjoint operator* $\hat{A}^*$ for which holds: $\langle f, \hat{A}[g]\rangle_\xi = \langle \hat{A}^*[f], g\rangle_\xi, \forall f,g \in L^2(\mathcal{Z},\xi)$. The operator is called *self-adjoint*, if $\hat{A} = \hat{A}^*$.

one can construct a *symmetrized transition operator* $\hat{P}_s^{\pi} := \frac{1}{2}\hat{P}^{\pi} + \frac{1}{2}(\hat{P}^{\pi})^*$ as a self-adjoint approximation of $\hat{P}^{\pi}$. Estimating $\hat{P}_s^{\pi}$ may take more samples than the connection graph $\mathbf{T}$ constructed by PVF, but it stands to reason that $\hat{P}_s^{\pi}$ is a better approximation to $\hat{P}^{\pi}$. This intuition is put to the test in Section 5.1. We find indeed that SFA features have on average smaller *per-feature errors* than PVF. For purely random transition kernels the advantage of SFA is minuscule, but the when $\hat{P}^{\pi}$ resembles a self-adjoint operator the difference is striking (see Figure 3 on Page 2092). The goal of encoding diffusion distances based on P^{π} appears thus best served by the eigenfunctions of the symmetrized transition operator $\hat{P}_s^{\pi}$. Lemma 5 shows[24] that in the limit of an infinite training sequence, SFA extracts these eigenfunctions in the order of their largest eigenvalues:

Lemma 5 *In the limit of an infinite ergodic Markov chain drawn by transition kernel P^{π} in $\mathcal{Z}$ with steady state distribution ξ holds $\mathcal{S}(f) = 2\,\langle f, (\hat{I} - \hat{P}^{\pi})[f]\rangle_{\xi}, \forall f \in L^2(\mathcal{Z}, \xi)$.*
Proof *see Appendix A, Page 2109.* ■

Note that the first, constant eigenfunction of $\hat{P}_s^{\pi}$ is not extracted, but has no influence on the encoded distance. Encoding any diffusion distance $d_t(\cdot,\cdot)$ would therefore need a potentially infinite number of SFA features. As the influence of each feature shrinks exponentially with the forecast parameter t, however, the encoding can be approximated well by the first p SFA features. Except for $t = 0$, this approximation is optimal in the least-squares sense. Note also that for fixed p the approximation quality *increases* with t. Predictions based on SFA features will therefore be more accurate in the long term than in the short term.

Theorem 6 *SFA features $\{\phi_i\}_{i=1}^{\infty}$ simultaneously encode all diffusion distances $d_t(\cdot,\cdot), \forall t \in \mathbb{N} \setminus \{0\}$, based on the symmetrized transition kernel $P_s^{\pi} = \frac{1}{2}P^{\pi} + \frac{1}{2}(P^{\pi})^*$. The first p SFA features are an optimal p-dimensional least-squares approximation to this encoding.*
Proof *The theorem follows directly from Definition 2 and Lemmas 3, 4, 5 and 15.* ■

A similar proposition can be made for PVF features and diffusion distances based on *transition possibilities*. The connection to reward-based features (Section 3.1) is less obvious. Concentrating naturally on immediate and short term reward, these basis functions depend on the *reward function* at hand. It is, however, possible to show the encoding of diffusion distances *on average*, given the reward function is drawn from a *white noise functional*[25] ρ.

Theorem 7 *On average over all reward functions $r^{\pi} : \mathcal{Z} \to \mathbb{R}$ drawn from a white noise functional ρ, the squared norm of a Krylov basis $\{\phi_i^K\}_{i=1}^{p}$ from an ergodic transition kernel P^{π} encodes squared diffusion distances based on $\hat{P}^{\pi}$ up to horizon $p-1$, that is*

$$d_t^2(x,y) \;=\; \mathbb{E}\left[\left\|\phi^K(x) - \phi^K(y)\right\|_{\varrho}^2 \,\middle|\, r^{\pi} \sim \rho\right], \quad \forall x,y \in \mathcal{Z},\, \exists \varrho \in (\mathbb{R}^+)^p,\, \forall t \in \{1,\ldots,p-1\}.$$

Proof *see Appendix A, Page 2110.* ■

Although Krylov bases are different for each reward function $r^{\pi} \in L^2(\mathcal{Z}, \xi)$ and the employed squared distances diverge slightly from Definition 1, Theorem 7 implies that on average they encode

24. Lemma 5 shows that the SFA optimization problem is equivalent to $\inf_{\phi} \langle \phi, (\hat{I} - \hat{P}^{\pi})[\phi]\rangle_{\xi} \equiv \sup_{\phi} \langle \phi, \hat{P}^{\pi}[\phi]\rangle_{\xi} = \sup_{\phi} \langle \phi, \hat{P}_s^{\pi}[\phi]\rangle_{\xi}$, due to the symmetry of the inner product.

25. A white noise functional is the Hilbert space equivalent to a Gaussian normal distribution (Holden et al., 2010). In our context it suffices to say that $\mathbb{E}[\langle f, r^{\pi}\rangle_{\xi}^2 \,|\, r^{\pi} \sim \rho] = \langle f, f\rangle_{\xi}, \forall f \in L^2(\mathcal{Z}, \xi)$.

diffusion distances up to time horizon $p-1$. The same results hold for BEBF (Petrik, 2007) and with minor modifications for BARB bases (Mahadevan and Liu, 2010).

In conclusion, reward-based features encode diffusion distances exactly up to some time horizon, whereas SFA and PVF approximate an encoding for *all* possible distances. So far the only connection to value function approximation is the intuition of a generalizing metric. However, in the next subsection we show striking parallels between diffusion distances and approximation errors.

4.2 Value Function Approximation

The analysis in Section 4.1 revealed a critical problem for the construction of *subspace-invariant features* (Parr et al., 2008): eigenfunctions of the transition operator $\hat{P}^{\pi}$ are not necessarily orthogonal and real-valued. Constructing a real-valued, orthonormal basis of subspace-invariant features is thus only possible in some rare cases of self-adjoint transition operators. Both SFA and PVF substitute therefore a "similar" self-adjoint transition operator for $\hat{P}^{\pi}$. SFA employs the *symmetrized* operator $\hat{P}^{\pi}_s := \frac{1}{2}\hat{P}^{\pi} + \frac{1}{2}(\hat{P}^{\pi})^*$ and PVF assigns equal probability to *all possible* neighbors[26] $\hat{T}^{\pi}$. Analytical comparison of the quality of these approximations is difficult, however.

On the other hand, the class of MDPs for which SFA features are subspace-invariant *contains* the class for which PVF are. To see this, imagine a transition kernel T^{π} for which PVF are subspace-invariant, which implies that for each state there exists a uniform distribution to end up in the set of its neighbors, with symmetric neighborhood relationships. As $\hat{T}^{\pi}$ is thus self-adjoint, any ergodic Markov chain from this kernel will yield subspace-invariant SFA features. Reversely, one can construct a transition kernel P^{π} with a self-adjoint transition operator but without uniform transition probabilities. PVF would no longer correspond to eigenfunctions of $\hat{P}^{\pi}$ and would thus not be subspace-invariant. SFA can in this sense be seen as a generalization of PVF.

Within the class of MDPs with self-adjoint transition operators, however, one can make some strong claims regarding *value function approximation* with LSTD (Section 2.3).

Lemma 8 *Let $\{\phi_i\}_{i=1}^p$ denote any p SFA features from a MDP with self-adjoint transition operator, then the LSTD fixed point $f^{\pi} = \hat{\Pi}^{\phi}_{\xi}\left[\hat{B}^{\pi}[f^{\pi}]\right]$ and the projection of true value function $v^{\pi} = \hat{B}^{\pi}[v^{\pi}]$ coincide, that is*

$$f^{\pi}(x) \;=\; \hat{\Pi}^{\phi}_{\xi}[v^{\pi}](x) \;=\; \sum_{i=1}^{p} \langle r^{\pi}, \phi_i \rangle_{\xi}\, \tau_i\, \phi_i(x)\,, \quad \forall x \in \mathcal{Z}, \quad \tau_i := \left(1-\gamma+\tfrac{\gamma}{2}\mathcal{S}(\phi_i)\right)^{-1}.$$

Proof *see Appendix A, Page 2110.* ■

Lemma 8 implies that for SFA features of symmetric transition models, the bound of Tsitsiklis and Van Roy (1997, introduced in Section 2.3) can be dramatically improved:

Corollary 9 *The approximation error of the LSTD solution f^{π} to the true value v^{π} for MDPs with self-adjoint transition operators using any corresponding SFA features $\{\phi_i\}_{i=1}^p$ is*

$$\left\| v^{\pi} - f^{\pi} \right\|_{\xi} \quad = \quad \left\| v^{\pi} - \hat{\Pi}^{\phi}_{\xi}[v^{\pi}] \right\|_{\xi}.$$

26. In the discrete case these are all observed transitions, in the continuous case neighborhood relationships are based on similarities in observation space $\mathcal{Z}$ (Mahadevan and Maggioni, 2007).

An analogous proposition can be made for PVF, but for a smaller subset of MDPs. Equivalent fixed point solutions for p reward-based features, on the other hand, do not appear generally possible as the behavior beyond $p-1$ time steps is not encoded (see Theorem 7). However, it is easy to see that *finite horizon* solutions can be computed *exactly* by projected value iteration (finite application of the projected Bellman operator, see, e.g., Bertsekas, 2007).

Corollary 10 *Finite horizon value functions* $v_h^{\pi} := (\hat{B}^{\pi})^h[r] = (\hat{\Pi}_{\xi}^{\phi^K}[\hat{B}^{\pi}])^h[r]$ *can be computed exactly up to horizon* $h = p-1$ *by projected value iteration with Krylov base* $\{\phi_i^K\}_{i=1}^p$.

The conclusions from Theorems 6 & 7 and Corollaries 9 & 10 are very similar: SFA/PVF optimally approximate/generalize *infinite-horizon* value functions for a subset of possible MDPs, whereas reward-based features represent/generalize *finite-horizon* value functions exactly without any such restrictions.

There have also been attempts to join both types of basis functions by selecting the subspace-invariant feature most similar to the current Bellman error (Petrik, 2007; Parr et al., 2008). To motivate this, Parr et al. (2007) gave a lower bound for the approximation-bound improvement if a BEBF feature ϕ_p^B is added to the set $\Phi_{p-1}^B := \{\phi_i^B\}_{i=1}^{p-1}$:

$$\left\|v^{\pi} - \hat{\Pi}_{\xi}^{\Phi_{p-1}^B}[v^{\pi}]\right\|_{\xi} - \left\|v^{\pi} - \hat{\Pi}_{\xi}^{\Phi_p^B}[v^{\pi}]\right\|_{\xi} \quad \geq \quad \left\|v^{\pi} - f^{(p-1)}\right\|_{\xi} - \left\|v^{\pi} - \hat{B}\left[f^{(p-1)}\right]\right\|_{\xi},$$

where $f^{(p-1)} \in \mathcal{F}_{\Phi_{p-1}^B}$ is the LSTD fixed point solution based on Φ_{p-1}^B. One can observe that for each added feature the bound shrinks by the Bellman error function ϕ_p^B. The PVF feature with the highest correlation to ϕ_p^B is thus a good subspace-invariant choice.

We can provide an even stronger assertion about the approximation error of SFA features here. Theorem 11 does not rely on the knowledge of a current LSTD solution, but on the similarity of reward function r^{π} and SFA feature ϕ_p. Given SFA features and reward, the basis can thus be selected *before* training begins.

Theorem 11 *Let* ξ *be the steady state distribution on* $\mathcal{Z}$ *of a MDP with policy* π *and a self-adjoint transition operator in* $L^2(\mathcal{Z},\xi)$. *Let further* $\Phi_p = \{\phi_i\}_{i=1}^p$ *be any set of p SFA features and* $v^{\pi} \in L^2(\mathcal{Z},\xi)$ *the true value of the above MDP. The improvement of the LSTD solution* $f^{(p)} := \hat{\Pi}_{\xi}^{\Phi_p}\left[\hat{B}^{\pi}[f^{(p)}]\right]$ *by including the p'th feature is bounded from below by*

$$\left\|v^{\pi} - f^{(p-1)}\right\|_{\xi} - \left\|v^{\pi} - f^{(p)}\right\|_{\xi} \quad \geq \quad \frac{1-\gamma}{2}\,\frac{\langle r^{\pi},\phi_p\rangle_{\xi}^2}{\|r^{\pi}\|_{\xi}}\,\tau_p^2, \qquad \tau_p := \left(1-\gamma+\frac{\gamma}{2}\mathcal{S}(\phi_p)\right)^{-1}.$$

Proof *see Appendix A, Page 2111.* ■

The bound improves with the similarity to reward function $r^{\pi} \in L^2(\mathcal{Z},\xi)$. The factor τ_p, defined in Lemma 8, is inversely related to the slowness of the feature ϕ_p. In $L^2(\mathcal{Z},\xi)$ we can guarantee[27] for

27. Lemma 5, Page 2085, shows that the slowness of eigenfunction ϕ_p of self-adjoint P^{π} is related to the corresponding eigenvalue λ_p by $\mathcal{S}(\phi_p) = 2(1-\lambda_p)$. Eigenvalues can be negative, but since $\lim_{p\to\infty}|\lambda_p| = 0$, every *finite* set of SFA features for *infinite* state/observation spaces will correspond to nonnegative eigenvalues only. In finite state spaces or in general RKHS with finite support, for example, in all sparse kernel algorithms, one can only guarantee $0 < \mathcal{S}(\phi_p) \leq 4$ and $\lim\limits_{p\to\infty}\lim\limits_{\gamma\to 1}\tau_p = \frac{1}{2}$.

infinite state/observation spaces that $\mathcal{S}(\phi_1) > 0$ and $\lim_{p\to\infty} \mathcal{S}(\phi_p) = 2$ that

$$\lim_{\gamma\to 0} \tau_p \quad = \quad 1\,, \qquad \lim_{\gamma\to 1} \tau_p \quad = \quad \frac{2}{\mathcal{S}(\phi_p)} \quad \geq \quad 1\,, \qquad \lim_{p\to\infty} \lim_{\gamma\to 1} \tau_p \quad = \quad 1\,.$$

One could use this bound to select the best feature set for a given MDP, similar to *matching pursuit* approaches (Mallat and Zhang, 1993). However, this is beyond the scope of this article and left for future works.

4.3 Encoding Anticipated Value Functions

The last subsection analyzed the properties of SFA for value function approximation of an MDP at hand. Constructed features still need to be represented somehow, for example with a RKHS or some larger set of given basis functions. Reward-based features like BEBF can reduce the value estimation time by incrementally increasing the feature set by the current Bellman error. Strictly, there is no reason to *remember* those features, though. One could instead simply remember the current value estimate. And since the features depend on the reward function, reusing them to solve another MDP is out of the question.

Subspace-invariant features, on the other hand, do not depend on the reward function but are very expensive to construct. This raises the question of *when* these features are actually computed. For example, constructing p RSK-SFA features based on a Markov chain of n observations with a sparse subset of m support observations yields a computational complexity of $\mathcal{O}(m^2 n)$. Sparse kernel LSTD (Xu, 2006) alone exhibits the same complexity but makes use of the full span of the sparse subset, instead of only a p-dimensional subspace thereof. Computing features and the value function at the same time therefore does not yield any computational advantage.

Alternatively, one could rely on previously experienced "similar" MDPs to construct the basis functions (*transfer learning*, Taylor and Stone, 2009). Ferguson and Mahadevan (2006) and Ferrante et al. (2008) followed this reasoning and constructed PVF out of experiences in MDPs with the same transition, but different reward model. This section aims to analyze this transfer effect without going into the details of how to choose the MDPs to learn from.

Instead of defining "similar" MDPs, we ask how well one can approximate all value functions for a set of *anticipated tasks* $\mathcal{M}$. The set of all value functions one might encounter during *value iteration* is huge. For LSTD, however, one only has to consider fixed points $f^\pi \in L^2(\mathcal{Z}, \xi)$ of the combined operator $\hat{\Pi}_\xi^\phi \left[\hat{B}^\pi[\cdot]\right]$ (see Section 2.3). Note that there is a unique fixed point f^π for every policy π from the set of allowed policies Ω, for example all deterministic policies. Moreover, Tsitsiklis and Van Roy (1997) have derived the upper bound

$$\left\| v^\pi - f^\pi \right\|_\xi \leq \frac{1}{\sqrt{1-\gamma^2}} \left\| v^\pi - \hat{\Pi}_\xi^\phi \left[v^\pi\right] \right\|_\xi ,$$

which means that the approximation error (left hand side) is bounded by the projection error of *true* value function $v^\pi \in L^2(\mathcal{Z}, \xi)$ onto $\mathcal{F}_\phi$ (right hand side). It stands to reason that a set of basis functions which minimizes the right hand side of this bound for all tasks in $\mathcal{M}$ and policies from Ω according to their occurrence can be called *optimal* in this sense.

Definition 12 *A set of p basis functions* $\{\phi_i\}_{i=1}^p \subset L^2(\mathcal{Z},\xi)$ *is called "optimal" w.r.t. the distributions* $\rho : \mathcal{B}(\mathcal{M}) \to [0,1]$ *and* $\omega : \mathcal{B}(\Omega) \to [0,1]$*, if they are a solution to*

$$\inf_{\phi \in (L^2(\mathcal{Z},\xi))^p} \mathbb{E}\left[\underbrace{\left\| v_m^\pi - \hat{\Pi}_\xi^\phi \left[v_m^\pi \right] \right\|_\xi^2}_{\textit{(bound)}} \;\middle|\; \begin{array}{l} m \sim \rho(\cdot) \\ \pi \sim \omega(\cdot) \end{array} \right].$$

The expectation integrates over all true value functions v_m^π which are determined by all policies $\pi \in \Omega$, drawn from some distribution $\omega : \mathcal{B}(\Omega) \to [0,1]$ (e.g., uniform) and all tasks m drawn from distribution[28] $\rho : \mathcal{B}(\mathcal{M}) \to [0,1]$. A similar definition of optimality has been proposed in the context of *shaping functions*[29] (Snel and Whiteson, 2011). Other definitions of optimality are discussed in Section 6.2.

The optimization problem in Definition 12 has no general analytic solution. In particular there is no solution for one MDP and all policies, which would be ideal for policy iteration (e.g., LSPI, Section 2.4). There is another special case, however, which demonstrates the setting under which SFA extracts nearly optimal approximation spaces.

For a fixed policy $\pi \in \Omega$ and task $m \in \mathcal{M}$ one can calculate the exact value function $v^\pi \in L^2(\mathcal{Z},\xi)$. Let $(\hat{I} - \gamma \hat{P}^\pi)^{-1}$ denote the inverse operator[30] to $(\hat{I} - \gamma \hat{P}^\pi)$, then

$$v^\pi \quad \stackrel{!}{=} \quad \hat{B}^\pi[v^\pi] \quad = \quad r^\pi + \gamma \hat{P}^\pi[v^\pi] \quad = \quad (\hat{I} - \gamma \hat{P}^\pi)^{-1}[r^\pi].$$

Substituting this into Definition 12, one can give an analytic solution $\phi_i \in L^2(\mathcal{Z},\xi), i \in \{1,\ldots,p\}$, for all tasks within the same environment,[31] that is, $\mathcal{M} := \{(\mathcal{X},\mathcal{A},P,r) \mid r \sim \rho\}$, restricted to the sampling policy π, that is, $\Omega := \{\pi\}$. The key insight is that the only allowed difference between tasks is the expected reward function $r^\pi : \mathcal{Z} \to \mathbb{R}$. If we do not constrain the possible reward functions (e.g., all states are possible goals for navigation), their statistics ρ can be described as a *white noise functional* in $L^2(\mathcal{Z},\xi)$ (Holden et al., 2010, see also Footnote 25 on Page 2085).

Theorem 13 *For any infinite ergodic Markov chain with steady state distribution* ξ *over state space* $\mathcal{Z}$*, SFA selects features from function set* $\mathcal{F} \subset L^2(\mathcal{Z},\xi)$ *that minimize an upper bound on the optimality criterion of Definition 12 for sampling policy* π *and discount factor* $\gamma > 0$*, under the assumption that the mean-reward functions* $r^\pi : \mathcal{Z} \to \mathbb{R}$ *are drawn from a white noise functional in* $L^2(\mathcal{Z},\xi)$*.*

Proof: see Appendix A, Page 2111. ■

The main result of Theorem 13 is that under the above assumptions, SFA approximates the *optimal basis functions* of Definition 12. To be exact, the SFA objective minimizes a *bound* on the optimality criterion. A closer look into the the proof of Theorem 13 on Page 2111 shows that the exact solution in Definition 12 requires a bias-free estimation of the term $\left\| (\hat{P}^\pi)^* [\phi_i] \right\|_\xi^2$, which is impossible without *double sampling* (see, e.g., Sutton and Barto, 1998). We argue therefore that *SFA constitutes the best approximation to optimal features one can derive using a single Markov chain*. Note that unlike the results in Sections 4.1 and 4.2, this conclusion is not restricted to self-adjoint transition operators.

28. To define a proper distribution ρ one must formally define all anticipated MDPs in $\mathcal{M}$ over the union of all involved state-action spaces. See Snel and Whiteson (2011) for an example of such an approach.
29. In the context of value iteration, shaping functions are equivalent to an initialization of the value function.
30. The existence of such an operator is shown in Lemma 14, Page 2112.
31. With the same state (observation) space $\mathcal{Z}$, action space $\mathcal{A}$ and transition kernel P. This class of tasks is also called *variable-reward* tasks (Mehta et al., 2008) and applies for example when a flying robot needs to execute different maneuvers, but is constraint by the same aerodynamics (Abbeel et al., 2007).

4.4 Conclusion

Although no direct relationship has been proven, this section has provided evidence for a strong connection between diffusion distances and value function approximation. Our analysis suggests that *reward-based* features can represent finite-horizon value functions exactly. They do not generalize to different MDPs or policies and can thus as well be forgotten after the value estimate is updated. *Subspace-invariant* features, on the other hand, approximate infinite-horizon values optimally if the transition kernel adheres to some restrictions. Moreover, we argue that even in the general case, SFA provides the best possible construction method based on a single Markov chain. Computational complexity prevents this class of features from performing cost-efficient dimensionality reduction for LSTD, though. On the other hand, subspace-invariant features provide on average an optimal basis for *all* reward functions within the same environment. This optimality is still restricted to the sampling policy π, but SFA features should have an advantage over PVF here, as they are subspace-invariant for a much *larger* class of MDPs.

Using such features effectively for transfer learning or within policy iteration requires them to perform well for other policies π', in other words to induce little *per-feature errors* when applied to $\hat{P}^{\pi'}$. In the absence of theoretical predictions a uniform sampling-policy π appears to be a reasonable choice here. Note that depending on the transition kernel P^{π}, this can still yield an arbitrary steady state distribution ξ. PVF features are in the limit not affected by ξ and *importance sampling* should be able to compensate the dependence of SFA (see Section 3.5). Theoretical statements about the influence of sampling policy and steady state distribution on SFA and PVF per-feature errors with other policies, however, are beyond the scope of this article and left for future works. See Section 6.1 for a discussion.

Although SFA is more sensitive to the sampling policy than PVF, the presented analysis suggests that it can provide better approximation spaces for value estimation, that is, LSTD.

5. Empirical Analysis

This section empirically evaluates the the construction of approximation spaces in light of the theoretical predictions of Section 4. Our analysis focuses on three questions:

1. How well does LSTD estimate the value function of a given Markov chain?

2. How good is the performance of policies learned by LSPI based on a random policy?

3. How does this performance depend on the approximation space metric?

We start with evidence for the relationship (hypothesized in Section 4.1) between *per-feature errors* (see Section 3) and how *self-adjoint* a transition operator is. Furthermore, Section 4.2 predicts that the set of MDPs for which PVF are *subspace-invariant* is a subset of the respective set of SFA. To test both possibilities we evaluated the first two questions on two discrete benchmark tasks: the *50-state chain* in Section 5.2 and the more complicated *puddle-world task* in Section 5.3. The third question can not be answered with a discrete metric. To test the influence of the observation space metric, we designed a simple but realistic robot navigation task with continuous state and observation spaces (Section 5.4). A robot must navigate into a goal area, using first-person video images as observations. Results are presented in Sections 5.5, 5.6 and 5.7.

5.1 Comparison of Subspace Invariance

In Section 4.1 we stated our intuition that the *symmetrized transition operator* of SFA approximates the true transition operator better than the *neighborhood operator* of PVF. Here we want to substantiate this intuition by testing the *per-feature errors* of theoretical SFA and PVF solutions applied to randomly generated MDPs.

5.1.1 THEORETICAL FEATURES

Discrete MDPs allow an exact calculation of the objectives described in Section 3. To test the limit case of an infinite Markov chain one can generate *theoretical* features, which is straight forward[32] for SFA. These features depend on *steady state distribution* (s.s.d.) ξ. Changing ξ has a surprising effect on per-feature errors, in particular if one assumes a uniform ξ. To demonstrate this effect, all experiments with theoretical features also include this case.

Theoretical PVF, on the other hand, require a proper definition of *neighborhood*. As all states could be connected by the transition kernel, *transition possibility* (as in Section 3.3) is not always an option. We followed the *k-nearest neighbor* approach of Mahadevan and Maggioni (2007) instead and defined[33] the k most probable transitions as neighbors.

For each feature ϕ_i from a set $\Phi_p := \{\phi_1, \ldots, \phi_p\}$ one can calculate the *per feature error* $\Delta_i^{\Phi_p} \in L^2(\mathcal{X}, \xi)$ (see Section 3). To measure how how strongly Φ_p diverges from *subspace invariance*, we add the norms of all p error functions together, that is, $\sum_{i=1}^{p} \|\Delta_i^{\Phi_p}\|_\xi$. This yields a measure of subspace invariance for each set of p features. To compare construction methods we also calculated the mean of the above measure over $p \in \{1, 2, \ldots, 100\}$.

5.1.2 SUBSPACE-INVARIANCE AND SELF-ADJOINT TRANSITION OPERATORS

To investigate whether SFA or PVF features approximate arbitrary transition models better, we tested the per-feature errors $\Delta_i^{\Phi_p}$ of random MDPs. 100 MDPs with $d = 100$ states each were created. Each state is connected with 10 random future states and each connection strength is uniformly i.i.d. drawn. The resulting matrix is converted into a probability matrix $\mathbf{P}^\pi$ by normalization. SFA features are subspace-invariant for *self-adjoint* transition operators and PVF only for a subset thereof. As the above generated transition matrices are usually not self-adjoint, we repeatedly applied a *symmetrization* operator $\hat{G} : \mathbb{R}^{d \times d} \to \mathbb{R}^{d \times d}$ to each matrix $\mathbf{P}$, that is, $\hat{G}[\mathbf{P}]_{ij} := (P_{ij} + P_{ji})/(1 + \sum_k P_{jk})$. With each application the resulting transition matrices come closer and closer to be self-adjoint.

Figure 3 shows the measure for subspace invariance for theoretical PVF and SFA with both sampling distributions ξ. The left figure plots this measure against the feature set size p. One can observe that all methods show similar errors for the original asymmetric MDP (solid lines). Application of the symmetrization operator (dashed lines), on the other hand, yields a clear advantage for one SFA method. This becomes even more apparent in the right plot of Figure 3. Here the mean measure over all feature set sizes p is plotted against the number of symmetrization operator applications. One can observe that (in difference to PVF) the per-feature errors of both SFA meth-

32. SFA minimizes the slowness, in the limit according to Lemma 5: $\mathcal{S}(\phi_i) = 2\langle \phi_i, (\hat{I} - \hat{P}^\pi)[\phi_i] \rangle_\xi$. Let $\mathbf{P}^\pi$ be the transition matrix and $\mathbf{\Xi}$ a diagonal matrix of steady state distribution ξ, which is the left eigenvector to the largest eigenvalue of $\mathbf{P}^\pi$. Expressing the objective in matrix notation, the theoretical SFA features are the eigenvectors to the smallest eigenvalues of the symmetric matrix $2\mathbf{\Xi} - \mathbf{\Xi}\mathbf{P}^\pi - (\mathbf{\Xi}\mathbf{P}^\pi)^\top$.

33. We tested $k \in \{1, 2, 5, 10, 20, 50, 100\}$ and present here the best results for $k = 10$.

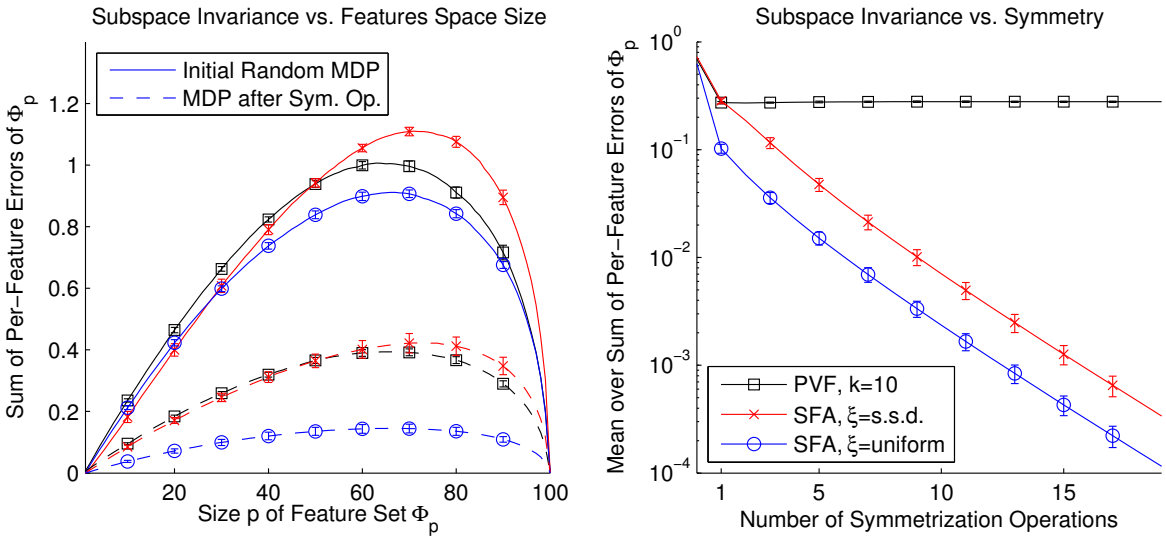


Figure 3: Subspace invariance of SFA and PVF features of random MDPs and symmetrized versions thereof. The left plot shows $\sum_{i=1}^{p} \|\Delta_i^{\Phi_p}\|_\xi$ for different feature set sizes p. The first symmetrization reduces per-feature errors of both methods, but all following symmetrization operations reduce only the error of SFA which is illustrated in the right plot. All means and standard deviations are w.r.t. 100 random MDPs. Note that the scale of per-feature errors differs between plots.

ods shrink the more self-adjoint the transition operator becomes. Also, SFA features with uniform distribution ξ are roughly 3 times as subspace-invariant as original SFA features with steady state distribution ξ.

We conclude that PVF and SFA methods approximate arbitrary asymmetric MDPs equally well. However, the more self-adjoint the transition operator, the larger the advantage of SFA. Furthermore, SFA features based on a uniform distribution ξ are on average more subspace-invariant than those based on the steady state distribution.

5.2 50-state Chain-Benchmark Task

First we investigate how well a basis constructed from a Markov chain can approximate the corresponding value function. The employed *50-state chain task* is based on a problematic 4-state MDP by Koller and Parr (2000) and has been extended in various variations by Lagoudakis and Parr (2003). Here we adopt the details from Parr et al. (2007). The task has a very similar transition- and neighborhood structure.

5.2.1 TASK

50 states are connected to a chain by two actions: move left and right. Both have a 90% chance to move in their respective direction and a 10% chance to do the opposite. Non-executable transitions at both ends of the chain remain in their state. Only the 10th and 41th state are rewarded. Executing any action there yields a reward of $+1$. The task is to estimate the value function with a discount

factor $\gamma = 0.9$ from a Markov chain. Note that transition probabilities of a random policy equal the neighborhood relationships. SFA features for this policy should therefore equal PVF features.

5.2.2 ALGORITHMS

We compare discrete versions of *slow feature analysis*, *proto value functions* and *Krylov bases* as described in Section 3. SFA feature sets also contained a constant feature. In this and the following experiments, higher Krylov bases have proven to be too similar for stable function approximation. We therefore orthonormalized each feature w.r.t. its predecessors, which solved the problem. After construction, the features were used to estimate the value function from the same training set with LSTD. Sampling influences value estimation here and to avoid the resulting bias we measure the difference (in some norm) to the LSTD solution with a *discrete* representation.

5.2.3 RESULTS

To explore the effect of the sampling policy, we tested the (non-deterministic) uniform and the (deterministic) optimal policy. Figure 4 plots mean and standard deviation of the LSTD solution difference in L_2 norm w.r.t. 1000 trials and 4000 samples each. To make sure all states are visited, an optimal policy trial is sampled in 40 trajectories with random start states and 100 samples each. Training sets that did not visit all states were excluded. As reported by previous works (Petrik, 2007; Parr et al., 2008; Mahadevan and Liu, 2010), reward-based features like Krylov bases perform in this task much better than subspace-invariant features. Solutions with PVF and SFA features are virtually identical for the random policy, as the transition probabilities of SFA equal the neighborhood relations of PVF. There are distinguishable differences for the optimal policy, but one can hardly determine a clear victor. Using the L_∞ norm for comparison (not shown) yields qualitatively similar results for the two feature spaces. Policy iteration did also not yield any decisive differences between SFA and PVF (not shown). In conclusion, the 50-state chain appears to belong to the class of MDPs for which features learned by SFA and PVF are not always identical, but *equally* able to estimate the value functions with LSTD.

5.3 Puddle-world–Benchmark Task

The puddle-world task was originally proposed by Boyan and Moore (1995), but details presented here are adapted from Sutton (1996). It is a continuous task which we discretize in order to compare reward-based features. This is a common procedure and allows to evaluate robustness by running multiple discretizations. In comparison to the 50-state chain the task is more complex and exhibits differing transition- and neighborhood-structures.

5.3.1 TASK

The state space is a two dimensional square of side length 1. Four actions move the agent on average 0.05 in one of the compass directions. The original task was almost deterministic (Sutton, 1996) and to make it more challenging we increased the Gaussian noise to a standard deviation of 0.05. Centered in the $(1,1)$ corner is an absorbing circular goal area with radius 0.1. Each step that does not end in this area induces a punishment of -1. Additionally, there exist two puddles, which are formally two lines $(0.1,0.75) \longleftrightarrow (0.45,0.75)$ and $(0.45,0.4) \longleftrightarrow (0.45,0.8)$ with a radius of 0.1 around them. Entering a state less than 0.1 away from one of the center-lines is pun-

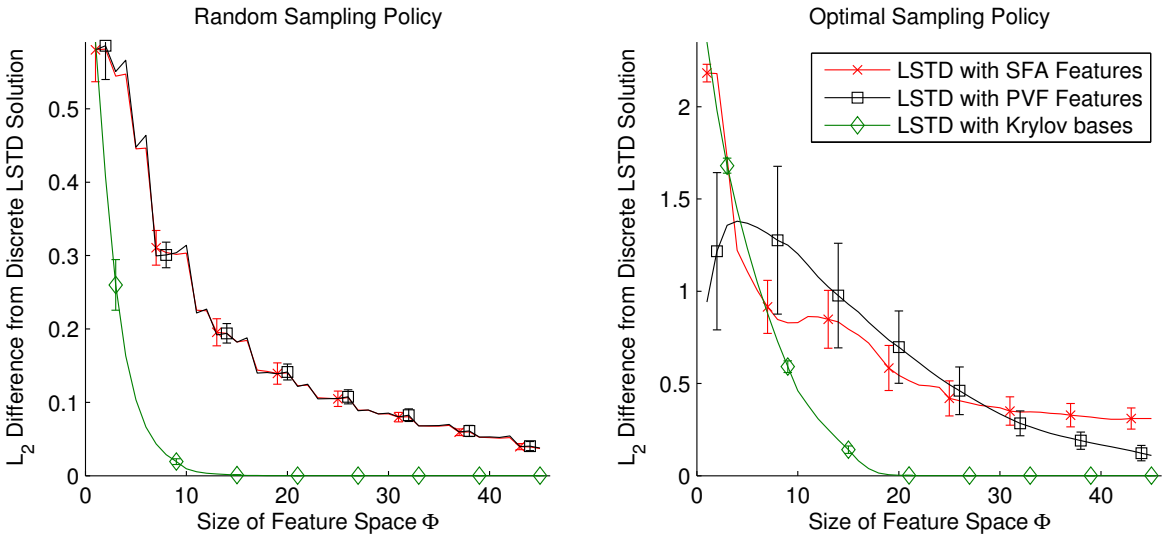


Figure 4: Difference in L_2 norm between approximated and discrete LSTD solutions vs. feature space size for random (left plot) and optimal (right plot) sampling policies in the 50-state chain. Means an standard deviations w.r.t. 1000 trials. L_∞ differences decrease slower but show otherwise the same qualitative trends.

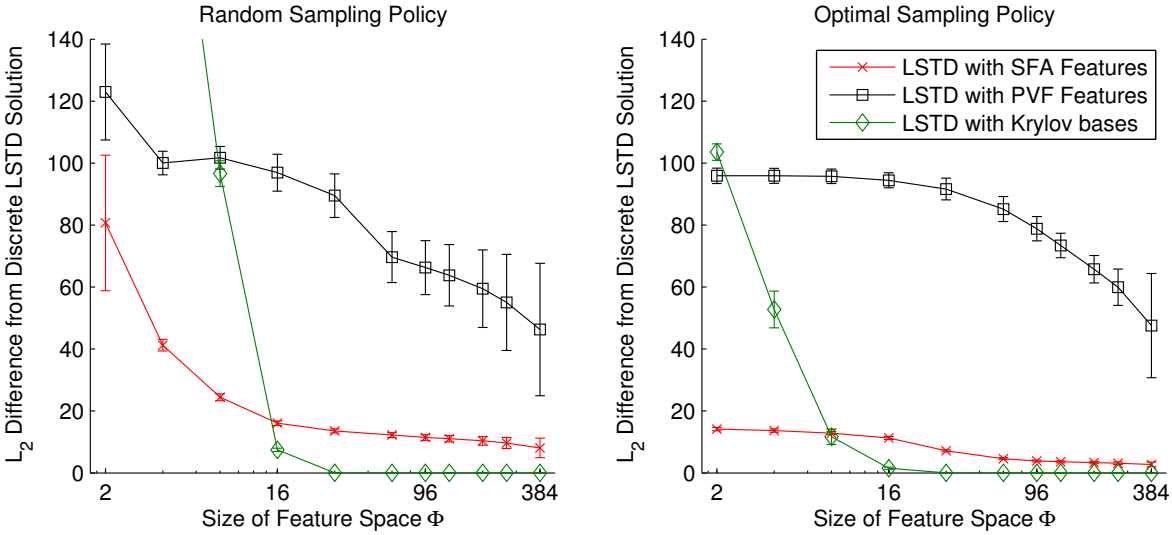


Figure 5: Difference in L_2 norm between approximated and discrete LSTD solutions vs. feature space size in puddle-worlds. Means and standard deviations are w.r.t. 10 training sets for each state space size $\{20\times 20, 25\times 25, \ldots, 50\times 50\}$, sampled with random (left plot) or optimal (right plot) policies. Note the logarithmic x-axis. Measuring the differences in L_∞ norm yields the same qualitative trends.

ished with -400 times the distance into the puddle. The task is to navigate into the goal, collecting as little punishment as possible. Seven discretizations with horizontal and vertical side length of $\bar{s} \in \{20, 25, \ldots, 50\}$ states were tested. Transition probabilities of states $\boldsymbol{x}_i$ to states $\boldsymbol{x}_j$ are the normalized *continuous* probabilities (average movement $\boldsymbol{\mu}_a$ plus Gaussian noise with standard deviation $\tilde{\sigma}$), that is

$$P_{ij}^a := \frac{\exp(-\frac{1}{2\tilde{\sigma}^2}\|\boldsymbol{x}_i + \boldsymbol{\mu}_a - \boldsymbol{x}_j\|_2^2)}{\sum_{k=1}^{\bar{s}^2}\exp(-\frac{1}{2\tilde{\sigma}^2}\|\boldsymbol{x}_i + \boldsymbol{\mu}_a - \boldsymbol{x}_k\|_2^2)}, \qquad \forall i, j \in \{1, \ldots, \bar{s}^2\}, \quad \forall a \in \{1, \ldots, 4\}.$$

5.3.2 ALGORITHMS

We evaluated discrete versions of *slow feature analysis* (SFA), *proto value functions* (PVF) and orthonormalized *Krylov bases*. To test the influence of sampling on feature construction we also evaluated the theoretical features of all three algorithms[34] (see Section 5.1). The algorithms were trained with a long (uniform) random policy Markov chain $\{\boldsymbol{x}_t\}_{t=1}^n$ in both LSTD and LSPI evaluations. To see the effect of different policies on LSTD, we trained all three on the *optimal* policy as well. In this case the training set consists of randomly initialized trajectories of length 20 to overcome sampling problems. We chose $n = 80\,\bar{s}^2$, as large state spaces require more samples to converge.

All constructed approximation spaces were tested with LSTD and LSPI[35] under discount factor $\gamma = 0.99$ on the same training sequence $\{\boldsymbol{x}_t\}_{t=1}^n$ used in feature construction. Policy iteration ended if the value of *all* samples differed by no more than 10^{-8} or after 50 iterations otherwise. To investigate asymptotic behavior we also tested LSPI with all state-action pairs and *true* mean future states as training set, corresponding to the limit of an infinite Markov chain. Performance of a (deterministic) policy learned with LSPI is measured by the *mean accumulated reward* of 1000 trajectories starting at random states. A test trajectory terminates after 50 transitions or upon entering the goal area.

5.3.3 LSTD EVALUATION

We tested the LSTD approximation quality as in Section 5.2. Figure 5 shows the difference in L_2 norm between the LSTD solution based on the constructed features and a discrete state representation vs. the number of employed features $p \in \{2, 4, 8, 16, 32, 64, 96, 128, 192, 256, 384\}$. Means and standard deviations are w.r.t. 10 training sets for each state space size $\{20\times 20, 25\times 25, \ldots, 50\times 50\}$, sampled with random (left plot) or optimal (right plot) policies. Note in comparison to Figure 4 that the x-axis is logarithmic. Reward-based Krylov bases have a clear advantage for $p \geq 16$ features (and reach perfection for $p \geq 64$), similar to the 50-state chain task. Fewer SFA features, on the other hand, capture the value function much better. This has also been observed for large discount factors γ when comparing BEBF and modified PVF (Mahadevan and Liu, 2010). The different encoding of diffusion distances (Theorems 6 & 7, Page 2085) provides a good explanation for this effect: Krylov bases represent finite-horizon value functions perfectly (Corollary 10, Page 2087),

34. We tested theoretic PVF with $k \in \{5, 10, 15, 20, 25, 50\}$ and chose $k = 10$. For larger k we observed slowly degrading performance, which is more pronounced under realistic LSPI sampling.

35. As convergence to the optimal policy can not be guaranteed for LSPI, the eventual policy depends also on the initial (randomly chosen) policy $\boldsymbol{\pi}_0$. In difference to Lagoudakis and Parr (2003) we used throughout this article a *true* (non-deterministic) random policy $\boldsymbol{\pi}_0(a|\boldsymbol{z}) = \frac{1}{|\mathcal{A}|}, \forall \boldsymbol{z} \in \mathcal{Z}, \forall a \in \mathcal{A}$. This heuristic appeared to be more robust in large feature spaces.

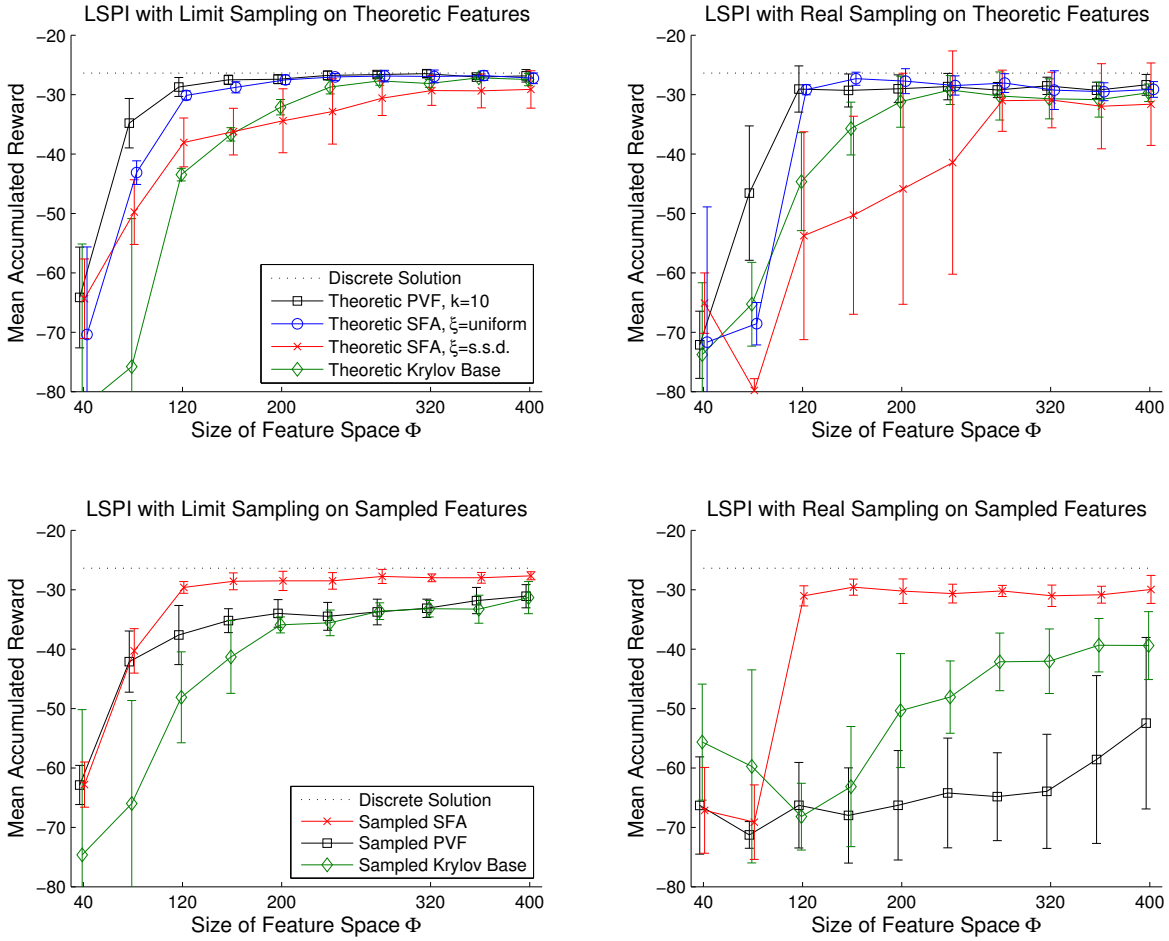


Figure 6: Performance of LSPI with optimal (left column) and realistic sampling (right column) in puddle-worlds. The upper row employed theoretical features and the lower row features constructed from samples. Means and standard deviations are w.r.t. state space sizes $\{20\times 20, 25\times 25, \ldots, 50\times 50\}$.

whereas SFA represents the infinite-horizon values approximatively (Corollary 9, Page 2086). However, Figure 5 clearly supports our hypothesis (proposed in Section 4.4) that *SFA can provide better approximation spaces for value function approximation with LSTD than PVF*. With respect to the previous subsection one should extend this hypothesis by adding *when transition and neighborhood structures are dissimilar*.

5.3.4 LSPI EVALUATION

Figure 6 shows the LSPI performance with both theoretic (upper row) and sampled features (lower row). LSPI was trained with an optimal training set (left column) and a realistic sequence drawn by

a random policy (right column). The top-left plot is the most hypothetical and the bottom-right plot the most realistic scenario. Means and standard deviation are w.r.t. state space sizes.

First and foremost, note that the sampled SFA features (crosses, lower row) perform significantly (more than one standard deviation) better, except in the under-fitting regime of 80 features and less. They are also the only sampled features that are robust against LSPI sampling, which can be seen by similar performance in both bottom plots. In comparison with theoretic SFA features, their performance resembles the solution with uniform distribution ξ (circles, upper row), which clearly outperforms SFA features based on the very similar steady state distribution ξ (crosses, upper row). This is surprising, as the two distributions only differ at the borders of the square state space. Similar to Section 5.1, the uniform distribution appears nonetheless to have an advantage here.

Secondly, note that although theoretic PVF features (squares, upper row) rival the best of SFA, sampled PVF features (squares, lower row) are less successful. In the most realistic case the performance appears almost unaffected by additional features. This is consistent with our observations of the LSTD solutions in Figure 5, where sampled PVF features performed equally bad under both policies. Reward-based Krylov Features (diamonds), on the other hand, appear relatively stable through most settings and only cave in at the most realistic scenario. However, note that for LSPI *reward-based* features do not have any (empirical or theoretical) advantage over *subspace-invariant* features.

Although not exactly predicted by theory, we see this as evidence that *discrete SFA can construct better approximation spaces for LSPI than PVF or Krylov bases*. Theoretical PVF may rival SFA, but realistic sampling appears to corrupt PVF features. This advantage of SFA may be lost for other sampling policies, though. As an example, observe the strong influence of minor changes in the sampling distribution ξ on the theoretical SFA solution.

5.4 Visual Navigation-Setup

We investigate our third question on Page 2090 with a simple but realistic continuous application task. Continuous state spaces impede the use of reward-based features, but allow an analysis of the presented metric by encoding observations with PCA. Our focal idea is to compare basis construction approaches based on different metrics in the *observation space* and the underlying *state space*. A *visual navigation task* guides a robot into a designated goal area. Observations are first-person perspective images from a head mounted video camera (see Figure 2, Page 2071, for a sketch of the control process). While robot coordinates come close to encoding diffusion distances of random policies, these observations clearly do not. A comparison between PCA and SFA features in these two cases can thus illuminate the role of the observation metric in the construction of continuous basis functions. Adequacy of SFA in this task is demonstrated with a real robot (Section 5.7) and extensively compared with PVF and PCA in a realistic simulation (Section 5.6). Section 5.5 provides an evaluation of the involved sparse subsets.

5.4.1 ROBOT

We used the wheeled PIONEER 3DX robot (Figure 7a), equipped with a head mounted BUMBLEBEE camera for the experiments. The camera recorded mono RGB images from a first-person perspective of the environment in front of the robot with a 66° field of vision (Figure 7b). The robot was able to execute 3 commands: Move approximately 30cm forwards; turn approximately 45° left or right.

a) Robot b) Robot View c) Simulator View

Figure 7: a) PIONEER 3DX wheeled robot. b) A first-person perspective image from the robot camera. c) Corresponding image of the simulator.

5.4.2 ENVIRONMENT

The robot had to navigate within a rectangular $3m \times 1.8m$ area surrounded by walls, approx. $1m$ in height. We covered the walls with different wallpaper to have a rich texture (sketched in Figure 10). The scenery was well-lit by artificial light. A camera installed at the ceiling allowed us to track the robot's position for analysis. We also ran simulations of the experiment for large scale comparison. Based on photographed textures, the JAVA3D engine rendered images from any position in the simulated environment. Those images were similar to the real experiment, but not photo-realistic (Figure 7c).

5.4.3 TASK

Starting from a random start position, the robot has to execute a series of actions (move forward, turn left, turn right) that lead to an unmarked goal area in as few steps as possible without hitting the walls. Learning and control are based on the current camera image $\boldsymbol{z}_t$ and a corresponding reward signal $r_t \in \{-1, 0, +1\}$ indicating whether the robot is in the goal area ($r_t = +1$), close to a wall ($r_t = -1$) or none of the above ($r_t = 0$).

5.4.4 ALGORITHMS

The algorithms *sparse kernel principal component analysis* (SK-PCA, Section 3.2), *k-nearest-neighbor extension of proto value functions* (kNN-PVF, Section 3.3) and *regularized sparse kernel slow feature analysis* (RSK-SFA Section 3.4), were implemented[36] using a Gaussian kernel $\boldsymbol{\kappa}(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\frac{1}{2\sigma^2}\|\boldsymbol{x} - \boldsymbol{y}\|^2), \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{Z}$. To ensure the respective *function sets* from which the basis functions were chosen are roughly equivalent, we set the Gaussian width parameter $\sigma = 5$ for all algorithms. Runtime deviated at most by a factor of 2, as SFA requires two eigenvalue decompositions. The importance sampling modification of SFA described in Section 3.5 assigned a weight[37] of $\vartheta(a_{move}) = 5$ to forward movements and $\vartheta(a_{turn}) = 1$ to rotations. This balanced out *relative velocities* and ensured that the first features encode both spatial and orientational information.

36. RSK-SFA and SK-PCA select functions from a RKHS, based on any positive semi-definite kernel, for example, the Gaussian kernel. kNN-PVF are based on a k-nearest-neighbors graph with edges weighted by a Gaussian kernel of the distance between nodes.
37. We tested other weights with less detail. The results appear stable around $\vartheta(a_{move}) = 5 \pm 1$ but exact statements require an order of magnitude more simulations than we were able to provide for this article.

All discussed algorithms also require a sparse subset of the data. For optimal coverage it appears straightforward to select this subset uniformly in observation space $\mathcal{Z}$. This can be achieved with the *matching pursuit for affine hull maximization* algorithm (MP-MAH, Böhmer et al., 2012). However, we show in Section 5.5 that this intuition is wrong and in fact one should instead select the subset uniformly distributed in the true state space $\mathcal{X}$, which can be achieved by applying MP-MAH on $\mathcal{X}$ instead of $\mathcal{Z}$. As $\mathcal{X}$ is usually not known explicitly, the comparison between algorithms in Section 5.6 is performed with randomly drawn subsets.

5.4.5 SIMULATED EXPERIMENTS

Drawing actions uniformly, we generated 10 independent random walks with 35,000 samples each. The rendered images were brightness corrected and scaled down to 32×16 RGB pixels, yielding observations $\boldsymbol{z}_t \in \mathcal{Z} \subset [0,1]^{1536}$. Each training set was used to construct one feature space for each of the above algorithms with a sparse subset of 4000 samples (see Section 5.5). The resulting basis functions were applied on the corresponding training set.

The control policy was learned by LSPI on the first $p \in \{2,4,8,\ldots,2048\}$ constructed features $\phi_i : \mathcal{Z} \to \mathbb{R}$ and a constant feature $\phi_0(\boldsymbol{z}) = 1, \forall \boldsymbol{z} \in \mathcal{Z}$. The discount factor was $\gamma = 0.9$ and the goal area was located in the lower right corner with a radius of 50cm around x=260cm and y=40cm (see right plot of Figure 10). A distance to the walls of 40cm or less was punished.

The resulting policies were each tested on 200 test trajectories from random start positions. Navigation performance is measured[38] as *fraction of successful trajectories*, which avoid the wall and hit the goal in less than 50 steps.

5.4.6 ROBOT EXPERIMENT

Running the robot requires a large amount of time and supervision, preventing a thorough evaluation. For RSK-SFA the continuous random walk video of approx. 10 hours length was sampled down to approx. 1 Hz and a sparse subset of 8,000 out of 35,000 frames was selected with the MP-MAH algorithm (Böhmer et al., 2012). The first 128 RSK-SFA and one constant feature were applied on a training set of 11,656 *transitions* sampled from the same video. LSPI was trained as in the simulator experiments and evaluated on each 20 test trajectories for the lower right and for a smaller center goal with a radius of 20cm (see Figure 10).

5.5 Visual Navigation-Sparse Subset Selection

The analysis of Böhmer et al. (2012) suggests that a sparse subset uniformly distributed in observation space $\mathcal{Z}$ improves the performance of RSK-SFA. We observed the same effect on the *slowness* (not shown), but interestingly not on the *navigation performance* of the respective LSPI solution. Figure 8 plots this performance against the number of features p used for LSPI. Random subset selection (crosses) outperforms MP-MAH selection on $\mathcal{Z}$ with the correct kernel width (upward triangle) significantly. Moreover, random selection yields high performance reliably (small standard deviation), whereas subsets that are uniformly distributed in $\mathcal{Z}$ result in unpredictable behavior for large approximation spaces. Shrinking the kernel parameter σ of MP-MAH (*not* of RSK-SFA) decreases the disadvantage as the algorithm converges theoretically to random selection in the limit

38. Other measures are possible. We tested "mean number of steps to goal" and also compared those to an almost optimal policy. However, the resulting plots were qualitatively so similar that we stuck to the simplest measure.

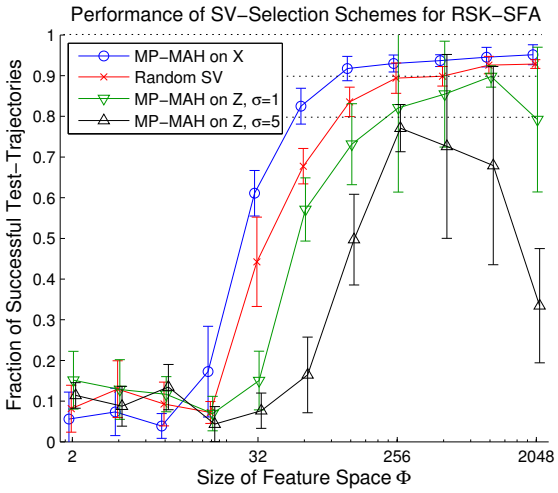


Figure 8: The influence of *sparse subset selection* for RSK-SFA on the performance of LSPI. Mean and standard deviation (over 10 independent training sets) of the navigation performance are plotted against the (logarithmic) number of features used in LSPI. Note that a subset equally distributed in the true state space $\mathcal{X}$ (circles) is most efficient and reliable, whereas for equal distribution in observation space $\mathcal{Z}$ (triangles) LSPI becomes unreliable in large approximation spaces $\mathcal{F}_\phi$.

$\sigma \to 0$ (downwards triangles). Using MP-MAH to select a subset uniformly distributed in the *true state space*[39] $\mathcal{X}$ (circles), however, demonstrates that this is not an over-fitting effect as the learned policies outperform those of random selection significantly. Section 6.3 attempts an explanation of these results and discusses some practical implications for sparse subset selection.

However, in practice $\mathcal{X}$ is usually not explicitly known. The main comparison in Section 5.6 is therefore performed with randomly drawn subsets. Note that our random walk sampled the state space $\mathcal{X}$ almost uniformly, which is the explanation for the good performance of randomly selected subsets. A random selection from biased random walks, for example, generated by other tasks, will severely decrease the navigation performance. We expect a similar effect in other sparse kernel RL methods (e.g., Engel et al., 2003; Xu, 2006).

5.6 Visual Navigation-Comparison of Algorithms

The left plot of Figure 9 shows a comparison of the effect of all discussed basis function construction algorithms on the control policy learned by LSPI. Note that the algorithms are based on the same randomly chosen sparse subset of 4000 samples: 4000 orthogonal features extracted by any algorithm span approximation space $\mathcal{F}_{\{\boldsymbol{\kappa}(\cdot,\boldsymbol{s}_i)\}_{i=1}^m}$. Therefore all algorithms perform equally well with enough ($p \geq 1024$) features. One can, on the other hand, observe that both the original RSK-SFA (crosses) and the modified algorithm (circles) outperform SK-PCA (triangles) and kNN-PVF[40]

39. Robot position and orientation coordinates for which the Euclidean metric resembles diffusion distances.
40. We tested the navigation performance based on kNN-PVF basis functions for the kNN parameters $k \in \{10, 25, 50\}$. As the results did not differ significantly, we omitted them here.

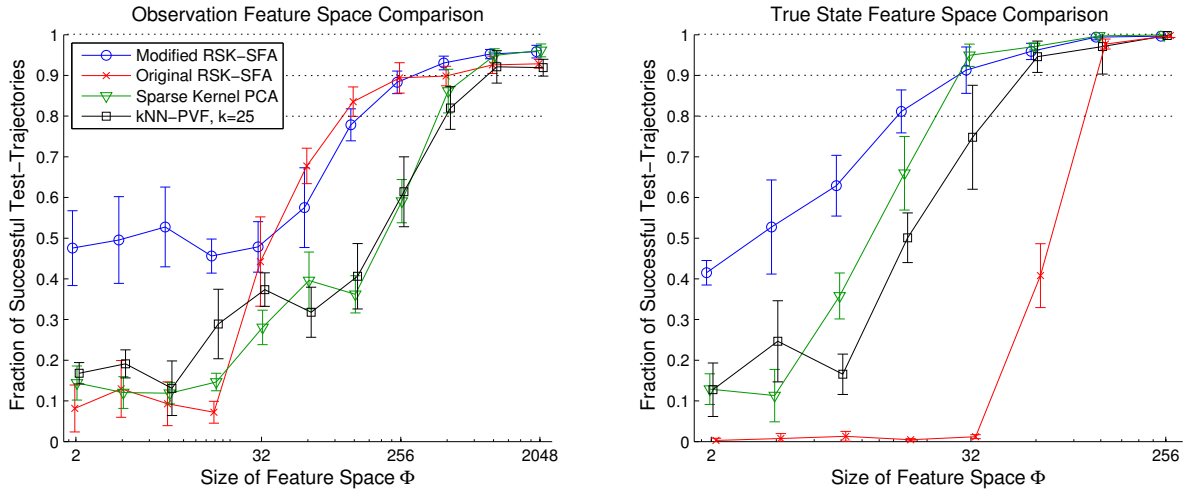


Figure 9: Comparison of subspace-invariant feature space construction algorithms on observations $\boldsymbol{z} \in \mathcal{Z}$ (camera images, left plot) and on the true state $\boldsymbol{x} \in \mathcal{X}$ (robot coordinates, right plot). Mean and standard deviation (over 10 independent training sets) of the navigation performance are plotted against the number of features used in LSPI. Note that the x-axis is logarithmic and differs between plots. The dotted lines indicate 80%, 90% and 100% performance levels.

(squares) significantly in medium sized feature spaces ($32 \leq p < 1024$ features). In particular the 80% and 90% levels of navigation performance are both reached with roughly a quarter of basis functions. We attribute this advantage to approximation spaces encoding diffusion distances rather than similarities in $\mathcal{Z}$.

Close inspection of the feature space revealed that the first RSK-SFA features encode spatial information only (not shown). This is due to the different velocities of rotations and movements of the robot in the true state space $\mathcal{X}$ (see Section 3.5 and Franzius et al., 2007). Consequentially, small feature spaces can not express policies involving rotation and thus perform poorly. The modified algorithm balances this handicap out and yields steady performance in small feature spaces ($2 \leq p < 32$ features). If the necessary orientational components are encoded in the original RSK-SFA basis functions ($p \geq 32$), both algorithms perform comparable as they span (almost) the same approximation space. We conclude that the modified RSK-SFA algorithm is the superior continuous basis function construction scheme, irrespective of feature space size p.

5.6.1 COMPARISON IN TRUE STATE SPACE $\mathcal{X}$

To confirm the above effect is due to the difference in the observed metric and the diffusion metric constructed by SFA, we run the same experiment with $\mathcal{Z} \approx \mathcal{X}$. For this purpose we applied all basis function construction algorithms on robot coordinates. The true state space $\mathcal{X}$ is supposed to be equipped with a diffusion metric, which should have a constant distance between successive states. We thus divided the spatial coordinates by an average movement of 30cm and the robots orientation by the average rotation of 45°. The kernel width was chosen $\sigma = 2$ to allow sufficient overlap.

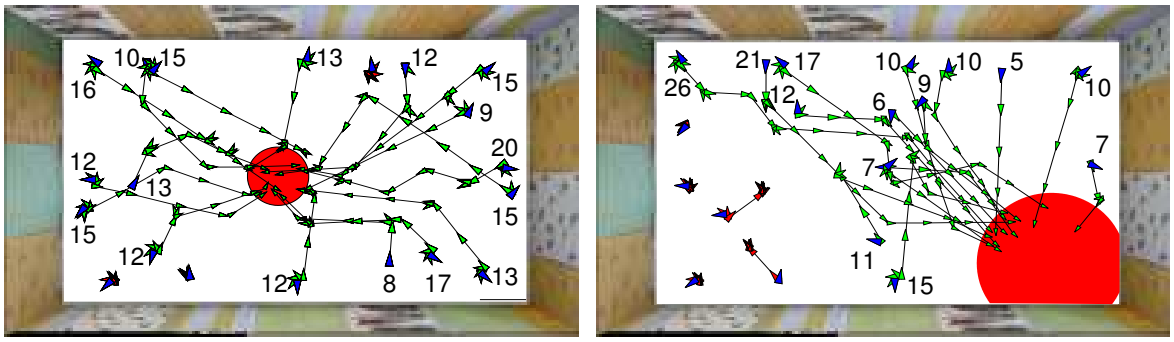


Figure 10: The recorded robot trajectories of a control policy learned by LSPI using RSK-SFA features. The plot shows 40 trajectories with random starting positions and orientation (dark triangles), aiming to hit the circular goal area. The numbers indicate the number of steps the robot required to reach the goal.

The right plot of Figure 9 shows the navigation performance of the learned LSPI policy. Due to the suitability of the observation space, a relative small number[41] of basis functions of $p \geq 128$ suffices for an almost perfect navigation performance with all algorithms. However, relative velocities influence the RSK-SFA solution as well, which renders the original algorithm (crosses) almost useless and demonstrates its sensitivity to the sampling policy. The modified algorithm (circles) performs best here as well, but demonstrates an advantage only in very small feature spaces ($2 \leq p < 32$). Together with the good performance of SK-PCA (triangles), which slightly outperforms the state-of-the-art method kNN-PVF (squares), this can be seen as evidence that methods based on Euclidean distances in $\mathcal{Z} \approx \mathcal{X}$, which are already close to diffusion distances, suffice to learn a good policy in this setup.

5.6.2 CONCLUSION

The results presented in this subsection provide ample evidence for the hypothesis that *SFA can construct better approximation spaces for LSPI than PVF or PCA*, but also demonstrates its sensitivity to the sampling policy. We have empirically shown that the novel modified RSK-SFA algorithm outperforms all continuous basis function construction schemes reviewed for this article. The advantage to baseline method PCA vanishes when the observations already conform to a diffusion metric. This suggests that SFA performs essentially PCA based on diffusion rather than Euclidean distances. It also implies an answer to our third question on Page 2090: *LSPI performance is facilitated by approximation spaces that encode diffusion distances of a uniform random policy*.

5.7 Visual Navigation-Robot Demonstration

A thorough reproduction of the experiments from Section 5.6 on a real robot is beyond the scope of this article. Measuring the navigation performance of 200 test trajectories can not easily be automated and requires an enormous amount of supervision. However, we demonstrate how our

41. Note that, in difference to the space of images, this observation space is three dimensional. Instead of (at least potentially) dimensionality reduction, these basis functions form an overcomplete basis.

key method, that is SFA in the form of RSK-SFA, performs on a real robot. After 10 LSPI iterations with 128 RSK-SFA basis functions, the control policy achieved a success rate of 75% in two separate tasks depicted in Figure 10. 17 out of 20 test trajectories hit a centered goal within 20 steps (left plot) and 13 out of 20 trajectories reached the much farther goal area in the lower right corner (right plot). The failed trajectories made at most one forward move and then started to oscillate between right and left rotations. These oscillations also appeared often in successful trajectories whenever the robot switched from one action to another. This can be seen in the large number of steps some trajectories require to reach the goal area. We attribute these oscillations to approximation errors and noise in the basis functions. Whenever the policy changes from one action to another, the respective Q-values must be close-by and small deviations from the true value can drastically influence the greedy action selection. If the wrongfully chosen action is a rotation, the correct reaction would be to rotate back. We observed these never ending oscillations in simulations as well. The problem can usually be diminished by adding more basis functions. The fact that the robot was sometimes able to break the oscillation is an indicator for noise in images and motors. This behavior could have been easily avoided by introducing some simple heuristics, for example "never rotate back" or "in doubt select the previous action". As this article investigates approximation spaces rather than optimizing the visual task itself, we omitted those heuristics in our experiments. For practical implementations, however, they should be taken under consideration.

6. Discussion

This section discusses implications of the presented results and points out open questions and potential directions of future research.

6.1 SFA Features as Basis Functions for LSPI

Theoretical predictions in Section 4 cover value function estimation with LSTD for the current sampling policy. When LSPI changes this policy, all statements become strictly invalid. Nonetheless, the results in Sections 5.3 and 5.6 demonstrate the applicability of SFA features as basis functions for LSPI. Both experiments also show that an *importance sampling* modification to SFA can yield even better results. Maybe there exists a policy τ and a state-distribution ζ which are *optimal* (in the sense of Definition 12) for at least all deterministic policies encountered during LSPI. In the experiments this appeared to be a uniform policy and a uniform state-distribution, but we can not claim any generality based on the presented evidence alone. We still want to suggest a possible explanation:

Uniform τ and ζ might be an optimal training sets for LSPI. Koller and Parr (2000) proposed this in light of a pathological 4-state chain MDP (similar to Section 5.2). LSTD weights the importance of value approximation errors and predicts thus accurate Q-values according to ζ. Using steady state distribution ξ of some sampling policy π implies that decisions at often visited states should be more reliable that those at seldom visited states. Take the optimal policy in a navigation task (like Section 5.3 or Section 5.4) as an example. Transition noise guarantees an ergodic Markov chain, but the steady state distribution will concentrate almost all mass around the goal. As a result, Q-values far away from the goal can be approximated almost arbitrarily bad and decisions become thus erratic. This is not the intended effect. To solve the task, one needs to control the approximation error of all states one will encounter until the task is complete. Without knowledge about certainty, every

decision along the way might be equally important. A uniform ζ reflects this. A similar argument can be made for a uniform τ in the context of policy iteration.

Future works might be able to identify the corresponding Radon-Nikodym derivatives $\frac{d\zeta}{d\xi} : \mathcal{Z} \to \mathbb{R}^+$ and $\frac{d\tau}{d\pi} : \mathcal{Z} \times \mathcal{A} \to \mathbb{R}^+$ and use *importance sampling* as discussed in Section 3.5. Until then the best approach appears to be sampling with a random policy and fine tuning with the modified SFA algorithm presented in this article.

6.2 Optimal Basis Functions

The analysis in Section 4.3 introduces the concept of *optimal basis functions* (Definition 12, Page 2088) to construct approximation spaces for LSTD with SFA. However, Theorem 13, Page 2089, implies that SFA features are only *optimal* for sampling policy π. This optimality is lost when policy iteration varies π, but optimizing the basis functions w.r.t. policy has no analytic solution and appears not feasible. Besides the question of feasibility, there are alternatives to the definition of *optimal basis functions* in Section 4.3. Here we suggest three possibilities:

1. Given one task $m = (\mathcal{X}, \mathcal{A}, P, R)$ only, the optimal basis functions for LSPI encode the true value function $v_m^\pi(\cdot)$ of m with all possible policies $\pi \in \Omega$, that is

$$\inf_{\phi \in L^2(\mathcal{Z},\xi)} \mathbb{E}\left[\left\| v_m^\pi - \hat{\Pi}_\xi^\phi [v_m^\pi] \right\|_\xi^2 \;\middle|\; \pi \sim \omega(\cdot) \right].$$

   As LSPI is based on a dictionary of transitions, however, the distribution ξ of the weighted projection operator $\hat{\Pi}_\xi^\phi$ corresponds to the sampling distribution instead of steady state distribution of policy π.

2. Definition 12 minimizes the mean approximation error over all expected task. An alternative would be to minimize the *worst case* bound instead, that is

$$\inf_{\phi \in L^2(\mathcal{Z},\xi)} \left(\sup_{m \in \mathcal{M}, \pi \in \Omega} \left\| v_m^\pi - \hat{\Pi}_\xi^\phi [v_m^\pi] \right\|_\xi^2 \right).$$

3. The presented *weighted Euclidean norm projection* $\hat{\Pi}_\xi^\phi$ is the most commonly used choice. However, Guestrin et al. (2001) have proposed an efficient algorithm based on *supremum norm projection* $\hat{\Pi}_\infty^\phi$ for approximation of the updated value function $\hat{B}^\pi[v](\cdot)$. It is straightforward to derive a bound analogous[42] to Tsitsiklis and Van Roy (1997) and thus to define a matching *optimality criterion*

$$\inf_{\phi \in L^2(\mathcal{Z},\xi)} \left(\sup_{m \in \mathcal{M}, \pi \in \Omega} \left\| v_m^\pi - \hat{\Pi}_\infty^\phi [v_m^\pi] \right\|_\infty \right).$$

As with the criterion of Definition 12, it might not be feasible to solve these optimization problems in practice. Future works could find feasible approximations thereof, though.

42. One needs to show that $\hat{\Pi}_\infty^\phi[\cdot]$ is a non-expansion and $\hat{B}^\pi[\cdot]$ a contraction in $\|\cdot\|_\infty$ (Bertsekas, 2007).

6.3 Sparse Subset Selection

Results in Section 5.5 suggest that sparse subsets uniformly distributed in $\mathcal{X}$ (rather than $\mathcal{Z}$) support the best basis functions for LSPI. One possible explanation is the effect of the sparse subset distribution $\chi : \mathcal{B}(\mathcal{X}) \to [0,1]$ onto the objective. In limit of an infinite subset of an infinite Markov chain, extracted features $\phi_i(\boldsymbol{z}) = \int \chi(d\boldsymbol{s})\, \alpha_i(\boldsymbol{s})\, \kappa(\boldsymbol{z},\boldsymbol{s}) = \langle \alpha_i, \kappa(\boldsymbol{z},\cdot)\rangle_\chi, \forall \boldsymbol{z} \in \mathcal{Z}$, are determined by coefficient functions $\alpha_i \in L^2(\mathcal{X},\chi)$. To determine these functions, RSK-SFA therefore approximates *generalized eigenfunctions* in $L^2(\mathcal{X},\chi)$, which are strongly affected by norm $\|\cdot\|_\chi$. Discrete SFA, on the other hand, represents every state by one unique variable, which corresponds to a uniform distribution χ in $\mathcal{X}$. It is easy to see that a uniform χ preserves the optimization problem best. Selecting a subset uniformly in $\mathcal{Z}$, however, does not generally yield a uniform χ in $\mathcal{X}$, due to the difference in Euclidean and diffusion distances. Results presented in Sections 5.3 and 5.6 demonstrate how sensitive the LSPI performance based on SFA features is to sampling policy and sampling state-distribution. Non-uniform χ will probably decrease performance similarly. It seem therefore reasonable to attribute the results of Section 5.5 to the above effect.

Nonetheless, this raises two question of practical concern:

1. how can we *select* sparse subsets uniformly in $\mathcal{X}$ and

2. how can we guarantee uniform *support*[43] in $\mathcal{Z}$?

The first question is for sparse kernelized RL algorithms (e.g., Engel et al., 2003; Xu, 2006) of utmost importance as the reported problem will most likely affect them as well. Future works must derive such an algorithm, maybe based on slowness or diffusion distances.

The related *radial basis function networks* (RBF, see, e.g., Haykin, 1998) have found an empirically answer to the second question: each support vector $\boldsymbol{s}_i$ is assigned an individual kernel width σ_i relative to the distance to its neighbors. For the Hilbert spaces of all kernels $\kappa(\cdot,\cdot)$ holds $\mathcal{H}_\kappa \subset L^2(\mathcal{Z},\xi)$ and one could thus perform inner products between Gaussian kernel functions of different width[44] in $L^2(\mathcal{Z},\xi)$. However, the math necessary to pose kernel SFA in this framework is quite advanced and calls for further research.

6.4 Visual Policies for Robots

Visual tasks are an interesting field of research as they expose elemental weaknesses in current RL approaches, for example the different Euclidean distances in observation and ideal approximation spaces discussed in this article. For applications in the field of robotics, however, the assumption of *isometry between observations and states* reaches its limits.

On the one hand, partial observability of the environment (POMDPs, Kaelbling et al., 1998) will jumble the observed transition structure and therefore all discussed feature construction methods.[45] This could be avoided by including partial *histories* of observations. For example, *predictive state representations* (PSR, Littman et al., 2001; Wingate, 2012) can construct sufficient statistics of

43. Areas in $\mathcal{Z}$ with less *support vectors* $\boldsymbol{s}_i$ will have an overall lower output of kernel functions and will thus exhibit worse generalization. This effect can be quantized for sample $\boldsymbol{z} \in \mathcal{Z}$ by the *approximation error* of the corresponding kernel function, that is, $\inf_{\boldsymbol{a}\in\mathbb{R}^m} \|\kappa(\cdot,\boldsymbol{z}) - \sum_i a_i \kappa(\cdot,\boldsymbol{s}_i)\|_{\mathcal{H}}$, and is called the *support* of $\boldsymbol{z}$.

44. Note that $\langle \kappa_a(\cdot,\boldsymbol{s}_i), \kappa_b(\cdot,\boldsymbol{s}_j)\rangle_\xi = \int \xi(d\boldsymbol{z})\, \kappa_a(\boldsymbol{z},\boldsymbol{s}_i)\, \kappa_b(\boldsymbol{z},\boldsymbol{s}_j)$ has an analytic solution if ξ is the uniform distribution, because the product of two Gaussian functions is a Gaussian function as well.

45. Basis construction might work well in a POMDP if applied on *beliefs* instead of observations, though.

observation history to solve the task without extensive knowledge of the underlying POMDP. Extensions to continuous state and observation spaces are rare (Wingate and Singh, 2007) and linear approximation not straight forward. Additionally, any traditional metric over histories will probably not reflect *diffusion distances* very well and thus perform suboptimal in techniques like LSPI (see our conclusion in Section 5.6). Therefore, the potential of feature construction techniques like *optimal basis functions* (Section 4.3) for PSR appear tremendous and should be investigated further.

Setting partial observability aside, on the other hand, every natural variable influencing the image will be treated as part of state space $\mathcal{X}$, for example angle and brightness of illumination, non-stationary objects, the view out of the window, etc. The resulting state space $\mathcal{X}$ grows exponentially in the number of these independent variables, as every combination of variables is a unique state. Besides encoding mostly useless information, this yields two problems for the method presented in this article: (i) the whole state space $\mathcal{X}$ must be *sampled* by the RL agent, which will eventually take too much time, and (ii) the basis functions must *support* the whole space, for example, a subset for sparse kernel methods must uniformly cover $\mathcal{X}$ (see Sections 5.5 and 6.3), which will eventually require too many computational resources. Both problems can not be resolved by current kernel methods to construct basis functions and/or standard linear RL approaches. *Factored MDP* approaches in combination with *computer vision* methods have the potential to solve this dilemma, though.

Using an array of highly invariant image descriptors (e.g., SIFT, Lowe, 1999), object recognition and position estimates (e.g., SLAM, Smith et al., 1990; Davison, 2003), the observation space $\mathcal{Z}$ can become much more regular. Smart choices of descriptors, for example, reacting to the window frame and not the view outside, will even make them invariant to most state dimensions in $\mathcal{X}$. If the descriptors include short-term memory, then the presented method could even be applied in a POMDP setup. However, an application of standard kernel methods takes the similarity between all descriptors at once into account and would therefore still require sampling and support on the whole space $\mathcal{X}$. *Factorizing* basis functions $\phi_i(\cdot) = \prod_j \psi_{ij}(\cdot)$, on the other hand, have full domain $\mathcal{Z}$ but are a product of multiple functions $\psi_{ij}(\cdot)$ with a domain of only a few descriptors. Integrals over $\mathcal{Z}$ break down into the product of multiple low dimensional integrals, which would each only require limited amount of sampling and support. If those factorized basis functions approximate the *optimal basis functions* discussed in Section 4.3 sufficiently close, factored MDP algorithms can be applied (Koller and Parr, 1999; Guestrin et al., 2001; Hauskrecht and Kveton, 2003; Guestrin et al., 2004). Future works must develop both the factorizing basis function construction method and some adequate factored linear RL algorithm to exploit them.

7. Summary

This article investigates *approximation spaces* for value estimation, in particularly the role of the *metric* in these spaces. This is relevant because this metric influences the Euclidean L_2 approximation error minimized by *least-squares temporal difference learning* (LSTD). We hypothesize that an *ideal Euclidean metric for LSTD should encode diffusion distances*, which reflect similar futures analogous to values. Furthermore, *slow-feature analysis (SFA) constructs the best subspace-invariant approximation spaces for LSTD*. To verify these hypotheses we compare *Krylov bases*, *proto-value functions* (PVF), *principal component analysis* (PCA) and SFA (see Section 3) theoretically and experimentally. We also derive a novel *importance sampling* modification to the SFA

algorithm to compensate for sampling imbalances of SFA. The novel algorithm showed excellent performance in Section 5.

Our theoretical analysis in Section 4 compares Krylov bases with SFA. We argue that the latter is a generalization of PVF and can construct typical *subspace-invariant* approximation spaces. Our analysis yields impressive statements for MDPs which are actually subspace-invariant under PVF or SFA. For example, Corollary 9 (Page 2086) shows a dramatically improved bound on the LSTD approximation error and Theorem 11 (Page 2087) gives a lower bound on the improvement thereof by adding another feature. However, compatible MDPs are not very common: SFA features are subspace-invariant for all MDPs with a *self-adjoint* transition operator and PVF are for all MDPs with a transition kernel visiting all neighbors uniformly. Note that the latter set is a subset of former. We argue further that *real-valued* subspace-invariant features can only be obtained for MDPs with *self-adjoint* transition operators. Both SFA and PVF can thus be interpreted as self-adjoint approximations of arbitrary MDPs, as empirical results in Section 5.1 demonstrate. This interpretation is formally supported by Theorem 13 (Page 2089). It states that SFA minimizes a mean bound over all tasks in the *same environment*, which means an arbitrary but fixed transition kernel and all possible reward functions. However, all above results hold only for the sampling policy.

It is therefore an empirical question how the discussed approximation spaces will fare when *least-squares policy iteration* (LSPI) changes the policy. We ask in Section 5:

- *How well does LSTD estimate the value function of a given Markov chain?* We predicted in Section 4.4 and verified in Sections 5.1 to 5.3: "SFA can provide better approximation spaces for LSTD than PVF".
- *How good is the performance of policies learned by LSPI based on a random policy?* Our empirical conclusion of Sections 5.3 and 5.6 is "[Modified] SFA can construct better approximation spaces for LSPI than PVF".
- *How does this performance depend on the approximation space metric?* The connection between diffusion distance and approximation error suggested itself in Sections 4.1 and 4.2. We empirically verified in Section 5.6: "LSPI performance is facilitated by approximation spaces that encode diffusion distances of a uniform random policy" because "SFA essentially performs PCA based on diffusion distances".

We see both theoretical and empirical results as evidence supporting our hypotheses. There are still too many open questions to be certain, like the undesirable dependence on the sampling distribution and other issues discussed in Section 6. However, especially the good performance with LSPI inspires hope and calls for further research.

Acknowledgments

We would like to thank Roland Vollgraf, Hannes Nickisch and Mathias Franzius for pointing us to SFA as a pre-processing method for video data. This work was funded by the *German science foundation* (DFG) within the SPP 1527 *autonomous learning*, the *German federal ministry of education and research* (grant 01GQ0850), the EPSRC grant #EP/H017402/1 (CARDyAL) and by the integrated graduate program on *human-centric communication* at Technische Universität Berlin.

Appendix A. Proofs of Section 4

For an introduction into the terminology see Section 2. The equivalency sign $\equiv$ is used to indicate that two optimization problems are solved by the same function.

Lemma 3 (repeated): Let ξ denote the steady state distribution of ergodic transition kernel P^π, which has a self-adjoint transition operator $\hat{P}^\pi = (\hat{P}^\pi)^* : L^2(\mathcal{Z},\xi) \to L^2(\mathcal{Z},\xi)$. The corresponding diffusion distance equals the Euclidean distance in the space spanned by $\psi_i^t(\cdot) := \lambda_i^t \phi_i(\cdot), \forall i \in \mathbb{N}$, where $\lambda_i \in \mathbb{R}$ and $\phi_i \in L^2(\mathcal{Z},\xi)$ are the eigenvalues and eigenfunctions of $\hat{P}^\pi$, that is

$$d_t(x,y) = \|\psi^t(x) - \psi^t(y)\|_2\,, \quad \forall x,y \in \mathcal{Z}\,, \forall t \in \mathbb{N} \setminus \{0\}\,.$$

Proof The diffusion distance $d_t(x,y)$ between states x and y is defined as the mean squared difference of the probability distributions after t steps (see Page 2084):

$$d_t(x,y) \quad := \quad \left\|\mu_x^t - \mu_y^t\right\|_\xi\,.$$

Under the formal restrictions mentioned of Assumption 1, Page 2083, $\mu_x^t \in L^2(\mathcal{Z},\xi), \forall t \in \mathbb{N} \setminus \{0\}$, and one can rewrite the inner product with arbitrary functions $f \in L^2(\mathcal{Z},\xi)$:

$$\langle \mu_x^t, f\rangle_\xi = \int \xi(dy)\left(\mu_x^t(y)\right) f(y) = \int (P^\pi)^t(dy|x)\, f(y) = (\hat{P}^\pi)^t[f](x)\,,$$

where $(\hat{P}^\pi)^t$ denotes t successive applications of the transition operator $\hat{P}^\pi$ in $L^2(\mathcal{Z},\xi)$.

$$\begin{aligned} d_t^2(x,y) &= \langle \mu_x^t, \mu_x^t\rangle_\xi - 2\langle \mu_x^t, \mu_y^t\rangle_\xi + \langle \mu_y^t, \mu_y^t\rangle_\xi \\ \langle \mu_x^t, \mu_y^t\rangle_\xi &= \int (P^\pi)^t(dz|x)\,\mu_y^t(z) \quad = \quad (\hat{P}^\pi)^t\left[\mu_y^t\right](x)\,. \end{aligned}$$

$\hat{P}^\pi$ is specified to be self-adjoint and due to the *Hilbert-Schmidt theorem* (e.g., Theorem 4.2.23 in Davies, 2007) holds for eigenfunctions $\hat{P}^\pi[\phi_i](\cdot) = \lambda_i \phi_i(\cdot)$, and $\langle \phi_i, \phi_j\rangle_\xi = \delta_{ij}, \forall i,j \in \mathbb{N}$:

$$\hat{P}^\pi[f](x) = \sum_{i=0}^{\infty} \langle f, \phi_i\rangle_\xi\, \lambda_i\, \phi_i(x)\,, \qquad \forall x \in \mathcal{Z}\,, \quad \forall f \in L^2(\mathcal{Z},\xi)\,.$$

Applying this t times, we can write

$$\begin{aligned} (\hat{P}^\pi)^t[\mu_y^t](x) &= \sum_{i=0}^{\infty} \phi_i(x)\,\lambda_i^t\,\langle \mu_y^t, \phi_i\rangle_\xi \;=\; \sum_{i=0}^{\infty} \phi_i(x)\,\lambda_i^t\,(\hat{P}^\pi)^t[\phi_i](y) \\ &= \sum_{i,j=0}^{\infty} \phi_i(x)\,\lambda_i^t\lambda_j^t\,\phi_j(y)\,\underbrace{\langle \phi_i, \phi_j\rangle_\xi}_{\delta_{ij}} \;=\; \psi^t(x)^\top \psi^t(y)\,. \end{aligned}$$

Therefore the diffusion distance $d_t(x,y)$ can be written as

$$d_t^2(x,y) = \psi^t(x)^\top\psi^t(x) - 2\psi^t(x)^\top\psi^t(y) + \psi^t(y)^\top\psi^t(y) = \|\psi^t(x) - \psi^t(y)\|_2^2\,.$$

■

CONSTRUCTION OF APPROXIMATION SPACES FOR REINFORCEMENT LEARNING

Lemma 4: Let P^{π} be an ergodic transition kernel in $\mathcal{Z}$ with steady state distribution ξ. The kernel induced by adjoint transition operator $(\hat{P}^{\pi})^*$ in $L^2(\mathcal{Z},\xi)$ is ξ-almost-everywhere an ergodic transition kernel with steady state distribution ξ.

Proof We first show that for *any* linear operator $\hat{A} : L^2(\mathcal{Z},\xi) \to L^2(\mathcal{Z},\xi)$ with *kernel* $A : \mathcal{Z} \times \mathcal{B}(\mathcal{Z}) \to \mathbb{R}^+$, that is, $\hat{A}[f](x) = \int A(dy|x)\, f(y)$, ξ-*almost-everywhere* (ξ-a.e.) in $\mathcal{Z}$ holds

$$\underbrace{\langle f, \hat{A}[\mathbf{1}]\rangle_\xi = \langle f, \mathbf{1}\rangle_\xi, \quad \forall f \in L^2(\mathcal{Z},\xi)}_{(i)} \qquad \Leftrightarrow \qquad \int \underbrace{A(dy|x) = 1, \quad \xi\text{-a.e.}}_{(ii)}$$

where $\mathbf{1}(x) = 1, \forall x \in \mathcal{Z}$, is the constant function in $L^2(\mathcal{Z},\xi)$.

Let's assume the induction $(i) \Rightarrow (ii)$ is *not* true, that is, (i) is true, but there exists a Borel set with non-zero measure ξ of states $x \in \mathcal{Z}$ that violate (ii). This set can be split up in $\int A(dy|x) > 1, \forall x \in B_+ \in \mathcal{B}(\mathcal{Z})$, and $\int A(dy|x) < 1, \forall x \in B_- \in \mathcal{B}(\mathcal{Z})$, with $\xi(B_+ \cup B_-) > 0$. Let furthermore $f \in L^2(\mathcal{Z},\xi)$ be defined as

$$f(x) = \begin{cases} 1 & ,\text{if } x \in B_+ \\ -1 & ,\text{if } x \in B_- \\ 0 & ,\text{otherwise} \end{cases},$$

which must adhere to claim (i):

$$\begin{aligned} \langle f, \hat{A}[\mathbf{1}]\rangle_\xi &= \textstyle\int_{B_+} \xi(dx) \int A(dy|x) - \int_{B_-} \xi(dx) \int A(dy|x) \\ &> \textstyle\int_{B_+} \xi(dx) - \int_{B_-} \xi(dx) \quad = \quad \langle f, \mathbf{1}\rangle_\xi . \end{aligned}$$

This is a contradiction and proves $(i) \Rightarrow (ii)$. The induction $(i) \Leftarrow (ii)$ is trivial, which proves $(i) \Leftrightarrow (ii)$.

Now we show that (i) holds for $(\hat{P}^{\pi})^*$, which is the adjoint operator to $\hat{P}^{\pi}$.

$$\langle f, (\hat{P}^{\pi})^*[\mathbf{1}]\rangle_\xi = \langle \hat{P}^{\pi}[f], \mathbf{1}\rangle_\xi = \int \underbrace{\int \xi(dx)\, P^{\pi}(dy|x)}_{\xi(dy)\text{ (ergodicity)}} f(y) = \langle f, \mathbf{1}\rangle_\xi, \quad \forall f \in L^2(\mathcal{Z},\xi).$$

This proves that the kernel of $(\hat{P}^{\pi})^*$ is a *transition kernel* ξ-almost-everywhere in $\mathcal{Z}$, which means the kernel adheres to (ii). Ergodicity can be proven using the same techniques as above:

$$\langle \mathbf{1}, \hat{A}[f]\rangle_\xi = \langle \mathbf{1}, f\rangle_\xi, \quad \forall f \in L^2(\mathcal{X},\xi) \qquad \Leftrightarrow \qquad \xi(B) = \int A(B|x)\, \xi(dx), \quad \forall B \in \mathcal{B}(\mathcal{X}).$$

$\langle \mathbf{1}, (\hat{P}^{\pi})^*[f]\rangle_\xi = \langle \mathbf{1}, f\rangle_\xi, \forall f \in L^2(\mathcal{X},\xi)$, and therefore the transition kernel of $(\hat{P}^{\pi})^*$ is ergodic with steady state distribution ξ. ■

Lemma 5: In the limit of an infinite ergodic Markov chain drawn by transition kernel P^{π} in $\mathcal{Z}$ with steady state distribution ξ holds $\mathcal{S}(f) = 2\,\langle f, (\hat{I} - \hat{P}^{\pi})[f]\rangle_\xi, \forall f \in L^2(\mathcal{Z},\xi)$.

Proof Due to a theorem of Jensen and Rahbek (2007) we can ensure that the *empirical mean* of functions over *sequences* of states converges in the limit to its expectation:

$$\begin{aligned} \mathcal{S}(f) &= \lim_{n\to\infty} \tfrac{1}{n} \sum_{t=0}^{n-1} \Big(f(x_{t+1}) - f(x_t)\Big)^2 = \iint \Big(f(y) - f(x)\Big)^2 P^{\pi}(dy|x)\, \xi(dx) \\ &= \langle f, f\rangle_\xi - 2\,\langle f, \hat{P}^{\pi}[f]\rangle_\xi + \int f^2(y) \underbrace{\int P^{\pi}(dy|x)\, \xi(dx)}_{\xi(dy)\text{ due to ergodicity}} = 2\,\langle f, (\hat{I} - \hat{P}^{\pi})[f]\rangle_\xi . \end{aligned}$$

■

Theorem 7: On average over all reward functions $r^\pi : \mathcal{Z} \to \mathbb{R}$ drawn from a white noise functional ρ, the squared norm of a Krylov basis $\{\phi_i^K\}_{i=1}^p$ from an ergodic transition kernel P^π encodes squared diffusion distances based on $\hat{P}^\pi$ up to horizon $p-1$, that is

$$d_t^2(x,y) \;=\; \mathbb{E}\left[\left\|\phi^K(x)-\phi^K(y)\right\|_{\varrho}^2 \,\middle|\, r^\pi \sim \rho\right], \quad \forall x,y \in \mathcal{Z},\, \exists \varrho \in (\mathbb{R}^+)^p,\, \forall t \in \{1,\dots,p-1\}.$$

Proof Let ξ denote the steady state distribution of transition kernel P^π. Given a reward function $r^\pi \in L^2(\mathcal{Z},\xi)$, a Krylov feature can be posed in terms of functions $\mu_x^t \in L^2(\mathcal{Z},\xi)$ (see Definition 1, Page 2083, and the Proof of Lemma 3), that is

$$\phi_i^K(x) \;:=\; (\hat{P}^\pi)^{i-1}[r^\pi](x) \;=\; \int (P^\pi)^{i-1}(dy|x)\, r^\pi(y) \;=\; \langle \mu_x^{i-1}, r^\pi\rangle_\xi, \qquad \forall x \in \mathcal{Z}.$$

One can use a property of white noise functionals (Footnote 25, p. 2085) to prove the theorem.

$$\begin{aligned}\mathbb{E}\left[\left\|\phi^K(x)-\phi^K(y)\right\|_{\varrho}^2\right] &= \sum_{i=1}^p \varrho_i\, \mathbb{E}\left[\left(\langle \mu_x^{i-1}-\mu_y^{i-1}, r^\pi\rangle_\xi\right)^2\right] \\ &= \sum_{i=1}^p \varrho_i \left\|\mu_x^{i-1}-\mu_y^{i-1}\right\|_\xi^2 \;=\; \sum_{i=1}^p \varrho_i\, d_{i-1}^2(x,y).\end{aligned}$$

$\varrho \in (\mathbb{R}^+)^p$ can be chosen freely; all diffusion distances with $t < p$ are therefore encoded. ■

Lemma 8: Let $\{\phi_i\}_{i=1}^p$ denote any p SFA features from a MDP with self-adjoint transition operator, then the LSTD fixed point $f^\pi = \hat{\Pi}_\xi^\phi\left[\hat{B}^\pi[f^\pi]\right]$ and the projection of true value function $v^\pi = \hat{B}^\pi[v^\pi]$ coincide, that is

$$f^\pi(x) \;=\; \hat{\Pi}_\xi^\phi[v^\pi](x) \;=\; \sum_{i=1}^p \langle r^\pi, \phi_i\rangle_\xi\, \tau_i\, \phi_i(x), \quad \forall x \in \mathcal{Z}, \quad \tau_i := \left(1-\gamma+\tfrac{\gamma}{2}\mathcal{S}(\phi_i)\right)^{-1}.$$

Proof Let $\psi_i \in L^2(\mathcal{Z},\xi)$ denote the (due to the Hilbert-Schmidt theorem orthonormal) eigenfunctions of $\hat{P}^\pi$ and λ_i the corresponding eigenvalues, that is, $\hat{P}^\pi[\psi_i] = \lambda_i \psi_i$. $\{\psi_i\}_{i=1}^\infty$ is a full basis of $L^2(\mathcal{Z},\xi)$ and thus $r^\pi = \sum_{i=1}^\infty \langle r^\pi, \psi_i\rangle_\xi \psi_i$ with $\phi_i = \psi_i, \forall i \le p$ and $\langle \phi_i, \psi_j\rangle_\xi = \delta_{ij}$. From this we can conclude $\hat{\Pi}_\xi^\phi[\psi_i] = \phi_i, \forall i \le p$, and $\hat{\Pi}_\xi^\phi[\psi_i] = 0, \forall i > p$. Due to the geometric series and Lemma 5 also holds $\sum_{t=0}^\infty \gamma^t \lambda_i^t = (1-\gamma\lambda_i)^{-1} = \tau_i$. Therefore,

$$\begin{aligned}\hat{\Pi}_\xi^\phi[v^\pi] &= \sum_{t=0}^\infty \gamma^t\, \hat{\Pi}_\xi^\phi\left[(\hat{P}^\pi)^t[r^\pi]\right] \;=\; \sum_{i=1}^\infty \langle r^\pi, \psi_i\rangle_\xi \sum_{t=0}^\infty \gamma^t\, \hat{\Pi}_\xi^\phi\left[(\hat{P}^\pi)^t[\psi_i]\right] \\ &= \sum_{i=1}^p \langle r^\pi, \phi_i\rangle_\xi\, \phi_i \sum_{t=0}^\infty \gamma^t \lambda_i^t \;=\; \sum_{i=1}^p \langle r^\pi, \phi_i\rangle_\xi\, \tau_i\, \phi_i, \\ f^\pi &= \hat{\Pi}_\xi^\phi\left[\hat{B}^\pi[f^\pi]\right] \;=\; \sum_{t=0}^\infty \gamma^t \left(\hat{\Pi}_\xi^\phi[\hat{P}^\pi]\right)^t \left[\hat{\Pi}_\xi^\phi[r^\pi]\right] \;=\; \sum_{i=1}^p \langle r^\pi, \phi_i\rangle_\xi\, \phi_i \sum_{t=0}^\infty \gamma^t \lambda_i^t.\end{aligned}$$

■

CONSTRUCTION OF APPROXIMATION SPACES FOR REINFORCEMENT LEARNING

Theorem 11: Let ξ be the steady state distribution on $\mathcal{Z}$ of a MDP with policy π and a self-adjoint transition operator in $L^2(\mathcal{Z},\xi)$. Let further $\Phi_p = \{\phi_i\}_{i=1}^p$ be any set of p SFA features and $v^\pi \in L^2(\mathcal{Z},\xi)$ the true value of the above MDP. The improvement of the LSTD solution $f^{(p)} := \hat{\Pi}_\xi^{\Phi_p}\left[\hat{B}^\pi[f^{(p)}]\right]$ by including the p'th feature is bounded from below by

$$\left\|v^\pi - f^{(p-1)}\right\|_\xi - \left\|v^\pi - f^{(p)}\right\|_\xi \quad \geq \quad \tfrac{1-\gamma}{2}\,\frac{\langle r^\pi,\phi_p\rangle_\xi^2}{\|r^\pi\|_\xi}\,\tau_p^2, \qquad \tau_p := \left(1-\gamma+\tfrac{\gamma}{2}\mathcal{S}(\phi_p)\right)^{-1}.$$

Proof Let $\{\phi_i\}_{i=1}^\infty$ denote the extension of Φ_p to a full orthonormal basis of $L^2(\mathcal{Z},\xi)$, that is, $f(\cdot) = \sum_{i=1}^\infty \langle f,\phi_i\rangle_\xi\,\phi_i(\cdot), \forall f \in L^2(\mathcal{Z},\xi)$. Lemma 8 shows that $f^{(p)} = \hat{\Pi}_\xi^{\Phi_p}[v^\pi], \forall p \in \mathbb{N}$,

$$\left\|v^\pi - \hat{\Pi}_\xi^{\Phi_{p-1}}[v^\pi]\right\|_\xi^2 - \left\|v^\pi - \hat{\Pi}_\xi^{\Phi_p}[v^\pi]\right\|_\xi^2 = \left\|\sum_{i=p}^\infty \langle v^\pi,\phi_i\rangle_\xi\phi_i\right\|_\xi^2 - \left\|\sum_{i=p+1}^\infty \langle v^\pi,\phi_i\rangle_\xi\phi_i\right\|_\xi^2 = \langle v^\pi,\phi_p\rangle_\xi^2.$$

Note further that $\langle v^\pi,\phi_p\rangle_\xi^2 = \langle r^\pi,\phi_p\rangle_\xi^2\,\tau_p^2$, and that one can bound the norm of v^π:

$$\left\|v^\pi\right\|_\xi \;=\; \left\|\sum_{t=0}^\infty \gamma^t\,(\hat{P}^\pi)^t[r^\pi]\right\|_\xi \;\leq\; \sum_{t=0}^\infty \gamma^t\left\|(\hat{P}^\pi)^t[r^\pi]\right\|_\xi \;\leq\; \sum_{t=0}^\infty \gamma^t\left\|r^\pi\right\|_\xi \;=\; \frac{1}{1-\gamma}\left\|r^\pi\right\|_\xi.$$

The first equality follows from the proof of Lemma 14, the first inequality from the property of norms, the last inequality from Lemma 15 and the last equality from the geometric series. Using the identity $a^2-b^2=(a-b)(a+b)$ and inequality $\|v^\pi - \hat{\Pi}_\xi^{\Phi_p}[v^\pi]\|_\xi \leq \|v^\pi\|_\xi$,

$$\left\|v^\pi - f\right\|_\xi - \left\|v^\pi - f'\right\|_\xi \;=\; \frac{\left\|v^\pi - f\right\|_\xi^2 - \left\|v^\pi - f'\right\|_\xi^2}{\left\|v^\pi - f\right\|_\xi + \left\|v^\pi - f'\right\|_\xi} \;\geq\; \frac{\langle v^\pi,\phi_p\rangle_\xi^2}{2\,\|v^\pi\|_\xi} \;\geq\; \tfrac{1-\gamma}{2}\,\frac{\langle r^\pi,\phi_p\rangle_\xi^2}{\|r^\pi\|_\xi}\,\tau_p^2,$$

where $f := f^{(p-1)}$ and $f' := f^{(p)}$. ■

Theorem 13: For any infinite ergodic Markov chain with steady state distribution ξ over state space $\mathcal{Z}$, SFA selects features from function set $\mathcal{F} \subset L^2(\mathcal{Z},\xi)$ that minimize an upper bound on the optimality criterion of Definition 12 for sampling policy π and discount factor $\gamma > 0$, under the assumption that the mean-reward functions $r^\pi : \mathcal{Z} \to \mathbb{R}$ are drawn from a white noise functional in $L^2(\mathcal{Z},\xi)$.

Proof Lemma 14 shows that for all $0 \leq \gamma < 1$ the operator $(\hat{I} - \gamma\hat{P}^\pi) : L^2(\mathcal{Z},\xi) \to L^2(\mathcal{Z},\xi)$ is invertible. Let $\hat{\Theta}^\pi$ denote this inverse operator. For any mean reward function $r^\pi \in L^2(\mathcal{Z},\xi)$ the corresponding true value function $v_r^\pi \in L^2(\mathcal{Z},\xi)$ can be determined analytically: $v_r^\pi(x) = r^\pi(x) + \gamma\hat{P}^\pi[v_r^\pi](x) = \hat{\Theta}^\pi[r^\pi](x), \forall x \in \mathcal{Z}$. According to the assumptions, the mean reward functions $r^\pi \sim \rho$ are distributed as a white noise functional, which implies

$$\int \langle f, r^\pi\rangle_\xi^2\,\rho(dr^\pi) \;=\; \langle f,f\rangle_\xi, \quad \forall f \in L^2(\mathcal{Z},\xi).$$

We will now show that the SFA objective minimizes an upper bound on Definition 12 and thus also minimizes the bound of Tsitsiklis and Van Roy (1997).

$$
\begin{aligned}
& \inf_{\phi\in(\mathcal{F})^p} \mathbb{E}\left[\left\|v_r^\pi - \hat{\Pi}_\xi^\phi[v_r^\pi]\right\|_\xi^2 \,\middle|\, r^\pi \sim \rho\right] \\
\equiv\ & \sup_{\phi\in(\mathcal{F})^p} \mathbb{E}\left[\langle v_r^\pi, \hat{\Pi}_\xi^\phi[v_r^\pi]\rangle_\xi \,\middle|\, r^\pi \sim \rho\right] \\
\equiv\ & \sup_{\phi\in(\mathcal{F})^p} \sum_{i=1}^p \mathbb{E}\left[\langle \phi_i, \hat{\Theta}^\pi[r^\pi]\rangle_\xi^2 \,\middle|\, r^\pi \sim \rho\right] && \text{s.t. } \langle \phi_i,\phi_j\rangle_\xi = \delta_{ij},\ \forall i,j && (C_{ij}{:=}\delta_{ij}) \\
\equiv\ & \sup_{\phi\in(\mathcal{F})^p} \sum_{i=1}^p \left\|(\hat{\Theta}^\pi)^*[\phi_i]\right\|_\xi^2 && \text{s.t. } \langle \phi_i,\phi_j\rangle_\xi = \delta_{ij},\ \forall i,j && \text{(assumption)} \\
\equiv\ & \inf_{\phi\in(\mathcal{F})^p} \sum_{i=1}^p \left\|\left(\hat{I}-\gamma(\hat{P}^\pi)^*\right)[\phi_i]\right\|_\xi^2 && \text{s.t. } \langle \phi_i,\phi_j\rangle_\xi = \delta_{ij},\ \forall i,j && \text{(lemma 16)} \\
\leq\ & \inf_{\phi\in(\mathcal{F})^p} \sum_{i=1}^p \left((1+\gamma^2)\underbrace{\langle \phi_i,\phi_i\rangle_\xi}_{1} - 2\gamma\langle \phi_i, \hat{P}^\pi[\phi_i]\rangle_\xi\right) && \text{s.t. } \langle \phi_i,\phi_j\rangle_\xi = \delta_{ij},\ \forall i,j && \text{(lemmas 15\&4)} \\
\equiv\ & \inf_{\phi\in(\mathcal{F})^p} -2\gamma\sum_{i=1}^p \langle \phi_i, \hat{P}^\pi[\phi_i]\rangle_\xi && \text{s.t. } \langle \phi_i,\phi_j\rangle_\xi = \delta_{ij},\ \forall i,j \\
\overset{(*)}{\equiv}\ & \inf_{\phi\in(\mathcal{F})^p} 2\sum_{i=1}^p \langle \phi_i, (\hat{I}-\hat{P}^\pi)[\phi_i]\rangle_\xi && \text{s.t. } \begin{array}{rcll}\langle \phi_i,\phi_j\rangle_\xi &=& \delta_{ij}, & \forall i,j \\ \langle \phi_i, 1\rangle_\xi &=& 0, & \forall i\end{array} && (\gamma{>}0) \\
\equiv\ & \inf_{\phi\in(\mathcal{F})^p} \sum_{i=1}^p \mathcal{S}(\phi_i) && \text{s.t. } \begin{array}{rcll}\langle \phi_i,\phi_j\rangle_\xi &=& \delta_{ij}, & \forall i,j \\ \langle \phi_i, 1\rangle_\xi &=& 0, & \forall i\end{array} && \text{(Lemma 5).}
\end{aligned}
$$

The equivalency marked $(*)$ holds because the infimum is the same for all $\gamma > 0$. In the limit $\gamma \to 1$, however, $(\hat{I}-\gamma\hat{P}^\pi)$ is not invertible. The fist (constant) right eigenfunction of $\hat{P}^\pi$ must thus be excluded by the zero mean constraint. The last equation is the SFA optimization problem, which therefore minimizes an upper bound on the optimality criterion of Definition 12. ■

Lemma 14 *For an ergodic transition operator $\hat{P}^\pi$ in $L^2(\mathcal{Z},\xi)$ with steady state distribution ξ and $0 \leq \gamma < 1$, the operator $(\hat{I}-\gamma\hat{P}^\pi)$ is invertible. Let $\hat{\Theta}^\pi$ denote the inverse,*

$$\left\|(\hat{I}-\gamma\hat{P}^\pi)\left[\hat{\Theta}^\pi[f]\right] - f\right\|_\xi = 0, \quad \forall f \in L^2(\mathcal{Z},\xi).$$

Proof Let $(\hat{P}^\pi)^t$ denote the composition of t operators $\hat{P}^\pi$ with $(\hat{P}^\pi)^0 = \hat{I}$ and let $\hat{\Theta}^\pi := \lim_{n\to\infty}\sum_{t=0}^{n-1}\gamma^t(\hat{P}^\pi)^t$, then $\forall f \in L^2(\mathcal{Z},\xi)$:

$$
\begin{aligned}
& \left\|\left(\hat{I}-\gamma\hat{P}^\pi\right)\left[\hat{\Theta}^\pi[f]\right] - f\right\|_\xi = \lim_{n\to\infty}\left\|\left(\hat{I}-\gamma\hat{P}^\pi\right)\left[\sum_{t=0}^{n-1}\gamma^t(\hat{P}^\pi)^t[f]\right] - f\right\|_\xi \\
= \lim_{n\to\infty}\left\|\left(\hat{I}-\gamma^n(\hat{P}^\pi)^n\right)[f] - f\right\|_\xi &= \lim_{n\to\infty}\gamma^n\left\|(\hat{P}^\pi)^n[f]\right\|_\xi \quad \leq \quad \lim_{n\to\infty}\gamma^n\|f\|_\xi \quad = \quad 0.
\end{aligned}
$$

The inequality holds because Lemma 15 shows that $\hat{P}^\pi$ is a non-expansion and the last equality because all $f \in L^2(\mathcal{Z},\xi)$ are bounded from above, that is, $\|f\|_\xi < \infty$. ■

Lemma 15 *An ergodic transition operator $\hat{P}^{\pi}$ in $L^2(\mathcal{Z},\xi)$ with steady state distribution ξ is a non-expansion, defined as*

$$\left\|\hat{P}^{\pi}[f]\right\|_{\xi} \leq \left\|f\right\|_{\xi}, \qquad \forall f \in L^2(\mathcal{Z},\xi).$$

Proof Due to Jensens inequality[46] we have

$$\left\|\hat{P}^{\pi}[f]\right\|_{\xi}^2 = \langle \hat{P}^{\pi}[f], \hat{P}^{\pi}[f]\rangle_{\xi} = \int \xi(dx)\Big(\int P^{\pi}(dy|x)f(y)\Big)^2$$
$$\leq \int \underbrace{\int \xi(dx)P^{\pi}(dy|x)}_{\xi(dy) \text{ due to ergodicity}} f^2(y) = \langle f,f\rangle_{\xi} = \left\|f\right\|_{\xi}^2, \quad \forall f \in L^2(\mathcal{Z},\xi).$$

■

Lemma 16 *For any invertible linear operator $\hat{A} : L^2(\mathcal{Z},\xi) \to L^2(\mathcal{Z},\xi)$ holds*

$$\sup_{\substack{f\in L^2(\mathcal{Z},\xi),\\ \langle f,f\rangle_{\xi}=1}} \left\|\hat{A}[f]\right\|_{\xi} \equiv \inf_{\substack{f\in L^2(\mathcal{Z},\xi),\\ \langle f,f\rangle_{\xi}=1}} \left\|\hat{A}^{-1}[f]\right\|_{\xi}.$$

Proof The *operator norm* of an operator $\hat{A}$ in $L^2(\mathcal{Z},\xi)$ is defined as

$$\|\hat{A}\|_{\xi} := \sup_{f\in L^2(\mathcal{Z},\xi)} \left\{ \frac{\|\hat{A}[f]\|_{\xi}}{\|f\|_{\xi}} \,\middle|\, f \neq 0 \right\} = \sup_{f\in L^2(\mathcal{Z},\xi)} \left\{ \|\hat{A}[f]\|_{\xi} \,\middle|\, \|f\|_{\xi} = 1 \right\}.$$

Using the one-to-one transformation $f \leftarrow \hat{A}^{-1}[f]$ in the equivalency marked $(*)$,

$$\sup_{\substack{f\in L^2(\mathcal{Z},\xi),\\ \langle f,f\rangle_{\xi}=1}} \left\|\hat{A}[f]\right\|_{\xi} \equiv \|\hat{A}\|_{\xi} \equiv \sup_{f\in L^2(\mathcal{Z},\xi)} \left\{ \tfrac{\|\hat{A}[f]\|_{\xi}}{\|f\|_{\xi}} \,\middle|\, f \neq 0 \right\} \equiv \inf_{f\in L^2(\mathcal{Z},\xi)} \left\{ \tfrac{\|f\|_{\xi}}{\|\hat{A}[f]\|_{\xi}} \,\middle|\, f \neq 0 \right\}$$
$$\overset{(*)}{\equiv} \inf_{f\in L^2(\mathcal{Z},\xi)} \left\{ \tfrac{\|\hat{A}^{-1}[f]\|_{\xi}}{\|f\|_{\xi}} \,\middle|\, f \neq 0 \right\} \equiv \inf_{\substack{f\in L^2(\mathcal{Z},\xi),\\ \langle f,f\rangle_{\xi}=1}} \left\|\hat{A}^{-1}[f]\right\|_{\xi}.$$

■

References

P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Advances in Neural Information Processing Systems*, pages 1–8, 2007.

M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

46. If P^{π} is a transition kernel, Jensens inequality (e.g., Boyd and Vandenberghe, 2004) allows $\big(\int P^{\pi}(dy|x)f(y)\big)^2 \leq \int P^{\pi}(dy|x)f^2(y), \forall x \in \mathcal{Z}, \forall f \in L^2(\mathcal{Z},\xi)$.

P. Berkes and L. Wiskott. Slow feature analysis yields a rich repertoire of complex cell properties. *Journal of Vision*, 5:579–602, 2005.

D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, 3rd edition, 2007.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

W. Böhmer, S. Grünewälder, H. Nickisch, and K. Obermayer. Generating feature spaces for linear algorithms with regularized sparse kernel slow feature analysis. *Machine Learning*, 89(1-2): 67–86, 2012.

M. Bowling, A. Ghodsi, and D. Wilkinson. Action respecting embedding. In *International Conference on Machine learning*, 2005.

J. A. Boyan and A. W. Moore. Generalization in reinforcement learning: safely approximating the value function. In *Advances in Neural Information Processing Systems*, pages 369–376, 1995.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

S. J. Bradtke and A. G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1/2/3):33–57, 1996.

R. Coifman, S. Lafon, A. Lee, M. Maggioni, B. Nadler, F. Warner, and S. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data. Part I: diffusion maps. *Proceedings of the National Academy of Science*, 102(21):7426 – 7431, May 2005.

L. Csató and M. Opper. Sparse on-line Gaussian processes. *Neural Computation*, 14(3):641–668, 2002.

E. Brian Davies. *Linear Operators and their Spectra*. Cambridge University Press, 2007.

A. J. Davison. Real-time simultaneous localization and mapping with a single camera. In *IEEE International Conference on Computer Vision*, volume 2, page 1403, 2003.

D.P. de Farias and B. Van Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51(6):850–865, 2003.

Y. Engel, S. Mannor, and R. Meir. Bayes meets Bellman: the Gaussian process approach to temporal difference learning. In *International Conference on Machine Learning*, pages 154–161, 2003.

K. Ferguson and S. Mahadevan. Proto-transfer learning in Markov decision processes using spectral methods. In *ICML Workshop on Transfer Learning*, 2006.

E. Ferrante, A. Lazaric, and M. Restelli. Transfer of task representation in reinforcement learning using policy-based proto-value functions. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, 2008.

P. Földiák. Learning invariance from transformation sequences. *Neural Computation*, 3(2):194–200, 1991.

M. Franzius, H. Sprekeler, and L. Wiskott. Slowness and sparseness leads to place, head-direction, and spatial-view cells. *PLoS Computational Biology*, 3(8):e166, 2007.

S. Grünewälder and K. Obermayer. The optimal unbiased value estimator and its relation to LSTD, TD and MC. *Machine Learning*, 83:289–330, 2011.

C. Guestrin, D. Koller, and R. Parr. Max-norm projections for factored MDPs. In *International Joint Conference on Artificial Intelligence*, pages 673–682, 2001.

C. Guestrin, M. Hauskrecht, and B. Kveton. Solving factored MDPs with continuous and discrete variables. In *Uncertainty in Artificial Intelligence*, pages 235–242, 2004.

M. Hauskrecht and B. Kveton. Linear program approximations for factored continuous-state Markov decision processes. In *Advances in Neural Information Processing Systems*, pages 895–902, 2003.

S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2nd edition, 1998. ISBN 978-0132733502.

G. E. Hinton and S. Osindero. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

H. Holden, B. Øksendal, J. Ubøe, and T. Zhang. *Stochastic Partial Differential Equations*. Springer Science+Business Media, 2nd edition, 2010.

O. C. Jenkins and M. J. Mataric. A spatio-temporal extension to Isomap nonlinear dimension reduction. In *International Conference on Machine Learning*, 2004.

S. T. Jensen and A. Rahbek. On the law of large numbers for (geometrically) ergodic Markov chains. *Economic Theory*, 23:761–766, 2007.

S. Jodogne and J. H. Piater. Closed-loop learning of visual control policies. *Journal of Artificial Intelligence Research*, 28:349–391, 2007.

L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

J. Kober and J. Peters. Policy search for motor primitives in robotics. In *Advances in Neural Information Processing Systems*, 2009.

D. Koller and R. Parr. Computing factored value functions for policies in structured MDPs. In *International Joint Conference on Artificial Intelligence*, pages 1332–1339, 1999.

D. Koller and R. Parr. Policy iteration for factored MDPs. In *Uncertainty in Artificial Intelligence*, pages 326–334, 2000.

V. R. Kompella, M. D. Luciw, and J. Schmidhuber. Incremental slow feature analysis: adaptive low-complexity slow feature updating from high-dimensional input streams. *Neural Computation*, 24 (11):2994–3024, 2012.

G. D. Konidaris, S. Osentoski, and P.S. Thomas. Value function approximation in reinforcement learning using the Fourier basis. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence*, 2011.

M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

S. Lange and M. Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *International Joint Conference on Neural Networks*, pages 1–8, 2010.

R. Legenstein, N. Wilbert, and L. Wiskott. Reinforcement learning on slow features of high-dimensional input streams. *PLoS Computational Biology*, 6(8):e1000894, 2010.

M. L. Littman, R. S. Sutton, and S. Singh. Predictive representations of state. In *In Advances In Neural Information Processing Systems 14*, 2001.

D. G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, 1999.

M. Luciw and J. Schmidhuber. Low complexity proto-value function learning from sensory observations with incremental slow feature analysis. In *International Conference on Artificial Neural Networks and Machine Learning*, volume III, pages 279–287. Springer-Verlag, 2012.

S. Mahadevan and B. Liu. Basis construction from power series expansions of value functions. In *Advances in Neutral Information Processing Systems*, pages 1540–1548, 2010.

S. Mahadevan and M. Maggioni. Proto-value functions: a Laplacian framework for learning representations and control in Markov decision processes. *Journal of Machine Learning Research*, 8: 2169–2231, 2007.

S. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions On Signal Processing*, 41:3397–3415, 1993.

N. Mehta, S. Natarajan, P. Tadepalli, and A. Fern. Transfer in variable-reward hierarchical reinforcement learning. *Machine Learning*, 73:289–312, 2008.

R. Parr, C. Painter-Wakefield, L. Li, and M. Littman. Analyzing feature generation for value-function approximation. In *International Conference on Machine Learning*, 2007.

R. Parr, L. Li, G. Taylor, C. Painter-Wakefiled, and M. L. Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *International Conference on Machine Learning*, 2008.

K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series 6*, 2(11):559 – 572, 1901.

M. Petrik. An analysis of Laplacian methods for value function approximation in MDPs. In *International Joint Conference on Artificial Intelligence*, pages 2574–2579, 2007.

M. Petrik and S. Zilberstein. Robust approximate bilinear programming for value function approximation. *Journal of Machine Learning Research*, 12:3027–3063, 2011.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

M. Reed and B. Simon. *Methods of Modern Mathematical Physics I: Functional Analysis*. Academic Press, 1980. ISBN 0-12-585050-6.

B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002. ISBN 978-0262194754.

B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

R. Smith, M. Slef, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous Robot Vehicles*. Springer-Verlag, 1990.

A.J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings to the 17th International Conference Machine Learning*, pages 911–918, 2000.

M. Snel and S. Whiteson. Multi-task reinforcement learning: Shaping and feature selection. In *European Workshop on Reinforcement Learning*, pages 237–248, 2011.

N. Sprague. Predictive projections. In *International Joint Conference on Artificial Intelligence*, pages 1223–1229, 2009.

H. Sprekeler. On the relationship of slow feature analysis and Laplacian eigenmaps. *Neural Computation*, 23(12):3287–3302, 2011.

Y. Sun, F. Gomez, M. Ring, and J. Schmidhuber. Incremental basis construction from temporal difference error. In *International Conference on Machine Learning*, pages 481–488, 2011.

R. S. Sutton. Generalization in reinforcement learning: successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems*, pages 1038–1044, 1996.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10:1633–1685, 2009.

J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global framework for nonlinear dimensionality reduction. *Science*, 290:2319 – 2323, 2000.

S. Thrun and A. Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the Fourth Connectionist Models Summer School*, 1993.

J.N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, 1997.

G. Wahba. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, 1990.

C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

D. Wingate. Predictively defined representations of state. In M. Wiering and M. van Otterlo, editors, *Reinforcement Learning: State-of-the-Art*, pages 415–439. Springer-Verlag Berlin Heidelberg, 2012.

D. Wingate and S. P. Singh. On discovery and learning of models with predictive representations of state for agents with continuous actions and observations. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1128–1135, 2007.

L. Wiskott. Slow feature analysis: a theoretical analysis of optimal free responses. *Neural Computation*, 15(9):2147–2177, 2003.

L. Wiskott and T. Sejnowski. Slow feature analysis: unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, 2002.

X. Xu. A sparse kernel-based least-squares temporal difference algorithm for reinforcement learning. In *Advances in Natural Computation*, volume 4221 of *Lecture Notes in Computer Science*, pages 47–56. Springer Berlin / Heidelberg, 2006.

A.3 Böhmer et al., KI 29(4):353–362 (2015a)

The article *Autonomous learning of state representations for control: an emerging field aims to autonomously learn state representations for reinforcement learning agents from their real-world sensor observations* (Böhmer et al., 2015) has been published in a special issue on *autonomous learning* in the journal **Künstliche Intelligenz**[3] (KI) on the 5th of March 2015.

The article reviews various approaches to *autonomously learn* state representations from real-world sensor measurements of robots. The first part, written by my collaborators, reviews their heuristically motivated approaches to train neural networks with auto-encoders. In the second part, I show that almost all learning mechanisms extract the same *isomorphic state space* and vary only in the associated *metric*. I summarize my analysis of *slow feature analysis* (SFA, Böhmer et al., 2013), which shows that under the right conditions the *slowness* objective enforces a nearly optimal metric. I also discuss the disadvantages of SFA for non-linear reinforcement learning algorithms and conclude with an outlook how to overcome these challenges. My conclusions are summarized in Chapter 4 and the notation follows closely Böhmer et al. (2013) in Appendix A.2 on Page 145.

The authors contributed in the following way:

- *Wendelin Böhmer* has written Sections 4 to 7 of the article.
- *Jost Tobias Springenberg* has written Sections 1 to 3 together with JB.
- *Joschka Boedecker* has written Sections 1 to 3 of the article with JTS.
- *Martin Riedmiller* gave comments to the article
- *Klaus Obermayer* gave comments to the article.

[3] The final publication is available at Springer via `http://dx.doi.org/10.1007/s13218-015-0356-1` .

Published in **KI - Künstliche Intelligez 29(4):352–362 (2015)**.
The final publication is available at Springer via http://dx.doi.org/10.1007/s13218-015-0356-1 .

Autonomous Learning of State Representations for Control

An emerging field aims to autonomously learn state representations for reinforcement learning agents from their real-world sensor observations.

Wendelin Böhmer · Jost Tobias Springenberg · Joschka Boedecker · Martin Riedmiller · Klaus Obermayer

Received: 03.02.2015 / Accepted: 05.03.2015 / Published online: 19.03.2015

Abstract This article reviews an emerging field that aims for autonomous reinforcement learning (RL) *directly* on sensor-observations. Straightforward *end-to-end* RL has recently shown remarkable success, but relies on large amounts of samples. As this is not feasible in robotics, we review two approaches to *learn* intermediate *state representations* from previous experiences: *deep auto-encoders* and *slow-feature analysis*. We analyze theoretical properties of the representations and point to potential improvements.

Keywords end-to-end reinforcement learning · representation learning · deep auto-encoder networks · slow feature analysis · autonomous robotics

1 Introduction

Feature engineering is an important part in solving machine learning problems. In the case of reinforcement learning agents for real-world control tasks, this task can be especially challenging as it involves finding representations from raw sensor measurements (such as images or laser scan data) which are rich enough to describe the full *state* of the agent (and its environment), while still being compact enough to enable fast convergence of the learning algorithm.

Taking learning in robotics applications as an example, the agent is faced with sensory data in form of a stream of relatively high dimensionality, e.g. from $4 - 7$ values of joint angles plus their velocities (and possibly accelerations) for robotic arms, up to 50+ joint angle values on very complex robots, hundreds of sensor values measured by pressure-sensitive robotic skins, or on the order of thousands to millions of pixels from camera images, nowadays often augmented with depth information.

As mentioned, the challenge is to extract meaningful bits of information, i.e. a low-dimensional representation relevant to the learning task from this high-dimensional load of data. Traditionally, an approach to achieve this is to use the insights of the system designer into the task at hand, and specify the most important task variables explicitly. Algorithms for extracting their values from the data stream are then usually hand-coded, and parameters are tuned manually. A concrete example from a robotic soccer application is the extraction of task variables for learning to dribble a soccer ball. In [44], the authors specified the velocities in x and y direction, rotation speed, and heading angle of the robot (with respect to some target) as the relevant pieces of information, which could be extracted from the raw data of the camera (using hand coded computer vision approaches) and odometry sensors of a robot in the RoboCup MidSize Soccer League. This representation contained sufficient information in order for the robot to learn quick, space-efficient turns with the ball without loosing it in the process. Similar hand-coded feature extraction pipelines underpin most successful applications of RL to robotics (we refer the interested reader to [21] for a recent review).

However, it is often a tedious, time-consuming process to find these useful features for a task at hand since the design choices underlying this step typically do not generalize over different tasks or control domains. Another limiting factor can be the experience of the system designer who needs to have enough insight into the problem in order to decide what constitutes useful features. Methods that learn representations from data automatically have emerged as a promising alternative; in

This work was partially funded by the German science foundation (DFG) within the priority program SPP 1527.

W. Böhmer · K. Obermayer
Neural Information Processing Group,
Technische Universität Berlin, Sekr. MAR 5-6
Marchstrasse 23, 10587 Berlin, Germay
E-mail: wendelin@ni.tu-berlin.de

J. T. Springenberg · J. Boedecker · M. Riedmiller
Machine Learning Lab, Universtät Freiburg

particular, automatic feature learning with deep convolutional neural networks is now the dominating method when it comes to image classification problems. In the field of robotics, these methods have found application as well, as we describe below. Note that despite the focus on one sensory modality in our examples, i.e. image data, similar problems arise for other modalities, such as tactile information used for solving robot control problems with contacts and even simple readings from joint encoders.

In the following, we will first present *representation-free* approaches, followed by *deep auto-encoder networks* and *slow feature analysis* as emerging techniques to learn state representations. A comparison of approximation properties and practical concerns in learning representations is completed by a discussion of open questions and promising directions of future research.

2 End-to-end reinforcement learning

The most direct approach to avoiding tedious hand crafting of representations is to learn a non-linear control policy directly operating on the raw sensory inputs (often referred to as *end-to-end* learning). The goal of end-to-end learning in RL is to directly train a non-linear function approximator that represents the target function (e.g. a Q-function or a policy π) we care about.

One prominent recent example of this line of work is the Deep Q-Networks (DQN) approach by [37]. Here the authors learn a policy for playing several ATARI games with human level performance directly from pixel images captured from an ATARI simulator. DQN trains a convolutional neural network (CNN) to approximate the highly non-linear Q-function with an online gradient descent approach from a large amount of interactions with the system. Formally this minimizes the squared *Bellman error* [2,51,19], of function $\tilde{Q}_\theta$ with parameters θ, for all training samples $\{z^t, a^t, r^t\}_{t=1}^n$:

$$\min_\theta \sum_{t=1}^{n-1} \Big(r^t + \gamma \max_a \tilde{Q}_\theta(z^{t+1}, a) - \tilde{Q}_\theta(z^t, a^t) \Big)^2, \qquad (1)$$

where $z^t \in \mathcal{Z} \subset \mathbb{R}^d$ denotes the sensor observation at time t a^t, r^t the chosen action and collected reward and γ is the discounting factor.

It should be noted, that the idea of using neural networks as non-linear (Q)-value function approximators has a long history in RL. Successes have, however, mainly been restricted to complicated control problems with low-dimensional inputs [43] and simple toy examples[1] [45,28]. There are several factors allowing DQN to succeed where previous attempts failed: (i) the advent of modern GPU computing allows for training extremely large neural networks on huge datasets (several million example images were used to train DQN). (ii) DQN makes use of a large deep CNN as compared to traditional shallow neural networks, thus having a large representational power while constraining representable functions with insights from image processing. (iii) DQN uses experience replay to circumvent sampling problems[2] that plague online RL.

In the last years several researchers have also considered end-to-end learning of behavioral policies π, represented by general function approximators. First attempts towards this include an actor-critic formulation termed NFQ-CA [14], in which both the policy π and the Q-function are represented using neural networks and learning proceeds by back-propagating the error from the Q-network directly into the policy network – which was, however, only applied to low dimensional control problems. Closely connected to NFQ-CA, recent work on policy gradient algorithms revealed a principled formulation for end-to-end learning of deep neural network policies using a deterministic policy gradient formulation [48]. Notably, this approach was successfully applied to control the high-dimensional problem of controlling a 30-DOF robotic arm. Other recent attempts towards learning neural network policies include applications of joint trajectory optimization and neural network policy learning for robotics problems (see e.g. [27,38]) as well as playing Go from raw visual input [32] and learning attention policies for object recognition [36].

The main advantage of end-to-end learning for RL is that it results in policies, without the need for intermediate representation learning. The main drawback is that end-to-end learning of deep neural network policies from raw visual input often requires thousands or even millions of samples, making these approaches extremely data hungry. This is often not feasible in robotics.

3 Deep representation of states

Keeping the number of necessary training samples low is also the key motivation for heuristic representations. Faced with some (usually continuous) set of possible *observations* $\mathcal{Z} \subset \mathbb{R}^d$ from d sensors, an expert uses his considerable knowledge of the task to specify a mapping $\phi : \mathcal{Z} \to \mathcal{X}$ into a p dimensional *representation* $\mathcal{X} \subset \mathbb{R}^p$. This "knowledge", however, has been inferred from previous experiences.

[1] Perhaps with the exception of TD-Gammon [53], which relied heavily on a well chosen representation as input.

[2] Sampling from trajectories with changing policies leads to non-stationary training distributions and prevents convergence in online gradient descent algorithms.

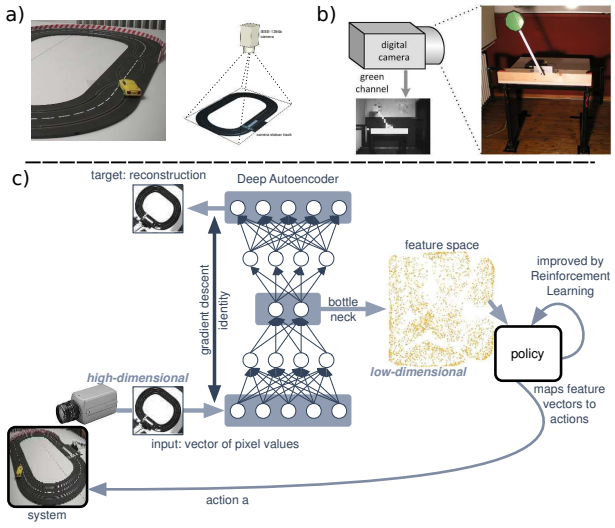


Fig. 1 Depiction of the experimental setup for learning with DFQ for (a) the slot-car scenario [25] and (b) the visual pole task [35]. (c) detailed visualization of representation learning and Q-learning as performed by DFQ.

Analogously, an autonomous robot could infer such a mapping from experiences in previous tasks or passive observations. While numerous unsupervised learning algorithms exist in literature, which could be employed to learn such a representation, the high-dimensionality of the sensory input in the problems we consider makes effective representation learning difficult. This is why only few learning algorithms have been successful in this setting [26,25,16,5].

To exemplify this, let us take a look at one such approach, that was successfully applied to learn a sensory representation for control of a *slot-car racer* on a track [25], as well as an inverted pendulum [35], using pixel information extracted from a high-resolution camera only: the *deep fitted Q* (DFQ) algorithm. The general setup of DFQ for both problems is depicted in Figure 1. It consists of an unsupervised learning component, that first learns to extract the necessary information from the images, as well as a reinforcement learning component, that carries out the task (steering the slot-car around the track or swinging up the pendulum) based on the learned representation. The figure already illustrates several key aspects necessary for successful learning:

1. Since control requires interaction with a real system, learning has to be data efficient. Only a few hundred experiments (resulting in few thousand observations) can be carried out without causing excessive wear of the system.
2. In order to enable efficient reinforcement learning, the learned representation $\phi(z) \in \mathcal{X} \subset \mathbb{R}^p$ has to be of low intrinsic dimensionality p.
3. We assume that each observation $\mathbf{z} \in \mathcal{Z}$ captures all information necessary to describe the state of the system we aim to control.
4. How the state is represented in $\mathcal{X}$ depends on an *optimization problem*, in the case of DFQ an *auto-encoder* [15,3] of the observations $\mathcal{Z}$.

The DFQ algorithm employs *deep neural networks* to represent both the *encoder* $\phi : \mathcal{Z} \to \mathcal{X}$ and the inverse *decoder* $\psi : \mathcal{X} \to \mathcal{Z}$. Pre-training with the auto-encoder minimizes the *least-squares reconstruction error* of all training samples $\{\boldsymbol{z}^t\}_{t=1}^n \subset \mathcal{Z}$ for the first layer:

$$\min_{\phi,\psi} \sum_{t=1}^{n} \left\| \psi\big(\phi(\boldsymbol{z}^t)\big) - \boldsymbol{z}^t \right\|_2^2 \qquad \text{s.t.} \quad p \ll d\,. \qquad (2)$$

After convergence, the parameters of the trained layer are frozen and its output is used as reconstruction targets for the next layer. Reducing the number of artificial neurons on each successive layer yields a low-dimensional representation $\mathcal{X}$, that is able to reconstruct the observation and thus must contain the *state*. After this layer-wise training all weights of the complete, stacked, auto-encoder are jointly fine-tuned to improve the reconstruction (post-training).

After learning the encoder network ϕ, a *fitted Q algorithm* is applied to learn an approximate Q function $\tilde{Q} : \mathcal{X} \to \mathbb{R}$ of the control problem[3]. A plethora of function approximators can be utilized to represent $\tilde{Q}$. In the case of DFQ, a clustering based algorithm was used. When applied to the slot-car task and the visual pole swing-up task, this algorithm successfully learns a controller solving the task from raw sensory input. In the case of the slot-car racer the policy is competitive to an experienced human on this task (see Table 1 for a performance comparison).

Similar to other successful applications of unsupervised representation learning to RL, the DFQ approach has several potential weaknesses which we will further discuss in Section 7: (i) in contrast to end-to-end learning, the state representation learned by DFQ is not reward-based and we hence cannot expect the resulting representation to be "goal directed"; (ii) learning auto-encoders for inputs with high variability (i.e. many objects of relevance) can be hard. Both problems could potentially be addressed by adding explicit regularization terms to the formulation from Eq. (2). An interesting recent attempt in that direction is described in [17]. Additionally recent research in the machine learning

[3] The back-propagated Bellman-error could potentially also be used to fine-tune the representation, but both [25] and [35] chose not to adapt the representation to the task.

Performance on Slot-Car

Controller	Time per round	Crash-free
Random	-	No
Constant velocity	6.408s	Yes
Experienced Human	$\approx 3s$	Yes
DFQ	**1.869s**	Yes

Performance on visual swing-up

Controller	Average reward	Success
Random	-1	No
Fitted-Q (True State)	$\mathbf{-0.15 \pm 0.01}$	Yes
DFQ	-0.205 ± 0.075	Yes

Table 1 Performance of the DFQ Algorithm on the slot-car benchmark (top) and on the visual swing-up task (bottom) over 20 trials. The best performance is marked bold. On the swing-up task DFQ performs about as good as a policy learned using fitted-Q iteration using the true state information (pole angular position and velocity). On the slot-car task DFQ completes a successful lap faster than an experienced human. Tables adapted from [25,35].

community has resulted in several auto-encoder variants and deep probabilistic models that might be easier to train (thus addressing problem (ii)) [54,20,42].

Despite these drawbacks DFQ also comes with advantages over end-to-end learning: since the auto-encoder merely learns to reconstruct sampled observations and it can be fed with samples generated by any sampling policy for any task, is thus less susceptible to non-stationarity of the training data. And more importantly, since the learned non-linear embedding into the representation space $\mathcal{X}$ is low-dimensional, an RL algorithm based on $\mathcal{X}$ can succeed using only few samples.

4 Slow feature analysis as state representation

Less restricted by the pitfalls of non-linear optimization, the field of discrete RL has developed their own methodology to learn representations. These fall roughly in two categories: *reward-based* and *subspace-invariant* features [39]. The first type aims to represent the propagated reward (Krylov-bases [41], BEBF [40] and BARB [33]). This allows context-dependent representations, but prohibits the (re-)use of samples from other sources. The second type is reward independent and can *transfer* knowledge from previous tasks [52]. In the following we will introduce two prime examples of such representations to compare them in theory and practice.

Proto-value functions (PVF, [34]) use *Laplacian eigenmaps* (LEM, [1]), a technique from *spectral clustering* [47], to provide a state representation $\mathcal{X}$. The learned features are the smallest eigenvectors of a *connectivity graph*, that is generated from a random-walk through the discrete state space. As this graph is identical for all tasks with the same transition model, PVF can use training data from previous tasks to reduce training time [11,12], similar to *shaping* [49]. Motivated by spectral clustering, a continuous extension of PVF extracts the eigenvectors of a *k-nearest-neighbor* graph [34], to apply PVF to high-dimensional *observation spaces*.

Recent work demonstrated that the unsupervised technique *slow feature analysis* (SFA, [57,56]) approximates LEM in a similar way as PVF [50]. In difference to PVF, however, the spectral encoding of SFA representations is based on the *transition probability* rather than the *connectivity* of states [5]. Instead of extracting eigenvectors explicitly, SFA minimizes the *slowness* of an observed sequence of observations $\{\boldsymbol{z}_t\}_{t=1}^n$ in $\mathcal{X}$:

$$\min_{\{\phi_i\}} \sum_{i=1}^{p} \underbrace{\sum_{t=1}^{n-1} \Big(\phi_i(\boldsymbol{z}^{t+1}) - \phi_i(\boldsymbol{z}^t)\Big)^2}_{\text{slowness of } \phi_i}, \tag{3}$$

under some constraints to avoid trivial or correlated solutions [57]. This objective can be implemented with non-linear function classes, for example, deep convolution neural networks [57,13] or sparse kernel methods [4]. Both methods need to be implicitly or explicitly regularized, though, as SFA is prone to over-fitting.

4.1 Theoretical analysis

Analysis of unrestricted SFA solutions and experiments in simulated environments have demonstrated that non-linear SFA is able to extract the underlying three dimensional state space of a wheeled robot in a static environment [13]. Furthermore, the learned feature space approximates a *Fourier-basis* in this space, which is known to be a universal basis for continuous functions. In this light it is not surprising that there have been many successful attempts using non-linear SFA to learn RL representations from observations, ranging from simple top-down perspective pixel-environments [26,30] to simulated and real-world first-person perspective robot experiments [5].

From a theoretical point of view, SFA and PVF both approximate *subspace-invariant features*. This classification has its origin in the the analysis of approximation errors in linear RL [39]. Here subspace-invariant features induce no errors when the future reward is *propagated* back in time. It can be shown that under these conditions the *least-squares temporal difference* algorithm (LSTD, [9]) is equivalent to supervised *least-squares regression* of the true value function [5]. However, this is only possible for the class of RL-tasks with a *self-adjoint transition model*. As this class is very rare, both SFA and PVF substitute a self-adjoint ap-

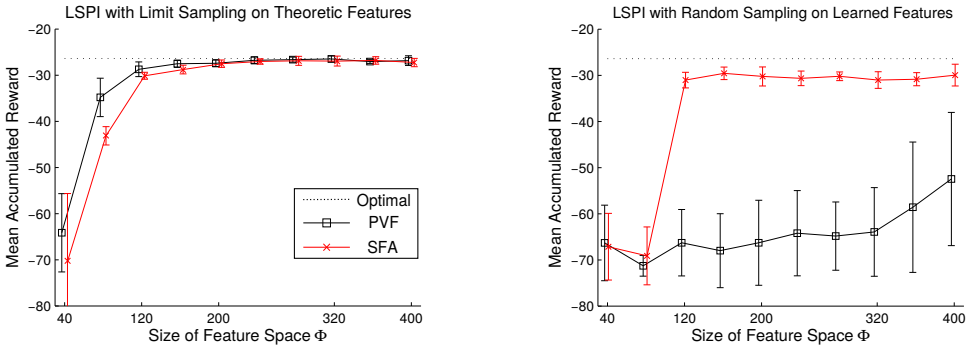


Fig. 2 Performance of LSPI [23] (y-axis) with learned SFA and PVF representations of varying size (x-axis), in the discrete *puddle-world* task. In the left plot representations and policy iteration are based on all state-action pairs, whereas the right plot uses a randomly drawn Markov chain as training set. Mean and standard deviation are w.r.t. state space sizes $\{20 \times 20, 25 \times 25, \ldots, 50 \times 50\}$. The dotted line marks the performance of the optimal policy. Figure modified from [5].

proximation of the transition model to compute *almost* subspace-invariant representations[4].

An analysis of the optimal solution[5] shows that SFA approximates eigenfunctions of the symmetrized transition operator [50]. Moreover, with a Gaussian prior for the reward, one can show that SFA representations minimize a bound on the expected LSTD error of *all tasks in the same environment* [5]. However, as the solution depends on the sampling distribution, straight forward application for transfer learning is less obvious than in the case of PVF. Future works may rectify this with some sensible importance sampling, though.

[4] See [5] for a comparison of SFA/PVF subspace-invariance.

[5] In the limit of infinite training samples, the optimization problem can be analyzed by function analysis in $L^2(\mathcal{Z}, \xi)$.

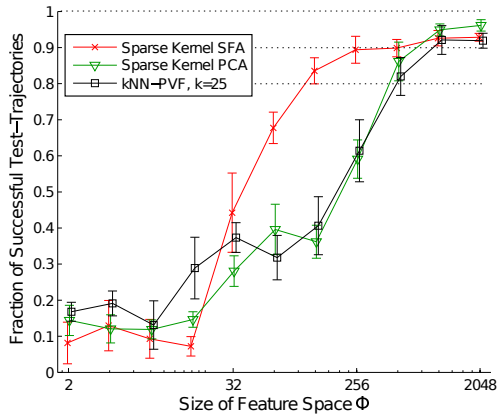


Fig. 3 Mean and standard deviation of the navigation performance in 10 independent training sets (y-axis) in a simulated *robot navigation* task based on first-person perspective images. The LSPI representations of varying size (logarithmic x-axis) have been learned by continuous sparse kernel SFA, PCA and PVF. Dotted lines represent the 80%, 90% and 100% performance levels. Figure modified from [5].

4.2 Empirical comparison

All theoretical arguments presented in this section hold strictly for LSTD only, that is, become invalid when for example *least squares policy iteration* (LSPI, [23]) changes the policy. How do SFA/PVF perform here?

The left side of Figure 2 compares the LSPI performance (mean accumulated reward) of SFA and PVF representations in discrete *puddle-world* tasks [8] of various sizes. Details can be found in [5]. The training set contained all state-action pairs and shows therefore the best possible performance. As number of features increases (x-axis), both representations similarly approach the optimal performance (dotted line). Under ideal conditions both representations are thus coequal. A different picture emerges when the training set is drawn by a random walk, as shown on the right side of Figure 2. Here the SFA representation caves in only slightly, whereas PVF essentially fails[6].

In the more realistic scenario of first-person perspective images from a wheeled robot, rendered in a virtual environment, the difference is less obvious but still visible. Figure 3 shows the LSPI performance (fraction of successful test-trajectories) based on sparse kernel SFA [4], sparse kernel PCA [46] and PVF of a k-nearest-neighbor graph [34]. All algorithms used the same set of 4000 support vectors with the same Gaussian kernel. For further details see [5]. Note that SFA representations reach both the 80% and 90% performance levels with only a quarter of the representation size required by both PVF and PCA, which behave very similar.

In summary, SFA and PVF approximate subspace-invariant features, which are especially suited for linear algorithms like LSTD. In particular SFA seems to be one of the prime contestants for good representations.

[6] It is not entirely clear *why* empirical PVF fail here. One can observe that ideal PVF features have higher frequencies than SFA's, which may be harder to estimate empirically.

5 Properties of good state representations

This section will compare the *properties* of a representation, with no concern how they are *learned*. In general, approximate RL methods based on a value function have common demands on the representation $\mathcal{X}$:

(a) $\mathcal{X}$ must be *Markov* (no partial observability),
(b) $\mathcal{X}$ must be able to *represent the true value* of the current policy well enough for policy improvement,
(c) $\mathcal{X}$ must *generalize* the learned value-function to unseen states with *similar futures* and
(d) $\mathcal{X}$ must be *low dimensional* for efficient estimation.

All discussed methods aim to construct such a representation $\mathcal{X}$ from a high-dimensional observation space $\mathcal{Z}$ with a non-linear mapping $\phi : \mathcal{Z} \to \mathcal{X}$. If $\mathcal{Z}$ is only partially observable, however, $\mathcal{X}$ cannot have the Markov property (a). There are several possibilities to make $\mathcal{Z}$ Markov, for example temporal embedding [37], *liquid state machines* [31], *belief states* [18] or *predictive state representations* [29, 55], which each come with their own disadvantages. Temporal embedding, for example, implicitly increases the state space uneccessarily by the history of actions. An adequate discussion of these techniques is beyond the scope of this article and we will in the following assume that $\mathcal{Z}$ has the Markov property.

5.1 Isomorphic representations

We intend to train the representation with data from previous tasks or passive observations, without knowledge of the reward function we will face. As each state may be rewarding, each state must therefore be distinguishable in $\mathcal{X}$ to represent the value function of the current policy. In this case, any representation $\mathcal{X}$ must be an *isomorphism* of $\mathcal{Z}$. The only conceptual difference between isomorphic $\mathcal{X}$ is the *metric* that measures how similar two states are in training and generalization. For example, take the set of all images $\mathcal{Z}$ a camera can record in a specific environment. If this environment is static and diverse enough, each image $z \in \mathcal{Z}$ will correspond to exactly one camera position $x \in \mathcal{X}$ and vice versa. $\mathcal{X}$ is therefore an isomorphism of $\mathcal{Z}$. Note the conceptual equivalence to auto-encoders in Section 3.

Representation $\mathcal{X}$ does not have to be a *vector-space*, though. The mapping $\phi : \mathcal{Z} \to \mathcal{X}$ can also define a *manifold* of representations $\mathcal{X}$, embedded in some p dimensional *feature space* $\mathbb{R}^p$. For linear RL algorithms (like LSTD) such a representation is necessary to fulfill demand (b). The embedding must provide a *functional basis* of the underlying state, able to approximate the value function sufficiently. An example would be a Fourier expansion of the above camera positions [22]. As mentioned in Section 4, non-linear SFA will approximate this feature space [13], which explains the good performance with linear RL algorithms [5].

5.2 Representations encode metrics

Non-linear algorithms like *neural fitted Q-iteration* (NFQ [43]) or *deep Q-networks* (DQN [37]), on the other hand, can in principle work on *any* isomorphic representation $\mathcal{X}$. These algorithms will nonetheless benefit in training and generalization from some embeddings. The reason is the aforementioned *metric*. For example, when a robot navigates between multiple rooms, the underlying space of positions is two-dimensional. Positions on both sides of a wall would *appear* to be very similar, but yet have *dissimilar* values. Any approximate RL algorithm will benefit from a representation that maps these two points far away from each other, but keeps positions the robot can immediately travel to similar.

The Euclidean metric in feature space $\mathcal{X} \subset \mathbb{R}^p$ should therefore be proportional to the travel-distances between states. In stochastic environments one can only compare probability distributions over future states based on a random policy, called *diffusion distances*. It can be shown that SFA approximates eigenfunctions of the symmetrized transition operator, which encode diffusion distances [5]. SFA features are therefore a good representation for non-linear RL algorithms as well.

In summary, SFA representations $\mathcal{X}$ seem *in principle* the better choice for both linear and non-linear RL: non-linear SFA extracts eigenfunctions of the transition model P^π, which are the *same* for every isomorphic observation space $\mathcal{Z}$, encode a diffusion metric that *generalizes* to states with similar futures and approximates a *Fourier basis* of the (unknown) underlying state space.

6 How to learn good state representations

Section 5 argued that SFA representations have outstanding properties for RL. However, in this section we will discuss conceptual problems that limit SFAs applicability and how deep networks can overcome them.

6.1 Slowness and representation size

Learning SFA representations provides challenges that reduce the practical benefits considerably. *Slowness* (Equation 3) is in the limit of an infinite training set

$$\min_{\{\phi_i\}} \sum_{i=1}^{p} \mathbb{E}\left[\left(\phi_i(z') - \phi_i(z) \right)^2 \,\middle|\, \begin{array}{l} z \sim \xi(\cdot) \\ z' \sim P^\pi(\cdot|z) \end{array} \right]. \qquad (4)$$

SFA features depend therefore on the *distribution* ξ of the training samples $z \in \mathcal{Z}$ and on the *sampling policy* π. Empirical studies suggest that both should be as close to uniform distributions as possible [5], which is not feasible if one reuses data from previous tasks. The effect could in principle be balanced out by *importance sampling*, and optimal importance weights are a promising field of future research.

Moreover, a Fourier basis as approximated by SFA grows exponential in the underlying state dimensionality. Linear algorithms, which depend on this basis to approximate the value function, are therefore restricted to low dimensional problems with few or no variables unrelated to the task. Non-linear RL algorithms, on the other hand, could work in principle well with only the first few SFA features of each state-dimension/variable. The *order* in which these variables are encoded as SFA features, however, depends on the *slowness of that variable*. This can in practice lead to absurd effects. Take our example of a wheeled robot, living in a naturally lit room. The underlying state space that the robot can control is three-dimensional, but the image will also depend on illumination, that is, the position of the sun. As the sun is by far the slowest variable, the majority of SFA features will encode its position and (non-existing) interactions with other state variables. This effect makes most of the representation $\mathcal{X}$ unrelated to the controllable state in the presence of *slow distractors*.

6.2 Slowness and deep networks

In contrast, *deep auto-encoder networks*, as introduced in Section 3, do not suffer the above problem. They encode the state-variables according to their influence in reconstructing observations $z \in \mathcal{Z}$, not the slowness of their representation $x \in \mathcal{X}$. On the negative side, deep representations encode metrics that can be arbitrarily bad for function approximation. Also, auto-encoders minimize the squared error over *all* input dimensions of $\mathcal{Z}$ equally. This can produce incomplete representations if a robot, for example, combines observations from a camera with measurements from multiple joints. Due to the large number of pixels, small improvements in the reconstruction of the image can outweigh large improvements in the reconstruction of the joint positions.

Recently an interesting compromise has been proposed: training a neural network with an objective that combines slowness with *predictability* of the successive state [16]. The learned representations are similarly compact as those from an auto-encoder, but encode a diffusion metric. Figure 4 compares these representations at the example of the slot-car task (see Figure 1). The right plot shows how the above objective captures the

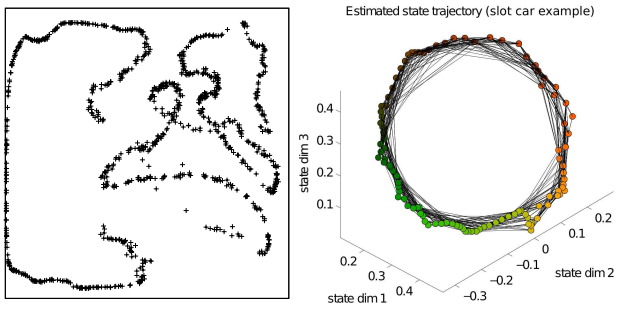


Fig. 4 Representations of the *slot-car* task (Fig. 1) learned by a *deep auto-encoder* from physical experiments (left, from [25]) and by a neural network trained with the objectives *slowness* and *predictability* from a simulated experiment (right, from [16]). Plots reproduced with authors permissions.

circular state metric of the task, which we attribute to its slowness term. The representation of the auto-encoder in the left plot, on the other hand, maps various dissimilar states close-by each other, which complicates the approximation of different values in these states. Furthermore, one can also extend this framework to suppress state-variables *not* related to the task by enforcing the predictability of rewards [17]. This demonstrates one possible way to deal with *slow distractors*.

6.3 Conclusion

We discussed two basic approaches to learn state representations from observations, *slow-feature analysis* and *deep auto-encoder networks*. SFA construct representations with a metric and embedding that is especially suited for linear RL, but also works very well with non-linear RL algorithms. The same properties curse SFA with huge feature spaces, when faced with a high-dimensional underlying state space. Deep networks, on the other hand, produce very compact representations. These do not control the encoded metric, which often complicates value approximation considerably.

Future works may train deep auto-encoder networks similar to [25], that enforce a suitable metric on the representation layer, similar to [16,17]. This could marry the generalization of deep networks with the preferable representation properties of SFA.

7 Outlook

This article presented the emerging field of autonomously learning state representations directly from observations. There are still many unresolved questions; in the following we will present some of the most pressing concerns and most promising research directions.

7.1 Unsolved problems of learning representations

In our opinion, the biggest challenge today is the *curse-of-dimensionality* of observed manifold $\mathcal{Z} \subset \mathbb{R}^d$. Notice that the dimensionality d of observation space $\mathbb{R}^d$ does not pose a problem, if the underlying state space $\mathcal{S}$ is low-dimensional. Adding sensors or pixels does not change the isomorphic state and methods like SFA will approximate the same representation. If the underlying state $\mathcal{S}$ is high dimensional, on the other hand, the discussed learning methods need to sample *all* regions of $\mathcal{S}$. We believe this to be the main reason why unsupervised learning of deep representations does not work for large environments like ATARI games [37], which remain the domain of end-to-end reinforcement learning.

Take the example of uncontrollable distractors like blinking lights or activity outside a window. Each distractor is an independent variable of the isomorphic state $\mathcal{S}$, and to learn an isomorphic representation $\mathcal{X}$ requires thus samples from all possible *combinations* of these variables. The required training samples grow exponentially in the number of distractors. By sacrificing isomorphy, one could suppress some of those variables similar to [17] and thus reduce the training set drastically. However, it is not clear how to *identify* controllable variables without restricting representable tasks.

As discussed for SFA in Section 6, all successful methods rely on *averages* over a training set. If the training distribution ξ or sampling policy π are biased, as one would expect when the observations are generated from previous tasks, large parts of the state space $\mathcal{S}$ would be inadequately encoded. Auto-encoders (see Section 3) are less affected by this, as they do not depend on the sampling policy. A training set uniformly sampled in $\mathcal{S}$ (not in $\mathcal{Z}$) would probably yield the most general representations [5]. If one could estimate such a distribution, its *inverse* would yield optimal importance sampling weights. Alternatively, one could change the objectives from the L^2 to the L^∞ norm. However, optimization in this norm is usually more expensive.

7.2 Factored representations and symbolic RL

If we can overcome the above challenges with some mapping $\phi : \mathcal{Z} \to \mathcal{X}$, we will still face a major underlying problem: it is not feasible to learn tasks in a representation $\mathcal{X}$ of the *full* isomorphic state $\mathcal{S}$ of most environments. $\mathcal{S}$ may simply be too large for sampling. Take the example of a household robot living in a kitchen: each object in the room represents multiple variables of $\mathcal{S}$ that the robot can interact with and that may be necessary for one of the many tasks the robot faces. On the other hand, solving such a task in a representation $\hat{\mathcal{X}}$ of some subset $\hat{\mathcal{S}} \subset \mathcal{S}$ could be feasible, as most of these tasks depend only on few variables.

$$\mathcal{Z} \overset{\text{map}}{\longmapsto} \mathcal{S} \overset{\text{task}}{\longmapsto} \hat{\mathcal{S}} \overset{\text{model}}{\longmapsto} \hat{\mathcal{X}} \overset{\text{rl}}{\longmapsto} \hat{Q}\,.$$

Learning a representation can thus be seen as learning a map ϕ into a set of *almost independent* variables s_i, which compose the isomorphic world state $\boldsymbol{s} \in \mathcal{S}$. Some separate procedure could then choose a small subset of variables $\hat{\mathcal{S}} \subset \mathcal{S}$ that is sufficient to solve the task at hand. A direct approach $\hat{Q} : \hat{\mathcal{S}} \to \mathbb{R}$ to learn the Q-value still requires many samples to estimate the transitions and interactions of variables. Instead, one could learn *independent transition models* for each s_i and *interaction models* between variables. For example, dishes and tables can be manipulated independently, unless one is placed upon the other. Except for those few states, the joint transition model is factorized [7]. For each selection $\hat{\mathcal{S}}$, one can therefore *generate* a representation $\hat{\mathcal{X}}$ that encodes the metric of the joint transition model (with or without interactions) and *learn* $\hat{Q} : \hat{\mathcal{X}} \to \mathbb{R}$, if possible by exploiting factorization [6].

Moreover, a group of variables can also be seen as an instance of a *class of objects*. Similar to clustering, classes could be learned by enforcing that all variables obey a small set of transition models. In a world of dishes and tables, for example, most transitions should be well predictable using only two transition models. Labeling each variable as part of one *object* also allows to use *symbolic RL* algorithms (e.g. relational RL [10, 24]) to select the subset $\hat{\mathcal{S}}$. It may be a long shot, but one could imagine a hierarchical framework that plans far ahead using symbolic RL and solves detailed subproblems in metric subspaces.

Most important, however, is the possibility to *regularize* the optimization of $\phi : \mathcal{Z} \to \mathcal{S}$ by sparse transition models. In the face of large underlying state spaces $\mathcal{S}$, regularization is necessary to keep the demand for samples feasible. If one learns the *state* $\mathcal{S}$ and its *transitions* at the same time, the sparsity of the above transition models would strongly constrain the possible mappings ϕ and therefore require much less training samples. Such a joint optimization will be challenging, but has the potential to *break* the curse-of-dimensionality.

Acknowledgements We would like to thank Sebastian Höfer and Rico Jonschkowski for many fruitful discussions.

References

1. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation **15**(6), 1373–1396 (2003)

2. Bellman, R.E.: Dynamic programming. Princeton University Press (1957)
3. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. In: Advances in Neural Information Processing Systems (2007)
4. Böhmer, W., Grünewälder, S., Nickisch, H., Obermayer, K.: Generating feature spaces for linear algorithms with regularized sparse kernel slow feature analysis. Machine Learning **89**(1-2), 67–86 (2012)
5. Böhmer, W., Grünewälder, S., Shen, Y., Musial, M., Obermayer, K.: Construction of approximation spaces for reinforcement learning. Journal of Machine Learning Research **14**, 2067–2118 (2013)
6. Böhmer, W., Obermayer, K.: Towards structural generalization: Factored approximate planning. ICRA Workshop on Autonomous Learning (2013). URL `http://autonomous-learning.org/wp-content/uploads/13-ALW/paper_1.pdf`
7. Boutilier, C., Dean, T., Hanks, S.: Decision-theoretic planning: Structural assumptions and computational leverage. Journal of Artificial Intelligence Research **11**, 1–94 (1999)
8. Boyan, J.A., Moore, A.W.: Generalization in reinforcement learning: safely approximating the value function. In: Advances in Neural Information Processing Systems, pp. 369–376 (1995)
9. Bradtke, S.J., Barto, A.G.: Linear least-squares algorithms for temporal difference learning. Machine Learning **22**(1/2/3), 33–57 (1996)
10. Džeroski, S., Raedt, L.D., Drissens, K.: Relational reinforcement learning. Machine Learning **43**, 7–52 (2001)
11. Ferguson, K., Mahadevan, S.: Proto-transfer learning in Markov decision processes using spectral methods. In: ICML Workshop on Transfer Learning (2006)
12. Ferrante, E., Lazaric, A., Restelli, M.: Transfer of task representation in reinforcement learning using policy-based proto-value functions. In: International Joint Conference on Autonomous Agents and Multiagent Systems (2008)
13. Franzius, M., Sprekeler, H., Wiskott, L.: Slowness and sparseness leads to place, head-direction, and spatial-view cells. PLoS Computational Biology **3**(8), e166 (2007)
14. Hafner, R., Riedmiller, M.: Reinforcement learning in feedback control. Machine Learning **27**(1), 55–74 (2011)
15. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
16. Jonschkowski, R., Brock, O.: Learning task-specific state representations by maximizing slowness and predictability (2013). URL `http://www.robotics.tu-berlin.de/fileadmin/fg170/Publikationen_pdf/Jonschkowski-13-ERLARS-final.pdf`
17. Jonschkowski, R., Brock, O.: State representation learning in robotics: Using prior knowledge about physical interaction. In: In Proceedings of Robotics: Science and Systems (2014)
18. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence **101**, 99–134 (1998)
19. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: a survey. Journal of Artificial Intelligence Research **4**, 237–285 (1996)
20. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR (2014)
21. Kober, J., Bagnell, D., Peters, J.: Reinforcement learning in robotics: A survey. International Journal of Robotics Research **32**(11), 1238–1274 (2013)
22. Konidaris, G.D., Osentoski, S., Thomas, P.: Value function approximation in reinforcement learning using the Fourier basis. In: Proceedings of the Twenty-Fifth Conference on Artificial Intelligence (2011)
23. Lagoudakis, M.G., Parr, R.: Least-squares policy iteration. Journal of Machine Learning Research **4**, 1107–1149 (2003)
24. Lang, T., Toussaint, M.: Planning with noisy probabilistic relational rules. Journal of Artificial Intelligence Research **39**, 1–49 (2010)
25. Lange, S., Riedmiller, M., Voigtlaender, A.: Autonomous reinforcement learning on raw visual input data in a real world application. In: International Joint Conference on Neural Networks, Brisbane, Australia (2012)
26. Legenstein, R., Wilbert, N., Wiskott, L.: Reinforcement learning on slow features of high-dimensional input streams. PLoS Computational Biology **6**(8), e1000,894 (2010)
27. Levine, S., Abbeel, P.: Learning neural network policies with guided policy search under unknown dynamics. In: Advances in Neural Information Processing Systems (2014)
28. Lin, L.J.: Reinforcement learning for robots using neural networks. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, USA (1992)
29. Littman, M.L., Sutton, R.S., Singh, S.: Predictive representations of state. In: In Advances In Neural Information Processing Systems 14 (2001)
30. Luciw, M., Schmidhuber, J.: Low complexity proto-value function learning from sensory observations with incremental slow feature analysis. In: International Conference on Artificial Neural Networks and Machine Learning, vol. III, pp. 279–287. Springer-Verlag (2012)
31. Maass, W., Natschlaeger, T., Markram, H.: Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural Computation **14**(11), 2531–2560 (2002)
32. Maddison, C.J., Huang, A., Sutskever, I., Silver, D.: Move evaluation in go using deep convolutional neural networks. arXiv preprint arXiv:1412.6564 (2014)
33. Mahadevan, S., Liu, B.: Basis construction from power series expansions of value functions. In: Advances in Neutral Information Processing Systems, pp. 1540–1548 (2010)
34. Mahadevan, S., Maggioni, M.: Proto-value functions: a Laplacian framework for learning representations and control in Markov decision processes. Journal of Machine Learning Research **8**, 2169–2231 (2007)
35. Mattner, J., Lange, S., Riedmiller, M.: Learn to swing up and balance a real pole based on raw visual input data. In: Proceedings of the 19th International Conference on Neural Information Processing (5) (ICONIP 2012), pp. 126–133. Dohar, Qatar (2012)
36. Mnih, V., Hees, N., Graves, A., Kavukcuoglu, K.: Recurrent models of visual attention. In: Advances in Neural Information Processing Systems (2014)
37. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. In: NIPS Deep Learning Workshop (2013)
38. Mordatch, I., Todorov, E.: Combining the benefits of function approximation and trajectory optimization. In: Proceedings of Robotics: Science and Systems (RSS) (2014)
39. Parr, R., Li, L., Taylor, G., Painter-Wakefiled, C., Littman, M.L.: An analysis of linear models, linear value-function approximation, and feature selection for rein-

forcement learning. In: International Conference on Machine Learning (2008)
40. Parr, R., Painter-Wakefield, C., Li, L., Littman, M.: Analyzing feature generation for value-function approximation. In: International Conference on Machine Learning (2007)
41. Petrik, M.: An analysis of Laplacian methods for value function approximation in MDPs. In: International Joint Conference on Artificial Intelligence, pp. 2574–2579 (2007)
42. Rezende, D.J., Mohamed, S., Wierstra, D.: Stochastic backpropagation and approximate inference in deep generative models. In: International Conference on Machine Learning (2014)
43. Riedmiller, M.: Neural fitted q iteration - first experiences with a data efficient neural reinforcement learning method. In: 16th European Conference on Machine Learning, pp. 317–328. Springer (2005)
44. Riedmiller, M., Gabel, T., Hafner, R., Lange, S.: Reinforcement learning for robot soccer. Autonomous Robots **27**(1), 55–74 (2009)
45. Sallans, B., Hinton, G.E.: Reinforcement learning with factored states and actions. Journal of Machine Learning Research **5**, 1063–1088 (2004)
46. Schölkopf, B., Smola, A., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation **10**(5), 1299–1319 (1998)
47. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **22**(8), 888–905 (2000)
48. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic policy gradient algorithms. In: The 31st International Conference on Machine Learning (ICML 2014) (2014)
49. Snel, M., Whiteson, S.: Multi-task reinforcement learning: Shaping and feature selection. In: European Workshop on Reinforcement Learning, pp. 237–248 (2011)
50. Sprekeler, H.: On the relationship of slow feature analysis and Laplacian eigenmaps. Neural Computation **23**(12), 3287–3302 (2011)
51. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (1998)
52. Taylor, M.E., Stone, P.: Transfer learning for reinforcement learning domains: A survey. Journal of Machine Learning Research **10**, 1633–1685 (2009)
53. Tesauro, G.: Temporal difference learning and td-gammon. Commun. ACM **38**(3), 58–68 (1995)
54. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. (JMLR) **11**, 3371–3408 (2010)
55. Wingate, D., Singh, S.P.: On discovery and learning of models with predictive representations of state for agents with continuous actions and observations. In: International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1128–1135 (2007)
56. Wiskott, L.: Slow feature analysis: a theoretical analysis of optimal free responses. Neural Computation **15**(9), 2147–2177 (2003)
57. Wiskott, L., Sejnowski, T.: Slow feature analysis: unsupervised learning of invariances. Neural Computation **14**(4), 715–770 (2002)

Appendix B

Factored Representation Publications

B.1 Böhmer and Obermayer, ICRA Workshop (2013b)

The article *Towards structural generalization: factored approximate planning* (Böhmer and Obermayer, 2013) has been presented during the Workshop on Autonomous Learning at the *International Conferences on Robotics and Automation* 2013 in Karlsruhe, Germany. It is available online at `http://autonomous-learning.org/wp-content/uploads/13-ALW/paper_1.pdf`.

The article addresses the challenge to estimate value functions in large state spaces. I assume a set of *factored basis functions* and show how to exploit their factored structure to plan an optimal control policy, given linear models of transitions and reward. The framework is evaluated on a low-dimensional continuous control task. The developed *factored approximate planning* algorithm (FAPI) is not very efficient and I discuss a vastly improved version (that estimates state values) in Section 5.3.1. Both algorithms assume transition models that can be derived from DBN (see Section 5.3.3 for details). However, the notation in the article differs somewhat from these sections:

- The state and action dimensionality is referred to as p and q instead of d and b.
- The steady-state distribution in state-action space $\mathcal{Z}$ is denoted ξ instead of ζ.
- While the name of factored state-action basis functions remains $\{\psi_i\}_{i=1}^m$, the factor functions are referred to as φ_i^α, that is, $\psi_i(\boldsymbol{z}) := \prod_{\alpha=1}^{p+q} \varphi_i^\alpha(z_\alpha), \forall z \in \mathcal{Z}$.

The authors contributed in the following way:

- *Wendelin Böhmer* has written the entire article, derived the presented algorithm, and designed and performed all experiments.
- *Klaus Obermayer* has supervised both development and experimentation and gave comments to the article.

Towards Structural Generalization: Factored Approximate Planning

Wendelin Böhmer* and Klaus Obermayer*

***Abstract*— Autonomous agents do not always have access to the amount of samples machine learning methods require. Structural assumptions like *factored MDP* allow to generalize experiences beyond traditional metrics to entirely new situations. This paper introduces a novel framework to exploit such knowledge for approximated policy iteration. At the heart of the framework a novel *factored approximate planning* algorithm is derived. The algorithm requires no real observations and optimizes control for given linear reward and transition models. It is empirically compared with least squares policy iteration in a continuous navigation task. Computational leverage in constructing the linear models without observing the entire state space and in representation of the solution are discussed as well.**

I. INTRODUCTION

FULLY AUTONOMOUS *reinforcement learning* agents (RL, see e.g. [1]) will hardly rely on traditional *machine learning* methods (ML, see e.g. [2]) alone to generalize to new situations (i.e. *states*). ML is based on statistical inference and generalizes to unseen states according to a similarity metric. Traditional ML must thus collect experiences (*external* samples) "close" enough to every state to ensure generalization for the entire state space. However, many systems (e.g. autonomous robots [3]) can not afford excessive sampling. Instead they have to exploit the structure of the problem space at hand [4]. Usually this is achieved by *planning ahead* with constructed or learned *transition models* [5]. The behaviorally relevant *value function* is then *approximated* using traditional ML techniques based on samples drawn from these models (*internal* samples). Generalization still requires sampling the whole state space, but no external action anymore. By assuming additional knowledge about the problem at hand, however, one can decouple the approximation methods from sampling. *Factored function approximation* can be applied if the state of the system is represented by a set of many *conditionally independent variables* [6]. This independence structure yields a regularity that can be exploited to (i) solve a problem without sampling the whole state space and (ii) generalize to unseen states that are very dissimilar in traditional metrics. To approximate the value function, a majority in *factored MDP* literature focuses on linear combinations of basis functions with restricted domains [7]–[11]. However, this approach can not guarantee a proper value representation [12].

This paper aims for a novel approach towards a factored version of *approximated policy iteration* [13]. We derive a novel *factored approximate planning* algorithm for both finite and continuous state-action spaces (FAPI, Sec. III). The algorithm does not require external sampling and scales linear in the number of state-action variables without domain constraints on the basis functions. FAPI is empirically compared with *least squares policy iteration* (LSPI, [14]) in a navigation task with continuous state and action space (Sec. IV). The novel algorithm is only the first jigsaw piece in a framework to exploit structural knowledge. We discuss the remaining pieces in Sec. V and give an outlook towards structural generalization in huge factored MDP.

*Neural information processing group, Technische Universität Berlin, Germany. For questions or comments please contact Wendelin Böhmer <wendelin@ni.tu-berlin.de>. This work was funded by the *German science foundation* (DFG) within SPP 1527 *autonomous learning* and the *German federal ministry of education and research* (grant 01GQ0850).

II. PRELIMINARIES

This section introduces the necessary mathematical framework. As we allow finite *and* continuous state-action spaces, the formalism is more demanding than in standard text books about reinforcement learning (e.g. [1], [15]). Also, our *function theoretic* approach differs from previous attempts to continuous action spaces (e.g. [16], [17]).

A. Factored Markov Decision Processes

Let $\mathcal{B}(\mathcal{X})$ denote the collection of all Borel sets of set $\mathcal{X}$. A Markov decision process $M := (\mathcal{X}, \mathcal{A}, P, R)$ consists of a *state space* $\mathcal{X}$, an *action space* $\mathcal{A}$, a *transition kernel* P and a reward distribution R. $P : \mathcal{X} \times \mathcal{A} \times \mathcal{B}(\mathcal{X}) \to [0, 1]$ is the conditional distribution over all successive states, given a start state and action. We can reduce the reward distribution $R : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \times \mathcal{B}(\mathbb{R}) \to [0, 1]$ for our purposes to the *mean reward function* $r(x, a) = \int_{\mathbb{R}} \int_{\mathcal{X}} r\, R(dr|x, a, x')\, P(dx'|x, a)$.

In *factored MDP*, state space $\mathcal{X}$ and action space $\mathcal{A}$ are a collection of p state and q action variables, i.e. $\mathcal{X} := \mathcal{X}_1 \times \ldots \times \mathcal{X}_p$ and $\mathcal{A} := \mathcal{A}_1 \times \ldots \times \mathcal{A}_q$. The transition kernel P is assumed to be the product of p transition kernels P_α,

$$P(d\boldsymbol{x}'|\boldsymbol{x}, \boldsymbol{a}) = \prod_{\alpha=1}^{p} P_\alpha(dx'_\alpha|\boldsymbol{\rho}_\alpha)\,, \ \forall d\boldsymbol{x}' \in \mathcal{B}(\mathcal{X}_1) \times \ldots \times \mathcal{B}(\mathcal{X}_p).$$

A *dynamic Bayesian network* (DBN) defines the parents $\boldsymbol{\rho}_\alpha \subset \{x_1, \ldots, x_p, a_1, \ldots, a_q\}$ of state variable $\mathcal{X}_\alpha$ [6].

The goal of *reinforcement learning* is to find a policy, i.e. a conditional distribution $\pi : \mathcal{X} \times \mathcal{B}(\mathcal{A}) \to [0, 1]$ over actions given a state, such that the *value* $\int \pi(d\boldsymbol{a}|\boldsymbol{x})\, q^\pi(\boldsymbol{x}, \boldsymbol{a})$ of the *Q-value* function q^π is maximized in each state $\boldsymbol{x} \in \mathcal{X}$:

$$q^\pi(\boldsymbol{x}, \boldsymbol{a}) = r(\boldsymbol{x}, \boldsymbol{a}) + \gamma \iint P(d\boldsymbol{x}'|\boldsymbol{x}, \boldsymbol{a})\, \pi(d\boldsymbol{a}'|\boldsymbol{x}')\, q^\pi(\boldsymbol{x}', \boldsymbol{a}'),$$

where *discount factor* $\gamma \in [0, 1)$ reduces the influence of farther rewards exponentially [1]. Note that this formalism covers both finite and continuous spaces.

B. Factored Hilbert Spaces

Let $\xi : \mathcal{B}(\mathcal{X}) \to [0,1]$ denote the *steady state distribution* of the *Markov chain* following policy $\pi : \mathcal{X} \times \mathcal{B}(\mathcal{A}) \to [0,1]$. The mean of two *square-integrable functions* $f, g : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ over this Markov chain is also the inner product of the Hilbert space $L^2(\mathcal{X} \times \mathcal{A}, \xi\pi)$, i.e.

$$\langle f, g \rangle_{\xi\pi} = \iint \xi(d\boldsymbol{x})\, \pi(d\boldsymbol{a}|\boldsymbol{x})\, f(\boldsymbol{x},\boldsymbol{a})\, g(\boldsymbol{x},\boldsymbol{a})\,.$$

For convenience we combine states and actions in one $p+q$ dimensional space $\mathcal{Z} := \mathcal{X} \times \mathcal{A}$, i.e. $\mathcal{Z}_\alpha := \mathcal{X}_\alpha$ and $\mathcal{Z}_{p+\alpha} := \mathcal{A}_\alpha$. It is our goal to approximate arbitrary functions with *factored functions*. We know from *Fubini's theorem* that integrals over functions with mutually disjoint domains can be factorized. However, this also requires independent measures. We thus define $\{\vartheta_\alpha : \mathcal{B}(\mathcal{Z}_\alpha) \to [0,1]\}_{\alpha=1}^{p+q}, \vartheta := \prod_{\alpha=1}^{p+q} \vartheta_\alpha$, as a set of independent distributions over each state and action variable individually. This corresponds to the *factored state distribution* $\zeta := \prod_{\alpha=1}^{p} \vartheta_\alpha$ and the state independent *factored policy* $\tau := \prod_{\alpha=1}^{q} \vartheta_{p+\alpha}$. If functions $f, g \in L^2(\mathcal{Z}, \vartheta)$ can be factorized w.r.t. their domains, i.e. $f(\boldsymbol{z}) := \prod_{\alpha=1}^{p+q} f_\alpha(z_\alpha), \forall \boldsymbol{z} \in \mathcal{Z}$, then the multi-dimensional integral of the inner product can be computed very efficiently by the product of one-dimensional integrals:

$$\langle f, g \rangle_\vartheta = \prod_{\alpha=1}^{p+q} \langle f_\alpha, g_\alpha \rangle_{\vartheta_\alpha}\,. \tag{1}$$

In the following we will exploit factorizing inner products of *basis functions* $\{\psi_i\}_{i=1}^m \subset L^2(\mathcal{Z}, \vartheta)$. For this purpose we define *factor functions* $\varphi_i^\alpha : \mathcal{Z}_\alpha \to \mathbb{R}$ such that

$$\psi_i(\boldsymbol{z}) := \prod_{\alpha=1}^{p+q} \varphi_i^\alpha(z_\alpha)\,, \quad \varphi_i^\alpha(z_\alpha) := \sum_{j=1}^{m_\alpha} A_{ji}^\alpha\, \phi_j^\alpha(z_\alpha)\,,$$

where $\{\phi_j^\alpha : \mathcal{Z}_\alpha \to \mathbb{R}\}_{j=1}^{m_\alpha}$ is a partial basis of $L^2(\mathcal{Z}_\alpha, \vartheta_\alpha)$. Examples are continuous *Fourier* and *Gaussian basis functions*, or discrete *Kronecker delta functions*, i.e.

$$\begin{aligned}
\phi_j^\alpha(z_\alpha) &:= \cos\big((j-1)\,\pi\, z_\alpha\big)\,, && \text{(Fourier)}\\
\phi_j^\alpha(z_\alpha) &:= \exp\big(-\tfrac{1}{2\sigma^2}(z_\alpha - \mu_j^\alpha)^2\big)\,, && \text{(Gaussian)}\\
\phi_j^\alpha(z_\alpha) &:= \{1 \text{ if } j = z_\alpha, 0 \text{ if } j \neq z_\alpha\}\,, && \text{(Kronecker)}
\end{aligned}$$

which (in the limits of $j \to \infty$ and $\sigma \to 0$) span the space $L^2(\mathcal{Z}_\alpha, \vartheta_\alpha)$. $\{\psi_i\}_{i=1}^\infty$ can also span the entire state-action space $L^2(\mathcal{Z}, \vartheta)$ if all combinations of indices $\boldsymbol{j} := [j_1, \dots, j_{p+q}] \in \mathbb{N}^{p+q}$ are assigned one basis function $\psi_{\boldsymbol{j}} = \prod_{\alpha=1}^{p+q} \phi_{j_\alpha}^\alpha$. However, practically any other set of factorizing functions can be constructed by adjusting the matrices $\{\mathbf{A}^\alpha \in \mathbb{R}^{m_\alpha \times m}\}_{\alpha=1}^{p+q}$. See Sec. V-B for a discussion.

C. Approximated Policy Iteration

Reinforcement learning concepts laid out in Sec. II-A can be tackled in Hilbert space $L^2(\mathcal{Z}, \vartheta)$. For example, the effect of transition kernel P on any function $f \in L^2(\mathcal{X}, \zeta)$ can be expressed as *linear operator*[1] $\hat{\mathrm{P}} : L^2(\mathcal{X}, \zeta) \to L^2(\mathcal{Z}, \vartheta)$ and of policy π as linear operator $\hat{\Gamma}_\pi : L^2(\mathcal{Z}, \vartheta) \to L^2(\mathcal{X}, \zeta)$:

$$\begin{aligned}
\hat{\mathrm{P}}[f](\boldsymbol{x},\boldsymbol{a}) &:= \textstyle\int P(d\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{a})\, f(\boldsymbol{x}')\,, \quad \forall f \in L^2(\mathcal{X}, \zeta)\,,\\
\hat{\Gamma}_\pi[g](\boldsymbol{x}) &:= \textstyle\int \pi(d\boldsymbol{a}|\boldsymbol{x})\, g(\boldsymbol{x},\boldsymbol{a})\,, \quad \forall g \in L^2(\mathcal{Z}, \vartheta)\,.
\end{aligned}$$

This allows to formulate the q-value function $q^\pi \in L^2(\mathcal{Z}, \vartheta)$ as the fixed-point of *Bellman operator* $\hat{\mathrm{B}}^\pi$,

$$\hat{\mathrm{B}}^\pi[f](\boldsymbol{z}) := r(\boldsymbol{z}) + \gamma \hat{\mathrm{P}}\big[\hat{\Gamma}_\pi[f]\big](\boldsymbol{z})\,, \quad q^\pi(\boldsymbol{z}) \stackrel{!}{=} \hat{\mathrm{B}}^\pi[q^\pi](\boldsymbol{z})\,.$$

To *represent* the fixed-point, however, any $f \in L^2(\mathcal{Z}, \vartheta)$ can be *approximated* by a linear combination of m basis functions $\{\psi_i\}_{i=1}^m \subset L^2(\mathcal{Z}, \vartheta)$ using the *projection operator* $\hat{\Pi}_\vartheta^\psi[f](\boldsymbol{z}) = \sum_{i=1}^m w_i\, \psi_i(\boldsymbol{z})$, i.e. with $C_{ij} := \langle \psi_i, \psi_j \rangle_\vartheta$,

$$\hat{\Pi}_\vartheta^\psi[f](\boldsymbol{z}) := \sum_{i=1}^m \underbrace{\textstyle\sum_{j=1}^m \langle f, \psi_j \rangle_\vartheta\, (\mathbf{C}^{-1})_{ij}}_{w_i}\, \psi_i(\boldsymbol{z})\,, \quad \forall \boldsymbol{z} \in \mathcal{Z}\,.$$

Alternating applications of $\hat{\mathrm{B}}^\pi$ and $\hat{\Pi}_\vartheta^\psi$ lead to *least squares difference learning* (LSTD, [18]) fixed-point $f^\pi(\boldsymbol{z}) = \hat{\Pi}_\vartheta^\psi\big[\hat{\mathrm{B}}^\pi[f^\pi]\big] = \boldsymbol{b}^\top \mathbf{A}^\dagger \boldsymbol{\psi}(\boldsymbol{z})$, where $\dagger$ denotes the *pseudo-inverse* and both matrix $\mathbf{A} \in \mathbb{R}^{m \times m}$ and vector $\boldsymbol{b} \in \mathbb{R}^m$,

$$A_{ij} := \langle \psi_i, \psi_j \rangle_\vartheta - \gamma \langle \hat{\mathrm{P}}\big[\hat{\Gamma}_\pi[\psi_i]\big], \psi_j \rangle_\vartheta\,, \quad b_i := \langle \psi_i, r \rangle_\vartheta\,, \tag{2}$$

can be bias-free estimated by samples $(\boldsymbol{x},\boldsymbol{a}) \sim \vartheta$, rewards r, and successive samples $\boldsymbol{x}' \sim P(\cdot|\boldsymbol{x},\boldsymbol{a})$ and $\boldsymbol{a}' \sim \pi(\cdot|\boldsymbol{x}')$. A set of n samples $\{\boldsymbol{x}_t, \boldsymbol{a}_t, r_t, \boldsymbol{x}_t'\}_{t=1}^n$ can thus be used to estimate the Q-value of *any* policy π by choosing $\{\boldsymbol{a}_t'\}_{t=1}^n$.

To find policies with maximal value, *least squares policy iteration* (LSPI, [14]) iteratively performs *Q-value estimation* with LSTD and *policy improvement* by choosing $\{\boldsymbol{a}_t'\}_{t=1}^n$ that maximize the current Q-value estimate of $\{\boldsymbol{x}_t'\}_{t=1}^n$. The first iteration is usually initialized with random actions.

III. FACTORED APPROXIMATE PLANNING

Approximated policy iteration (API, Sec. II-C or e.g. [13]) solves multi-dimensional integrals in (2) by sampling from $\mathcal{Z}$. Generalization to arbitrary situations requires thus samples "close" enough to each state-action pair $\boldsymbol{z} \in \mathcal{Z}$. Sampling each variable at s locations yields $n = s^{p+q}$ samples covering $\mathcal{Z}$. Factored Hilbert spaces using (1) can achieve the same accuracy with only $n = s\,(p+q)$ samples. This section derives a novel *factored approximate planning* algorithm, able to perform policy iteration given linear transition and reward models and factored basis functions.

A. Factored Q-value Estimation Step

For the purpose of LSTD (2), the vector $\boldsymbol{b} \in \mathbb{R}^m$ is a sufficient model of the reward function. Matrix $\mathbf{A} = \mathbf{C} - \gamma \mathbf{D}^\pi$ consists two parts. $\mathbf{C} = \langle \boldsymbol{\psi}, \boldsymbol{\psi}^\top \rangle_\vartheta$ can be computed factored as in (1). Matrix $\mathbf{D}^\pi := \langle \hat{\mathrm{P}}[\hat{\Gamma}_\pi[\boldsymbol{\psi}]], \boldsymbol{\psi}^\top \rangle_\vartheta$, on the other hand, depends on transition operator $\hat{\mathrm{P}}$ and policy π. To overcome the latter dependency, we approximate the policy *explicitly* as its *Radon-Nikodym derivative* $\frac{d\pi}{d\tau} \in L^2(\mathcal{Z}, \vartheta)$ with factored policy $\tau := \prod_{\alpha>p}^{p+q} \vartheta_\alpha$:

$$\begin{aligned}
\hat{\Gamma}_\pi[\psi_i](\boldsymbol{x}) &= \textstyle\int \tau(d\boldsymbol{a})\, \frac{d\pi}{d\tau}(\boldsymbol{x},\boldsymbol{a})\, \psi_i(\boldsymbol{x},\boldsymbol{a})\\
&\approx \textstyle\sum_{k=1}^{m'} \varpi_k\, \hat{\Gamma}_\tau[\psi_k' \cdot \psi_i](\boldsymbol{x})\,, \quad \forall \boldsymbol{x} \in \mathcal{X}\,.
\end{aligned}$$

[1] *Linear operators* are extensions of matrices to Hilbert spaces. Let $A : \mathcal{Z} \times \mathcal{B}(\mathcal{X}) \to [0,1]$ be a *bounded conditional measure*, then $\hat{\mathrm{A}}[f](\boldsymbol{z}) := \int A(d\boldsymbol{x}'|\boldsymbol{z})\, f(\boldsymbol{x}')$ is the corresponding linear operator $\hat{\mathrm{A}} : L^2(\mathcal{X}, \zeta) \to L^2(\mathcal{Z}, \vartheta)$. We also use the *pointwise product* $[f \cdot g](\boldsymbol{z}) := f(\boldsymbol{z})\, g(\boldsymbol{z})$ later.

Here we linearly approximate $\frac{d\pi}{d\tau}(\boldsymbol{z}) \overset{!}{\approx} \sum_{k=1}^{m'} \varpi_k \, \psi'_k(\boldsymbol{z})$ with a set of basis functions $\{\psi'_k\}_{k=1}^{m'} \subset L^2(\mathcal{Z}, \vartheta)$. Note that these can (but no not have to) be the different from the basis $\{\psi_i\}_{i=1}^{m}$ used to represent the Q-value function. Now $\mathbf{D}^\pi$ can be approximated by

$$\langle \hat{\mathrm{P}}\big[\hat{\Gamma}_\pi[\psi_i]\big], \psi_j \rangle_\vartheta \approx \sum_{k=1}^{m'} \varpi_k \, \langle \psi'^a_k, \psi^a_i \rangle_\tau \langle \hat{\mathrm{P}}\big[\psi'^x_k \cdot \psi^x_i\big], \psi_j \rangle_\vartheta \,,$$

where $\psi^x_i := \prod_{\alpha=1}^{p} \varphi^\alpha_i \in L^2(\mathcal{X}, \zeta)$ and $\psi^a_i := \prod_{\alpha>p}^{p+q} \varphi^\alpha_i \in L^2(\mathcal{A}, \tau)$. The transition operator $\hat{\mathrm{P}}$ calculates the *conditional correlation* over successive states of functions ψ'^x_k and ψ^x_i. Estimating a full correlation matrix conditioned on each state-action pair will take an exorbitant amount of samples. To reduce sampling we propose an approximation that assumes an unconditional *covariance matrix*[2] $\boldsymbol{\Sigma}$ of these functions, but conditional means $\hat{\mathrm{P}}[\psi^x_i]$ and $\hat{\mathrm{P}}[\psi'^x_k]$, i.e.

$$\hat{\mathrm{P}}[\psi'^x_k \cdot \psi^x_i](\boldsymbol{z}) \approx \Sigma_{ki} + \hat{\mathrm{P}}[\psi'^x_k](\boldsymbol{z}) \; \hat{\mathrm{P}}[\psi^x_i](\boldsymbol{z}) \,.$$

Means can be estimated with reasonable amounts of data:

$$\hat{\mathrm{P}}[\psi^x_i](\boldsymbol{z}) \approx \sum_{j=1}^{m} P_{ji} \, \psi_j(\boldsymbol{z}) \,, \quad \hat{\mathrm{P}}[\psi'^x_i](\boldsymbol{z}) \approx \sum_{j=1}^{m} P'_{ji} \, \psi_j(\boldsymbol{z}) \,.$$

Under these assumptions one can approximate matrix $\mathbf{D}^\pi$ without solving high dimensional integrals numerically, i.e.

$$\begin{aligned} D^\pi_{ij} \quad \approx \quad & \sum_{k=1}^{m'} \varpi_k \, \langle \psi'^a_k, \psi^a_i \rangle_\tau \\ & \cdot \big(\langle \psi_j, 1 \rangle_\vartheta \boldsymbol{\Sigma} + \mathbf{P}'^\top \langle [\boldsymbol{\psi} \cdot \psi_j], \boldsymbol{\psi}^\top \rangle_\vartheta \mathbf{P} \big)_{ki} . \end{aligned} \tag{3}$$

Note that the tensor $\langle [\psi_k \cdot \psi_j], \psi_i \rangle_\vartheta = \prod_{\alpha=1}^{p+q} \langle [\varphi^\alpha_k \cdot \varphi^\alpha_j], \varphi^\alpha_i \rangle_{\vartheta_\alpha}$ is fully factorizable following (1), i.e. can be computed by multiplying $p+q$ one-dimensional integrals. Additionally,

$$\langle [\varphi^\alpha_k \cdot \varphi^\alpha_j], \varphi^\alpha_i \rangle_{\vartheta_\alpha} = \sum_{s=1}^{m_\alpha} \sum_{t=1}^{m_\alpha} \sum_{u=1}^{m_\alpha} A^\alpha_{sk} \, A^\alpha_{tj} \, A^\alpha_{ui} \, \langle [\phi^\alpha_s \cdot \phi^\alpha_t], \phi^\alpha_u \rangle_{\vartheta_\alpha} \,,$$

which requires only $\sum_{\alpha=1}^{p+q} m_\alpha^3$ highly redundant integrals. In some cases those integrals can even be solved analytically. The right side tensor is also much smaller and can usually be precomputed and stored.

The algorithm's Q-value estimation step requires a reward model $\boldsymbol{b} \in \mathbb{R}^m$, policy parameters $\boldsymbol{\varpi} \in \mathbb{R}^{m'}$, covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{m' \times m}$ (not for deterministic systems) and transition models $\mathbf{P} \in \mathbb{R}^{m \times m}$ and $\mathbf{P}' \in \mathbb{R}^{m \times m'}$.

B. Factored Policy Improvement Step

Improving the policy usually requires significant sampling as well. LSPI applies a *greedy policy* on the training set, i.e. π_{i+1} is a conditional *point-distribution* selecting for each state $\boldsymbol{x} \in \mathcal{X}$ exclusively one $\boldsymbol{a} \in \mathcal{A}$ that maximizes the estimated Q-value $q^{\pi_i}(\boldsymbol{x}, \boldsymbol{a}) \approx \sum_{j=1}^{m} w_j \, \psi_j(\boldsymbol{x}, \boldsymbol{a})$. Explicit approximation of the corresponding Radon-Nikodym derivative $\frac{d\pi_{i+1}}{d\tau}(\boldsymbol{z}) \approx \sum_{k=1}^{m'} \varpi_k \, \psi'_k(\boldsymbol{z})$ is always possible[3], but can require an unacceptable amount of basis functions. In any case, the maximization would have to be performed for each state $\boldsymbol{x} \in \mathcal{X}$ individually, which again requires exponentially many samples. Instead of this state-wise maximization, we settle here for an improvement of the *mean value*, i.e.

$$\sup_{\pi_{i+1}} \langle q^{\pi_i}, 1 \rangle_{\zeta \pi_{i+1}} \quad \approx \quad \sup_{\boldsymbol{\varpi}} \boldsymbol{\varpi}^\top \langle \boldsymbol{\psi}', \boldsymbol{\psi}^\top \rangle_\vartheta \, \boldsymbol{w} \,.$$

However, the solution must also adhere to the constraints of a conditional probability distribution, i.e.

$$\pi_{i+1}(d\boldsymbol{a}|\boldsymbol{x}) \; \geq \; 0 \quad \text{and} \quad \int \pi_{i+1}(d\boldsymbol{a}|\boldsymbol{x}) \; = \; 1 \,.$$

The equivalent first constraint $\boldsymbol{\varpi}^\top \boldsymbol{\psi}'(\boldsymbol{x}, \boldsymbol{a}) \geq 0$ can be fulfilled by choosing non-negative ψ'_i and constraining the policy-parameters $\boldsymbol{\varpi} \succeq 0$. The second can usually not be achieved perfectly, i.e. $\boldsymbol{\varpi}^\top \hat{\Gamma}_\tau[\boldsymbol{\psi}'](\boldsymbol{x}) \approx 1$. Instead one can constraint the solution to minimize the induced error

$$\inf_{\boldsymbol{\varpi}} \tfrac{1}{2} \Big\| \boldsymbol{\varpi}^\top \hat{\Gamma}_\tau[\boldsymbol{\psi}'] - 1 \Big\|^2_\zeta \; \Rightarrow \; \langle \hat{\Gamma}_\tau[\boldsymbol{\psi}'], \hat{\Gamma}_\tau[\boldsymbol{\psi}'^\top] \rangle_\zeta \boldsymbol{\varpi} \overset{!}{=} \langle \boldsymbol{\psi}', 1 \rangle_\vartheta .$$

The right hand side of the above equation retains sufficient degrees of freedom, yielding the optimization problem

---

Algorithm 1 Factored approximated planning (FAPI)

Input: $\mathbf{P}$, $\mathbf{P}'$, $\boldsymbol{\Sigma}$, $\boldsymbol{b}$, γ

// Prepare factored integrals as in (1)

$\boldsymbol{c} \leftarrow \prod_{\alpha=1}^{p+q} \langle \boldsymbol{\varphi}^\alpha, 1 \rangle_{\vartheta_\alpha} \quad \in \mathbb{R}^m$

$\mathbf{C} \leftarrow \prod_{\alpha=1}^{p+q} \langle \boldsymbol{\varphi}^\alpha, \boldsymbol{\varphi}^{\alpha\top} \rangle_{\vartheta_\alpha} \quad \in \mathbb{R}^{m \times m}$

$\mathbf{C}' \leftarrow \prod_{\alpha=1}^{p+q} \langle \boldsymbol{\varphi}'^\alpha, \boldsymbol{\varphi}^{\alpha\top} \rangle_{\vartheta_\alpha} \quad \in \mathbb{R}^{m' \times m}$

$\mathbf{C}'' \leftarrow \prod_{\alpha>p}^{p+q} \langle \boldsymbol{\varphi}'^\alpha, \boldsymbol{\varphi}^{\alpha\top} \rangle_{\vartheta_\alpha} \quad \in \mathbb{R}^{m' \times m}$

$\boldsymbol{c}' \leftarrow \prod_{\alpha>p}^{p+q} \langle \boldsymbol{\varphi}'^\alpha, 1 \rangle_{\vartheta_\alpha} \quad \in \mathbb{R}^{m'}$

$\mathbf{C}''' \leftarrow (\boldsymbol{c}' \, \boldsymbol{c}'^\top) \cdot \prod_{\alpha=1}^{p} \langle \boldsymbol{\varphi}'^\alpha, \boldsymbol{\varphi}'^{\alpha\top} \rangle_{\vartheta_\alpha} \quad \in \mathbb{R}^{m' \times m'}$

$\boldsymbol{c}' \leftarrow \boldsymbol{c}' \cdot \prod_{\alpha=1}^{p} \langle \boldsymbol{\varphi}'^\alpha, 1 \rangle_{\vartheta_\alpha}$

$\mathbf{T} \leftarrow \prod_{\alpha=1}^{p+q} \langle [\boldsymbol{\varphi}^\alpha \cdot \boldsymbol{\varphi}^\alpha], \boldsymbol{\varphi}^{\alpha\top} \rangle_{\vartheta_\alpha} \quad \in \mathbb{R}^{m \times m \times m}$

// Initialize value function with reward function

$\boldsymbol{w} \leftarrow \mathbf{C}^{-1} \boldsymbol{b}; \quad \boldsymbol{w}' \leftarrow \infty \quad \in \mathbb{R}^m$

// Start approximate policy iteration loop

while $\|\boldsymbol{w} - \boldsymbol{w}'\| > \epsilon$ **do**

$\boldsymbol{w}' \leftarrow \boldsymbol{w}$

// Factored policy improvement

$\boldsymbol{\varpi} \leftarrow \sup_{\boldsymbol{\varpi}} \boldsymbol{\varpi}^\top \mathbf{C}' \boldsymbol{w}, \quad \text{s.t. } \boldsymbol{\varpi} \succeq 0, \quad \mathbf{C}''' \boldsymbol{\varpi} = \boldsymbol{c}'$ (4)

// Construct linear transition matrix $\mathbf{D}^\pi \in \mathbb{R}^{m \times m}$

for $j \in \{1, \ldots, m\}$ **do**

$\mathbf{D}_{:j} \leftarrow \boldsymbol{\varpi}^\top \Big(\big(c_j \, \boldsymbol{\Sigma} + \mathbf{P}'^\top \mathbf{T}_{::j} \, \mathbf{P} \big) \cdot \mathbf{C}'' \Big)$ (3)

end for

// Factored Q-value estimation

$\boldsymbol{w} \leftarrow \big(\mathbf{C} - \gamma (\mathbf{D}^\pi)^\top \big)^\dagger \boldsymbol{b}$ (2)

end while

Output: $\boldsymbol{\varpi}$, $\boldsymbol{w}$

---

[2] Note that covariance matrix $\boldsymbol{\Sigma}$ is always zero in deterministic systems.

[3] E.g. the *Dirac delta function* on $\mathcal{A}$ is not always in $L^2(\mathcal{A}, \tau)$. It is the limit of a series of Gaussian functions with shrinking width in $L^2(\mathcal{A}, \tau)$, though. Any $\frac{d\pi}{d\tau}$ can therefore be arbitrarily well approximated in $L^2(\mathcal{Z}, \vartheta)$.

$$\begin{aligned} \sup_{\boldsymbol{\varpi}} \quad & \boldsymbol{\varpi}^\top \langle \boldsymbol{\psi}', \boldsymbol{\psi}^\top \rangle_\vartheta \, \boldsymbol{w} \\ \text{s.t.} \quad & \boldsymbol{\varpi} \succeq 0 \qquad \text{with} \qquad \boldsymbol{\psi}' \succeq 0 \,, \\ & \boldsymbol{\varpi}^\top \langle \hat{\Gamma}_\tau[\boldsymbol{\psi}'], \hat{\Gamma}_\tau[\boldsymbol{\psi}'^\top] \rangle_\zeta \;=\; \langle \mathbf{1}, \boldsymbol{\psi}'^\top \rangle_\vartheta \,. \end{aligned} \tag{4}$$

This *linear program* with linear equality and inequality constraints can be solved by a variety of algorithms [19]. We employed the `linprog` function of MATLAB. Note that all involved inner products adhere to (1) and therefore do not require sampling of all states in $\mathcal{X}$ or $\mathcal{Z}$. The non-negativity constraint restricts the choice of ψ'_i, though.

C. Factored Approximated Planning Algorithm

There are multiple ways to compute the tensor in (3). Algorithm 1 (FAPI) shows an illustrative yet inefficient version with computational complexity $\mathcal{O}(m^3\, s\,(p\,+\,q))$ and memory complexity $\mathcal{O}(m^3)$. This can be reduced to $\mathcal{O}(s\,\sum_{\alpha=1}^{p+q} m_\alpha^3)$ and $\mathcal{O}(\sum_{\alpha=1}^{p+q} m_\alpha^3)$, respectively. Note that the policy iteration loop of Algorithm 1 has computational complexity $\mathcal{O}(m^3 m')$ and memory complexity $\mathcal{O}(m^2)$ per iteration. Computation time thus exceeds LSPI by a factor of m', but avoids *any* sampling. Each LSPI iteration, on the other hand, induces sampling costs of magnitude $\mathcal{O}(m^2\, s^{p+q})$. For large $p+q$ this can be a decisive advantage. Moreover, all samples for LSPI are *external*[4], i.e. follow the true transition and reward model. FAPI outsources this into its linear models. Sec. V-A discusses how to exploit factored and/or rule-based MDP structure to build models with a minimum of external sampling.

IV. EMPIRICAL EVALUATION

This section empirically compares FAPI with LSPI on a standard low-dimensional continuous control problem to test the reliability of the adopted approximations.

The task is to navigate into a circular goal area with a diameter of 4 movements (first row of Fig. 1). Reaching this area is rewarded with +1. The state space is a subset of the *two dimensional plane*, the action space the *direction* in which the agent moves a fixed distance. If the movement is blocked (crosses a line in Fig. 1), the agent stops at half the distance to the boundary and is punished with -1.

Two environments were tested (first row of Fig. 1):

1) U-shaped room (left) is 10 movements wide and long.
2) S-shaped room (right) is 12 steps wide and 16 long.

A. Choice of Basis Functions

As indicated in Sec. III-B, the basis functions for the Q-value $\{\psi_i\}_{i=1}^m$ and the policy $\{\psi'_i\}_{i=1}^{m'}$ can be different as the latter must be strictly non-negative. We chose the same Fourier bases for the Q-value function of FAPI and LSPI. The action variable was encoded with a *circular boundary condition*, i.e. even frequencies and accompanying phase

[4]One should be able to use a linear model of $\hat{\mathrm{P}}[\psi_i^x]$ to sample LSPI *internally* as well. For each state action pair $\boldsymbol{z}$ the *future decision* $a^*(\boldsymbol{z}) = \arg\sup_{\boldsymbol{a}} \boldsymbol{w}^\top [\hat{\mathrm{P}}[\boldsymbol{\psi}^x](\boldsymbol{z}) \cdot \boldsymbol{\psi}^a(\boldsymbol{a})]$ can be used to express $\hat{\mathrm{P}}\big[\hat{\Gamma}_\pi[\psi_i]\big](\boldsymbol{z}) = \hat{\mathrm{P}}[\psi_i^x](\boldsymbol{z}) \cdot \psi_i^a\big(a^*(\boldsymbol{z})\big)$, which is required for the LSTD estimate (2).

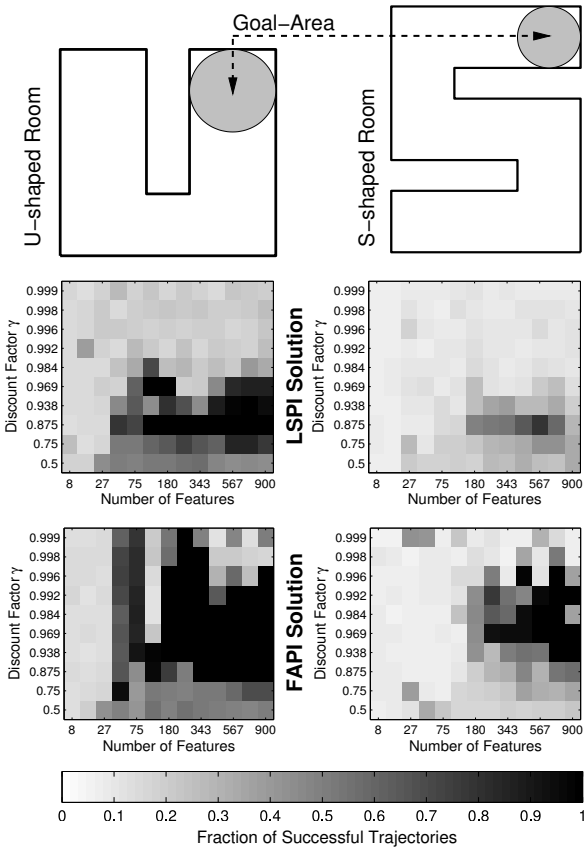


Fig. 1. Navigation performance in a U-shaped room (left column, first row) and a S-shaped room (right column, first row) after 10 iterations of LSPI (mean of 10 initializations, second row) and the novel factorized planning algorithm (third row). Grayscale intensity represents the fraction of successful test trajectories from random start-positions for different Fourier/Gaussian feature sets and discount factors γ.

shifted sine-wave features. The policy in FAPI is represented by (non-negative) Gaussian functions with equally spaced means μ_i^α two standard deviations σ apart from each other. The action variable is also encoded circular. The tested number of features in the state variables x/y and the action variable a are given in Table I, as well as the resulting number of basis functions m/m'. Both feature sets $\{\psi_i\}_{i=1}^m$ and $\{\psi'_i\}_{i=1}^{m'}$ were chosen to have the same size, i.e. $m = m'$.

TABLE I
FEATURES PER VARIABLE AND RESULTING SET SIZES

x/y	2	3	3	4	5	5	6	7	7	8	9	9	10
a	2	2	3	3	3	5	5	5	7	7	7	9	9
m/m'	8	18	27	48	75	125	180	245	343	448	567	729	900

Factored integrals in FAPI were numerically estimated with 1000 equally spaced samples in each variable, yielding 3000 samples which correspond to 10^9 samples in $\mathcal{Z}$. Integrals in LSPI were estimated based on external sampling.

B. Estimation of Reward and Transition Models

The FAPI algorithm requires linear models of reward $\boldsymbol{b} \in \mathbb{R}^m$ and transitions in both feature spaces $\mathbf{P} \in \mathbb{R}^{m \times m}$ and

$\mathbf{P}' \in \mathbb{R}^{m \times m'}$. Without going into details how to construct these models *efficiently* (see Sec. V-A), we estimated them with the same samples used for LSPI. These samples were transitions from equidistant state-action pairs: 50 samples for each state variable with 10 actions (directions) each. Note that some of the 25,000 samples are "outside" the allowed state space. Here all actions resulted in no movements and the agent received the punishment for blocked movements. The reward model $\boldsymbol{b}$ was estimated as defined in (2). Transition models can be estimated $\mathbf{P}^\top \boldsymbol{\psi}(\boldsymbol{z}) = \hat{\Pi}^{\boldsymbol{\psi}}_{\vartheta}\big[\hat{\mathrm{P}}[\boldsymbol{\psi}^{\boldsymbol{x}}]\big](\boldsymbol{z})$, i.e.

$$\mathbf{P} := \mathbf{C}^{-1}\langle \boldsymbol{\psi}, \hat{\mathrm{P}}[\boldsymbol{\psi}^{\boldsymbol{x}\top}]\rangle_\vartheta\,, \qquad \mathbf{P}' := \mathbf{C}^{-1}\langle \boldsymbol{\psi}, \hat{\mathrm{P}}[\boldsymbol{\psi}'^{\boldsymbol{x}\top}]\rangle_\vartheta\,.$$

Due to the deterministic transitions, $\boldsymbol{\Sigma} = \mathbf{0}$.

C. Results

As measure of success we estimated the *navigation performance*, i.e. the fraction of 1000 trajectories (with random start positions) hitting the goal area within 50 steps. Choosing a deterministic action for state $\boldsymbol{x}$ w.r.t. tested policy π_i necessitates to solve the optimization problems $\sup_{\boldsymbol{a}}\{\boldsymbol{w}^\top \boldsymbol{\psi}(\boldsymbol{x}, \boldsymbol{a})\}$ for LSPI or $\sup_{\boldsymbol{a}}\{\boldsymbol{\varpi}^\top \boldsymbol{\psi}'(\boldsymbol{x}, \boldsymbol{a})\}$ for FAPI. In high dimensional action spaces this would require an elaborate optimization algorithm, but for the sake of simplicity we evaluated these functions on 24 equally spaced actions[5] and chose the maximum instead.

Fig. 1 shows grayscale plots which represent the navigation performance for different combinations of feature set sizes m/m' (see Table I) and discount factors $\gamma \in \{1-2^{-i}\}_{i=1}^{10}$. One can observe that the S-shaped room (right column) is considerably harder to navigate than the U-shaped room (left column). Note that both algorithms work well (if they work at all) in roughly the same range of feature sets. However, FAPI clearly outperforms LSPI throughout a range of larger discount factors γ in both environments.

This is puzzling as FAPI is supposed to be an *approximation* of LSPI, except in terms of factored inner products and policy improvement. To shed some light on this, Fig. 2 shows the *development* of the policy over 4 iterations of both algorithms in the U-shaped room. One can observe that the first LSPI policy (after LSTD with random actions) navigates surprisingly well for large γ. In this regime a random policy propagates enough reward to induce an almost optimal policy everywhere. Performance for lower γ increases steadily as improving policies carry the reward further. However, those initial achievements for high γ *destabilize* and drop to chance level. Apparently, this destabilization is little affected by the feature space. It may be a sign of *over-fitting* in the policy improvement step of LSPI, though. FAPI, on the other hand, shows gradual (although not very homogeneous) improvement for all but the largest γ.

V. DISCUSSION

The novel Algorithm 1 (FAPI) is the first jigsaw piece of a framework to exploit structural knowledge for API. The complete framework is beyond the scope of this paper, but this section outlines the remaining pieces.

[5]Note that the evaluated function and thus the policy is still continuous.

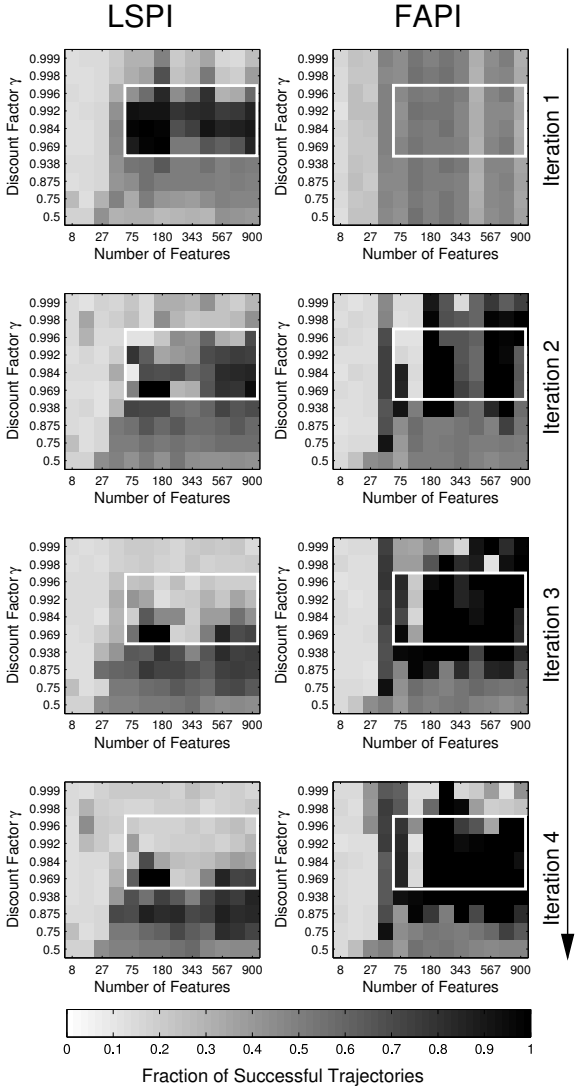


Fig. 2. Development of navigation performance in the U-shaped room for LSPI (mean of 10 initializations, left column) and FAPI (right column). After 4 iterations the solution is more or less stable. The white square marks roughly the range of initial LSPI success. Results of the S-shaped room show the same qualitative effect on a larger time scale. See Fig. 1 for a description of the grayscale plots and performance results at iteration 10.

A. Sample-efficient Model Estimation

In difference to most factored MDP algorithms (e.g. [7], [8], et seq.), FAPI does not make direct use of the *dynamic Bayesian network* (DBN). One can exploit this structural assumption in the estimation of the *linear transition models*, though. Straight forward estimation as in Sec. IV-B requires excessive external sampling. A DBN, on the other hand, allows efficient linear approximation of *factored transition models* $\hat{\mathrm{P}}[\psi_i^{\boldsymbol{x}}](\boldsymbol{z}) = \prod_{\alpha=1}^{p} \sum_{j=1}^{m_\alpha} A^\alpha_{ji}\, \hat{\mathrm{P}}_\alpha[\phi_j^\alpha](\boldsymbol{\rho}_\alpha)$ (see p. 1):

$$\hat{\mathrm{P}}_\alpha[\phi_j^\alpha](\boldsymbol{z}) \quad \approx \quad \sum_{k=1}^{m} P^\alpha_{kj}\, \psi_k(\boldsymbol{z})\,, \quad \text{with}$$

$$\mathbf{P}^{\alpha} := \mathbf{C}^{-1}\Big[\big(\prod_{\beta \notin \rho_\alpha} \langle \boldsymbol{\varphi}^{\beta}, \mathbf{1}^{\top} \rangle_{\vartheta_\beta}\big) \cdot \Big(\big\langle \prod_{\beta \in \rho_\alpha} \boldsymbol{\varphi}^{\beta}, \hat{\mathrm{P}}_{\alpha}[\boldsymbol{\phi}^{\alpha\top}] \big\rangle_{\prod_{\beta \in \rho_\alpha} \vartheta_\beta}\Big)\Big].$$

Note that the left hand side inner products are one-dimensional and the right hand side is $|\boldsymbol{\rho}_\alpha|$-dimensional. *External sampling* for factored transition models scales thus exponentially with the *maximum in-degree* $\max_\alpha\{|\boldsymbol{\rho}_\alpha|\}$ [20] instead of the number of variables $p + q$, which is feasible in many cases. However, being able to predict *each variable* does not directly yield a linear transition model $\mathbf{P}$, i.e.

$$\mathbf{P} := \mathbf{C}^{-1}\langle \boldsymbol{\psi}, \hat{\mathrm{P}}[\boldsymbol{\psi}^{x\top}] \rangle_{\vartheta} \approx \mathbf{C}^{-1}\langle \boldsymbol{\psi}, \prod_{\alpha=1}^{p} \boldsymbol{\psi}^{\top}\mathbf{P}^{\alpha}\mathbf{A}^{\alpha} \rangle_{\vartheta}\,, \quad (5)$$

which is not factorizable due to the product of sums and thus has sample complexity of $\mathcal{O}(s^{p+q})$. On the other hand, given $\{\mathbf{P}^{\alpha}\}_{\alpha=1}^{p}$ this inner product can be estimated with *internal sampling* alone, which is acceptable in many domains.

If it is not acceptable, sampling complexity can be reduced by exploiting *rule-based* structural assumptions [4]. For example, assume two variables describe the position of two physical objects. Moving the objects will only affect each variable individually, *except* when their target locations overlap. Instead of sampling the full state-action space, one could *initialize* (5) assuming independence and then *update* this estimate with samples fitting the above rule.

Both approaches to exploit MDP structure to reduce external sampling are promising and demand further research.

B. Constructing Basis Functions

General representations of Q-values in factored MDP usually grow exponential in the number of variables. For example, a Fourier basis with $m_\alpha := s$ basis functions for each variable $\mathcal{Z}_\alpha$ requires s^{p+q} state-action basis functions $\psi_i \in L^2(\mathcal{Z}, \vartheta)$. Trading sample for representation complexity is certainly not of computational advantage. Instead one can use observations and structural assumptions to construct a more compact basis $\{\psi_i\}_{i=1}^{m} \subset L^2(\mathcal{Z}, \vartheta)$ [21]–[24]. Starting with a set of *anticipated tasks*, e.g. all MDP with the observed transition kernel, one aims for bases that minimize the approximation error $\|q^\pi - \hat{\Pi}_{\vartheta}^{\psi}[q^\pi]\|_\vartheta$ for all policies π. Following the analysis in [24], one should extract eigenfunctions of the (symmetrized) transition operator, i.e. find weights $\{\{A_{ji}^{\alpha}\}_{j=1}^{m_\alpha}\}_{\alpha=1}^{p+q}$ of function ψ_i, such that

$$\sup_{\psi_i} \big\{ \langle \psi_i, \hat{\mathrm{P}}\big[\hat{\Pi}_\tau[\psi_i]\big] \rangle_\vartheta \big\} \;\equiv\; \sup_{\psi_i} \Big\{ \langle \psi_i, \hat{\mathrm{P}}[\psi_i^x] \rangle_\vartheta \cdot \langle \psi_i^a, 1 \rangle_\tau \Big\}.$$

The similarity to (5) suggests a connection between an efficient representation and the estimation of linear transition models. Future works must exploit this to learn both representation and prediction at the same time.

C. Conclusion

The presented FAPI algorithm is the first piece of a larger framework exploiting structure for API. The algorithm has proven itself under standard conditions by performing at least as well as state-of-the-art method LSPI, but allowing more flexibility in choosing γ. Future works on structured transition models and factored representations will allow FAPI to run efficiently in huge state-action spaces and structurally generalize to unexperienced but predictable situations.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006.

[3] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to aerobatic helicopter flight," in *Advances in Neural Information Processing Systems*, 2007, pp. 1–8.

[4] T. G. Dietterich, P. Domingos, L. Getoot, S. Muggleton, and P. Tadepalli, "Structured machine learning: the next ten years," *Machine Learning*, vol. 73, pp. 3–23, 2008.

[5] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cognitive Processing*, vol. 12, pp. 319–340, 2011.

[6] C. Boutilier, T. Dean, and S. Hanks, "Decision-theoretic planning: Structural assumptions and computational leverage," *Journal of Artificial Intelligence Research*, vol. 11, pp. 1–94, 1999.

[7] D. Koller and R. Parr, "Computing factored value functions for policies in structured MDPs," in *International Joint Conference on Artificial Intelligence*, 1999, pp. 1332–1339.

[8] C. Guestrin, D. Koller, and R. Parr, "Max-norm projections for factored MDPs," in *International Joint Conference on Artificial Intelligence*, 2001, pp. 673–682.

[9] M. Hauskrecht and B. Kveton, "Linear program approximations for factored continuous-state Markov decision processes," in *Advances in Neural Information Processing Systems*, 2003, pp. 895–902.

[10] C. Guestrin, M. Hauskrecht, and B. Kveton, "Solving factored MDPs with continuous and discrete variables," in *Uncertainty in Artificial Intelligence*, 2004, pp. 235–242.

[11] D. Chakraborty and P. Stone, "Structure learning in ergodic factored mdps without knowledge of the transition function's in-degree," in *International Conference on Machine Learning*, 2011, pp. 737–744.

[12] E. Allender, S. Arora, M. Kearns, C. Moore, and A. Russell, "A note on the representational incompatibility of function approximation and factored dynamics," in *Advances in Neural Information Processing Systems*, 2002, pp. 431–437.

[13] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Athena Scientific, 2007, vol. 2.

[14] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of Machine Learning Research*, vol. 4, pp. 1107–1149, 2003.

[15] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey." *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[16] J. D. R. Millán, D. Posenato, and E. Dedieu, "Continuous-action Q-learning," *Machine Learning*, vol. 49, pp. 247–265, 2002.

[17] A. Rodriguez, M. Gagliolo, P. Vrancx, R. Grau, and A. Nowé, "Improving the performance of continuous action reinforcement learning automata," in *European Workshop on Reinforcement Learning*, 2011.

[18] S. J. Bradtke and A. G. Barto, "Linear least-squares algorithms for temporal difference learning," *Machine Learning*, vol. 22, no. 1/2/3, pp. 33–57, 1996.

[19] S. Boyd and L. Vandenberghe, *Convex Optimization*. Canbridge University Press, 2004.

[20] T. Hester and P. Stone, "Learning and using models," in *Reinforcment Learning: State-of-the-Art*, M. Wiering and M. van Otterlo, Eds. Springer-Verlag Berlin Heidelberg, 2012, pp. 111–141.

[21] S. Mahadevan and M. Maggioni, "Proto-value functions: A laplacian framework for learning representations and control in markov decision provesses," *Journal of Machine Leanring Research*, vol. 8, pp. 2169–2231, 2007.

[22] M. Petrik, "An analysis of laplacian methods for value function approximation in mdps," in *International Joint Conference on Artificial Intelligence*, 2007, pp. 2574–2579.

[23] R. Parr, C. Painter-wakefield, L. Li, and M. Littman, "Analyzing feature generation for value-function approximation," in *International Conference on Machine Learning*, 2007.

[24] W. Böhmer, S. Grünewälder, Y. Shen, M. Musial, and K. Obermayer, "Construction of approximation spaces for reinforcement learning," *Journal of Machine Learning Research*, under review.

B.2 Böhmer and Obermayer, ECML/PKDD (2015b)

The article *Regression with linear factored functions* (Böhmer and Obermayer, 2015) has been presented at the the *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databased 2015*[1] in Porto, Portugal.

The article derives a regression algorithm that learns a linear combination of *factored functions* (LFF), similar to the factored basis functions I assumed in Böhmer and Obermayer (2013). My approach constructs the basis functions one after another in a greedy fashion. I show at the example of established benchmarks that the novel algorithm performs (in terms of the root mean squared error) comparable to *Gaussian processes*. The article specifies the framework to optimize LFF, that I discuss in greater detail in Section 5.1.2, and the regression Algorithm 8 derived in Section 5.2.2. Chapter 5 expands on this work and develops a multitude of additional algorithms for LFF.

The authors contributed in the following way:

- *Wendelin Böhmer* has written the entire article, derived the presented algorithm, and designed and performed all experiments.
- *Klaus Obermayer* has supervised both development and experimentation and gave comments to the article.

[1] The final publication is available at Springer via `http://dx.doi.org/10.1007/978-3-319-23528-8_8` .

Regression with Linear Factored Functions

Wendelin Böhmer and Klaus Obermayer

Neural Information Processing Group, Technische Universität Berlin,
Sekr. MAR5-6, Marchstr. 23, D-10587 Berlin, Germany
{wendelin,oby}@ni.tu-berlin.de, http://www.ni.tu-berlin.de

Abstract. Many applications that use empirically estimated functions face a *curse of dimensionality*, because integrals over most function classes must be approximated by sampling. This paper introduces a novel *regression*-algorithm that learns *linear factored functions* (LFF). This class of functions has structural properties that allow to analytically solve certain integrals and to calculate point-wise products. Applications like *belief propagation* and *reinforcement learning* can exploit these properties to break the curse and speed up computation. We derive a regularized greedy optimization scheme, that learns factored basis functions during training. The novel regression algorithm performs competitively to *Gaussian processes* on benchmark tasks, and the learned LFF functions are with 4-9 factored basis functions on average very compact.

Keywords: regression, factored functions, curse of dimensionality

1 Introduction

This paper introduces a novel regression-algorithm, which performs competitive to *Gaussian processes*, but yields *linear factored functions* (LFF). These have outstanding properties like analytical *point-wise products* and *marginalization*.

Regression is a well known problem, which can be solved by many non-linear architectures like *kernel methods* (Shawe-Taylor and Cristianini, 2004) or *neural networks* (Haykin, 1998). While these perform well, the estimated functions often suffer a *curse of dimensionality* in later applications. For example, computing an integral over a neural network or kernel function requires to sample the entire input space. Applications like *belief propagation* (Pearl, 1988) and *reinforcement learning* (Kaelbling et al., 1996), on the other hand, face large input spaces and require therefore efficient computations. We propose LFF for this purpose and showcase its properties in comparison to kernel functions.

1.1 Kernel regression

In the last 20 years, kernel methods like *support vector machines* (SVM, Vapnik, 1995; Boser et al., 1992) have become a de facto standard in various practical applications. This is mainly due to a sparse representation of the learned classifiers with so called *support vectors* (SV). The most popular kernel method for

regression, *Gaussian processes* (GP, see Rasmussen and Williams, 2006; Bishop, 2006), on the other hand, requires as many SV as training samples. Sparse versions of GP aim thus for a small subset of SV. Some select this set based on constraints similar to SVM (Vapnik, 1995; Tipping, 2001), while others try to conserve the spanned linear function space (*sparse GP*, Csató and Opper, 2002; Rasmussen and Williams, 2006). There exist also attempts to construct new SV by averaging similar training samples (e.g. Wang et al., 2012).

Well chosen SV for regression are usually not sparsely concentrated on a decision boundary as they are for SVM. In fact, many practical applications report that they are distributed uniformly in the input space (e.g. in Böhmer et al., 2013). Regression tasks restricted to a small region of the input space may tolerate this, but some applications require predictions everywhere. For example, the *value function* in reinforcement learning must be generalized to each state. The number of SV required to *represent* this function equally well in each state grows exponentially in the number of input-space dimensions, leading to Bellman's famous curse of dimensionality (Bellman, 1957).

Kernel methods derive their effectiveness from linear optimization in a non-linear *Hilbert space* of functions. Kernel-functions parameterized by SV are the non-linear *basis functions* in this space. Due to the functional form of the kernel, this can be a very ineffective way to select basis functions. Even in relatively small input spaces, it often takes hundreds or thousands SV to approximate a function sufficiently. To alleviate the problem, one can construct complex kernels out of simple prototypes (see a recent review in Gönen and Alpaydın, 2011).

1.2 Factored basis functions

Diverging from all above arguments, this article proposes a more radical approach: to construct the non-linear basis functions directly during training, without the detour over kernel functions and support vectors. This poses two main challenges: to select a *suitable functions space* and to *regularize the optimization* properly. The former is critical, as a small set of basis functions must be able to approximate any target function, but should also be easy to compute in practice.

We propose *factored functions* $\psi_i = \prod_k \psi_i^k \in \mathcal{F}$ as basis functions for regression, and call the linear combination of m of those bases a *linear factored function* $f \in \mathcal{F}^m$ (LFF, Section 3). For example, *generalized linear models* (Nelder and Wedderburn, 1972) and *multivariate adaptive regression splines* (MARS, Friedman, 1991) are both LFF. Computation remains feasible by using *hinge functions* $\psi_i^k(x_k) = \max(0, x_k - c)$ and restricting the scope of each factored function ψ_i. In contrast, we assume the general case without restrictions to functions or scope.

Due to their structure, LFF can solve certain integrals analytically and allow very efficient computation of point-wise products and marginalization. We show that our LFF are universal function approximators and derive an appropriate *regularization* term. This regularization promotes smoothness, but also retains a high degree of variability in densely sampled regions by linking smoothness to uncertainty about the sampling distribution. Finally, we derive a novel regression algorithm for LFF based on a greedy optimization scheme.

Functions learned by this algorithm (Algorithm 1, see pages 7 and 16) are very compact (between 3 and 12 bases on standard benchmarks) and perform competitive with Gaussian processes (Section 4). The paper finishes with a discussion of the computational possibilities of LFF in potential areas of application and possible extensions to *sparse regression* with LFF (Section 5).

2 Regression

Let $\{\boldsymbol{x}_t \in \mathcal{X}\}_{t=1}^n$ be a set of n *input samples*, i.i.d. drawn from an input set $\mathcal{X} \subset \mathbb{R}^d$. Each so called "training sample" is *labeled* with a real number $\{y_t \in \mathbb{R}\}_{t=1}^n$. *Regression* aims to find a function $f : \mathcal{X} \to \mathbb{R}$, that predicts the labels to all (previously unseen) test samples as well as possible. Labels may be afflicted by *noise* and f must thus approximate the mean label of each sample, i.e., the function $\mu : \mathcal{X} \to \mathbb{R}$. It is important to notice that *conceptually* the noise is introduced by two (non observable) sources: noisy labels y_t and noisy samples $\boldsymbol{x}_t$. The latter will play an important role for regularization. We define the conditional distribution χ of observable samples $\boldsymbol{x} \in \mathcal{X}$ given the non-observable "true" samples $\boldsymbol{z} \in \mathcal{X}$, which are drawn by a distribution ξ. In the limit of infinite samples, the *least squares* cost-function $\mathcal{C}[f|\chi, \mu]$ can thus be written as

$$\lim_{n\to\infty} \inf_f \frac{1}{n} \sum_{t=1}^{n} \Big(f(\boldsymbol{x}_t) - y_t\Big)^2 \quad = \quad \inf_f \iint \xi(d\boldsymbol{z})\, \chi(d\boldsymbol{x}|\boldsymbol{z}) \Big(f(\boldsymbol{x}) - \mu(\boldsymbol{z})\Big)^2 . \quad (1)$$

The cost function $\mathcal{C}$ can never be computed *exactly*, but *approximated* using the training samples[1] and assumptions about the unknown noise distribution χ.

3 Linear factored functions

Any non-linear function can be expressed as a linear function $f(\boldsymbol{x}) = \boldsymbol{a}^\top \boldsymbol{\psi}(\boldsymbol{x})$, $\forall \boldsymbol{x} \in \mathcal{X}$, with m non-linear basis functions $\psi_i : \mathcal{X} \to \mathbb{R}$, $\forall i \in \{1 \ldots, m\}$. In this section we will define *linear factored functions* (LFF), that have *factored basis functions* $\psi_i(\boldsymbol{x}) := \psi_i^1(x_1) \cdot \ldots \cdot \psi_i^d(x_d) \in \mathcal{F}$, a regularization method for this function class and an algorithm for regression with LFF.

3.1 Function class

We define the class of linear factored functions $f \in \mathcal{F}^m$ as a linear combination (with linear parameters $\boldsymbol{a} \in \mathbb{R}^m$) of m factored basis functions $\psi_i : \mathcal{X} \to \mathbb{R}$ (with parameters $\{\mathbf{B}^k \in \mathbb{R}^{m_k \times m}\}_{k=1}^d$):

$$f(\boldsymbol{x}) \;:=\; \boldsymbol{a}^\top \boldsymbol{\psi}(\boldsymbol{x}) \;:=\; \boldsymbol{a}^\top \Big[\prod_{k=1}^{d} \boldsymbol{\psi}^k(x_k)\Big] \;:=\; \sum_{i=1}^{m} a_i \prod_{k=1}^{d} \sum_{j=1}^{m_k} B_{ji}^k\, \phi_j^k(x_k)\,. \quad (2)$$

[1] The distribution ξ of "true" samples $\boldsymbol{z}$ can *not* be observed. We approximate in the following ξ with the training-sample distribution. This may be justified if the sample-noise χ is comparatively small. Although not strictly rigorous, the presented formalism helps to put the regularization derived in Proposition 2 into perspective.

4

LFF are formally defined in Appendix A. In short, a basis function ψ_i is the *point-wise product* of one-dimensional functions ψ_i^k in each input dimension k. These are themselves constructed as linear functions of a corresponding one-dimensional base $\{\phi_j^k\}_{j=1}^{m_k}$ over that dimension and ideally can approximate arbitrary functions[2]. Although each factored function ψ_i is very restricted, a linear combination of them can be very powerful:

Corollary 1 *Let $\mathcal{X}_k$ be a bounded continuous set and ϕ_j^k the j'th Fourier base over $\mathcal{X}_k$. In the limit of $m_k \to \infty, \forall k \in \{1, \dots, d\}$, holds $\mathcal{F}^\infty = L^2(\mathcal{X}, \vartheta)$.*

Strictly this holds in the limit of infinitely many basis functions ψ_i, but we will show empirically that there exist close approximations with a small number m of factored functions. One can make similar statements for other bases $\{\phi_j^k\}_{j=1}^\infty$. For example, for Gaussian kernels one can show that the space $\mathcal{F}^\infty$ is in the limit equivalent to the corresponding *reproducing kernel Hilbert space* $\mathcal{H}$.

LFF offer some structural advantages over other universal function approximation classes like neural networks or reproducing kernel Hilbert spaces. Firstly, the *inner product* of two LFF in $L^2(\mathcal{X}, \vartheta)$ can be computed as products of one-dimensional integrals. For some bases[3], these integrals can be calculated analytically without any sampling. This could in principle break the curse of dimensionality for algorithms that have to approximate these inner products numerically. For example, input variables can be *marginalized* (integrated) out analytically (Equation 9 on Page 12). Secondly, the *point-wise product* of two LFF is a LFF as well[4] (Equation 10 on Page 13). See Appendix A for details. These properties are very useful, for example in *belief propagation* (Pearl, 1988) and *factored reinforcement learning* (Böhmer and Obermayer, 2013).

3.2 Constraints

LFF have some degrees of freedom that can impede optimization. For example, the norm of $\psi_i \in \mathcal{F}$ does not influence function $f \in \mathcal{F}^m$, as the corresponding linear coefficients a_i can be scaled accordingly. We can therefore introduce the *constraints* $\|\psi_i\|_\vartheta = 1, \forall i$, without restriction to the function class. The factorization of inner products (see Appendix A on Page 12) allows us furthermore to rewrite the constraints as $\|\psi_i\|_\vartheta = \prod_k \|\psi_i^k\|_{\vartheta^k} = 1$. This holds as long as the product is one, which exposes another unnecessary degree of freedom. To finally make the solution unique (up to permutation), we define the constraints as $\|\psi_i^k\|_{\vartheta^k} = 1, \forall k, \forall i$. Minimizing some $\mathcal{C}[f]$ w.r.t. $f \in \mathcal{F}^m$ is thus equivalent to

$$\inf_{f \in \mathcal{F}^m} \mathcal{C}[f] \qquad \text{s.t.} \quad \|\psi_i^k\|_{\vartheta^k} = 1\,, \quad \forall k \in \{1, \dots, d\}\,, \quad \forall i \in \{1, \dots, m\}\,. \tag{3}$$

[2] Examples are Fourier bases, Gaussian kernels or hinge-functions as in MARS.

[3] E.g. Fourier bases for continuous, and Kronecker-delta bases for discrete variables.

[4] One can use the trigonometric product-to-sum identities for Fourier bases or the Kronecker delta for discrete bases to construct LFF from a point-wise product without changing the underlying basis $\{\{\phi_i^k\}_{i=1}^{m_k}\}_{k=1}^d$.

The *cost function* $\mathcal{C}[f|\chi,\mu]$ of Equation 1 with the constraints in Equation 3 is equivalent to *ordinary least squares* (OLS) w.r.t. linear parameters $\boldsymbol{a} \in \mathbb{R}^m$. However, the optimization problem is *not* convex w.r.t. the parameter space $\{\mathbf{B}^k \in \mathbb{R}^{m_k \times m}\}_{k=1}^d$, due to the nonlinearity of products.

Instead of tackling the global optimization problem induced by Equation 3, we propose a *greedy* approximation algorithm. Here we optimize at iteration $\hat{\imath}$ one linear basis function $\psi_{\hat{\imath}} =: g =: \prod_k g^k \in \mathcal{F}$, with $g^k(x_k) =: \boldsymbol{b}^{k\top}\boldsymbol{\phi}^k(x_k)$, at a time, to fit the residual $\mu - f$ between the true *mean label* function $\mu \in L^2(\mathcal{X},\vartheta)$ and the current regression estimate $f \in \mathcal{F}^{\hat{\imath}-1}$, based on all $\hat{\imath}-1$ previously constructed factored basis functions $\{\psi_i\}_{i=1}^{\hat{\imath}-1}$:

$$\inf_{g\in\mathcal{F}} \mathcal{C}[f+g|\chi,\mu] \qquad \text{s.t.} \quad \|g^k\|_{\vartheta^k} = 1\,, \quad \forall k \in \{1,\dots,d\}\,. \tag{4}$$

3.3 Regularization

Regression with any powerful function class requires regularization to avoid overfitting. Examples are *weight decay* for neural networks (Haykin, 1998) or parameterized *priors* for Gaussian processes. It is, however, not immediately obvious how to regularize the parameters of a LFF and we will derive a regularization term from a Taylor approximation of the cost function in Equation 1.

We aim to enforce smooth functions, especially in those regions our knowledge is limited due to a lack of training samples. This *uncertainty* can be expressed as the *Radon-Nikodym derivative*[5] $\frac{\vartheta}{\xi} : \mathcal{X} \to [0,\infty)$ of our factored measure ϑ (see Appendix A) w.r.t. the sampling distribution ξ. Figure 1 demonstrates at the example of a uniform distribution ϑ how $\frac{\vartheta}{\xi}$ reflects our empirical knowledge of the input space $\mathcal{X}$.

We use this uncertainty to modulate the *sample noise distribution* χ in Equation 1. This means that frequently sampled regions of $\mathcal{X}$ shall yield low, while scarcely sampled regions shall yield high variance. Formally, we assume $\chi(d\boldsymbol{x}|\boldsymbol{z})$ to be a Gaussian probability measure over $\mathcal{X}$ with mean $\boldsymbol{z}$ and a *covariance matrix* $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$, scaled by the local uncertainty in $\boldsymbol{z}$ (modeled as $\frac{\vartheta}{\xi}(\boldsymbol{z})$):

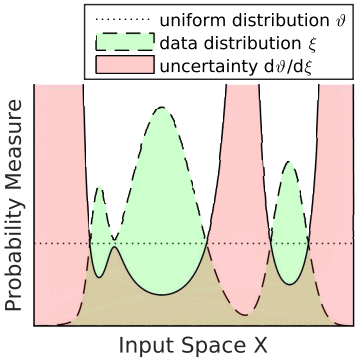


Fig. 1. We interpret the Radon-Nikodym derivative $\frac{d\vartheta}{d\xi}$ as *uncertainty measure* for our knowledge of $\mathcal{X}$. Regularization enforces smoothness in uncertain regions.

$$\int \chi(d\boldsymbol{x}|\boldsymbol{z})(\boldsymbol{x}-\boldsymbol{z}) = \mathbf{0}\,, \quad \int \chi(d\boldsymbol{x}|\boldsymbol{z})(\boldsymbol{x}-\boldsymbol{z})(\boldsymbol{x}-\boldsymbol{z})^\top = \tfrac{\vartheta}{\xi}(\boldsymbol{z})\cdot\boldsymbol{\Sigma}\,, \quad \forall \boldsymbol{z} \in \mathcal{X}\,. \tag{5}$$

[5] Technically we have to assume that ϑ is *absolutely continuous* in respect to ξ. For "well-behaving" distributions ϑ, like the uniform or Gaussian distributions we discuss in Appendix A, this is equivalent to the assumption that in the limit of infinite samples, each sample $\boldsymbol{z} \in \mathcal{X}$ will *eventually* be drawn by ξ.

6

In the following we assume without loss of generality[6] the matrix $\mathbf{\Sigma}$ to be diagonal, with the diagonal elements called $\sigma_k^2 := \Sigma_{kk}$.

Proposition 2 *Under the assumptions of Equation 5 and a diagonal covariance matrix $\mathbf{\Sigma}$, the first order Taylor approximation of the cost $\mathcal{C}$ in Equation 4 is*

$$\tilde{\mathcal{C}}[g] \quad := \quad \underbrace{\|g - (\mu - f)\|_{\xi}^2}_{\text{sample-noise free cost}} \; + \; \sum_{k=1}^{d} \sigma_k^2 \underbrace{\|\tfrac{\partial}{\partial x_k} g + \tfrac{\partial}{\partial x_k} f\|_{\vartheta}^2}_{\text{smoothness in dimension } k} . \tag{6}$$

Proof: see Appendix C on Page 14. □

Note that the approximated cost $\tilde{\mathcal{C}}[g]$ consists of the sample-noise free cost (measured w.r.t. training distribution ξ) and d regularization terms. Each term prefers functions that are smooth[7] in one input dimension. This enforces smoothness everywhere, but allows exceptions where enough data is available. To avoid a cluttered notation, in the following we will use the symbol $\nabla_k f := \frac{\partial}{\partial x_k} f$.

3.4 Optimization

Another advantage of cost function $\tilde{\mathcal{C}}[g]$ is that one can optimize one factor function g^k of $g(\boldsymbol{x}) = g^1(x_1) \cdot \ldots \cdot g^d(x_d) \in \mathcal{F}$ at a time, instead of time consuming *gradient descend* over the entire parameter space of g. To be more precise:

Proposition 3 *If all but one factor function g^k are considered constant, Equation 6 has an analytical solution. If $\{\phi_j^k\}_{j=1}^{m_k}$ is a Fourier base, $\sigma_k^2 > 0$ and $\vartheta \ll \xi$, then the solution is also unique.*

Proof: see Appendix C on Page 15. □

One can give similar guarantees for other bases, e.g. Gaussian kernels. Note that Proposition 3 does *not* state that the optimization problem has a unique solution in $\mathcal{F}$. Formal convergence statements are not trivial and empirically the parameters of g do not converge, but evolve around orbits of equal cost instead. However, since the optimization of *any* g^k cannot increase the cost, any sequence of improvements will converge to (and stay in) a *local minimum*. This implies a *nested* optimization approach, that is formulated in Algorithm 1 on Page 16:

– An *inner loop* that optimizes one factored basis function $g(\boldsymbol{x}) = g^1(x_1) \cdot \ldots \cdot g^d(x_d)$ by selecting an input dimension k in each iteration and solve Equation 6 for the corresponding g^k. A detailed derivation of the optimization steps of the inner loop is given in Appendix B on Page 13. The choice of k influences

[6] Non-diagonal covariance matrices $\mathbf{\Sigma}$ can be cast in this framework by projecting the input samples into the eigenspace of $\mathbf{\Sigma}$ (thus diagonalizing the input) and use the corresponding eigenvalues λ_k instead of the regularization parameters σ_k^2's.

[7] Each regularization term is measured w.r.t. the factored distribution ϑ. We also tested the algorithm without consideration of "uncertainty" $\frac{\vartheta}{\xi}$, i.e., by measuring each term w.r.t. ξ. As a result, regions outside the hypercube containing the training set were no longer regularized and predicted arbitrary (often extreme) values.

Algorithm 1 (abstract) – a detailed version can be found on Page 16

```
while new factored basis function can improve solution do
  initialize new basis function g as constant function
  while optimization improves cost in Equation 6 do
    for random input dimension k do
      calculate optimal solution for g^k without changing g^l, ∀l ≠ k
    end for
  end while   // new basis function g has converged
  add g to set of factored basis functions and solve OLS
end while     // regression has converged
```

the solution in a non-trivial way and further research is needed to build up a rationale for any meaningful decision. For the purpose of this paper, we assume k to be chosen randomly by permuting the order of updates.
The *computational complexity* of the inner loop is $\mathcal{O}(m_k^2 n + d^2 m_k m)$. Memory complexity is $\mathcal{O}(d\, m_k m)$, or $\mathcal{O}(d\, m_k n)$ with the optional cache speedup of Algorithm 1. The loop is repeated for random k until the cost-improvements of all dimensions k fall below some small ϵ.

– After convergence of the inner loop in (outer) iteration $\hat{\imath}$, the new basis function is $\psi_{\hat{\imath}} := g$. As the basis has changed, the linear parameters $\boldsymbol{a} \in \mathbb{R}^{\hat{\imath}}$ have to be readjusted by solving the ordinary least squares problem

$$\boldsymbol{a} \;=\; (\boldsymbol{\Psi}\boldsymbol{\Psi}^\top)^{-1}\boldsymbol{\Psi}\boldsymbol{y}\,, \;\text{ with }\; \Psi_{it} \;:=\; \psi_i(\boldsymbol{x}_t)\,,\; \forall i \in \{1,\dots,\hat{\imath}\}\,,\; \forall t \in \{1,\dots,n\}\,.$$

We propose to stop the approximation when the newly found basis function $\psi_{\hat{\imath}}$ is no longer *linearly independent* of the current basis $\{\psi_i\}_{i=1}^{\hat{\imath}-1}$. This can for example be tested by comparing the *determinant* $\det(\frac{1}{n}\boldsymbol{\Psi}\boldsymbol{\Psi}^\top) < \varepsilon$, for some very small ε.

4 Empirical evaluation

In this section we will evaluate the novel LFF regression Algorithm 1, printed in detail on Page 16. We will analyze its properties on low dimensional toy-data, and compare its performance with sparse and traditional Gaussian processes (GP, see Rasmussen and Williams, 2006; Bishop, 2006).

4.1 Demonstration

To showcase the novel Algorithm 1, we tested it on an artificial two-dimensional regression toy-data set. The $n = 1000$ training samples were drawn from a noisy spiral and labeled with a sinus. The variance of the Gaussian sample-noise grew with the spiral as well:

$$\boldsymbol{x}_t \;=\; 6\tfrac{t}{n}\begin{bmatrix}\cos\left(6\frac{t}{n}\pi\right)\\ \sin\left(6\frac{t}{n}\pi\right)\end{bmatrix} + \mathcal{N}\Big(\mathbf{0}, \tfrac{t^2}{4n^2}\mathbf{I}\Big)\,,\quad y_t \;=\; \sin\Big(4\tfrac{t}{n}\pi\Big)\,,\quad \forall t \in \{1,\dots,n\}\,. \tag{7}$$

8

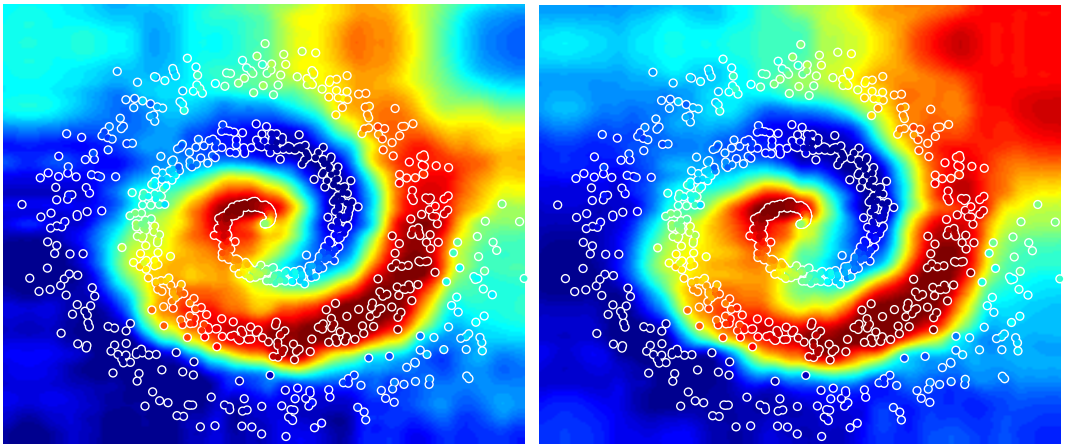


Fig. 2. Two LFF functions learned from the same 1000 training samples (white circles). The color inside a circle represents the training label. Outside the circles, the color represents the prediction of the LFF function. The differences between both functions are rooted in the randomized order in which the factor functions g^k are updated. However, the similarity of the sampled region indicates that poor initial choices can be compensated by subsequently constructed basis functions.

Figure 2 shows one training set plotted over two learned[8] functions $f \in \mathcal{F}^m$ with $m = 21$ and $m = 24$ factored basis functions, respectively. Regularization constants were in both cases $\sigma_k^2 = 0.0005, \forall k$. The differences between the functions stem from the randomized order in which the factor functions g^k are updated. Note that the sampled regions have similar predictions. Regions with strong differences, for example the upper right corner, are never seen during training.

In all our experiments, Algorithm 1 always converged. Runtime was mainly influenced by the input dimensionality ($\mathcal{O}(d^2)$), the number of training samples ($\mathcal{O}(n)$) and the eventual number of basis functions ($\mathcal{O}(m)$). The latter was strongly correlated with approximation quality, i.e., bad approximations converged fast. Cross-validation was therefore able to find good parameters efficiently and the resulting LFF were always very similar near the training data.

4.2 Evaluation

We compared the regression performance of LFF and GP with cross-validation on five regression benchmarks from the *UCI Manchine Learning Repository*[9]:

– The *concrete compressive strength* data set (*concrete*, Yeh, 1998) consists of $n = 1030$ samples with $d = 8$ dimensions describing various concrete

[8] Here (and in the rest of the paper), each variable was encoded with 50 Fourier cosine bases. We tested other sizes as well. Few cosine bases result effectively in a low-pass filtered function, whereas every experiment with more than 20 or 30 bases behaved very similar. We tested up to $m_k = 1000$ bases and did not experience over-fitting.

[9] `https://archive.ics.uci.edu/ml/index.html`

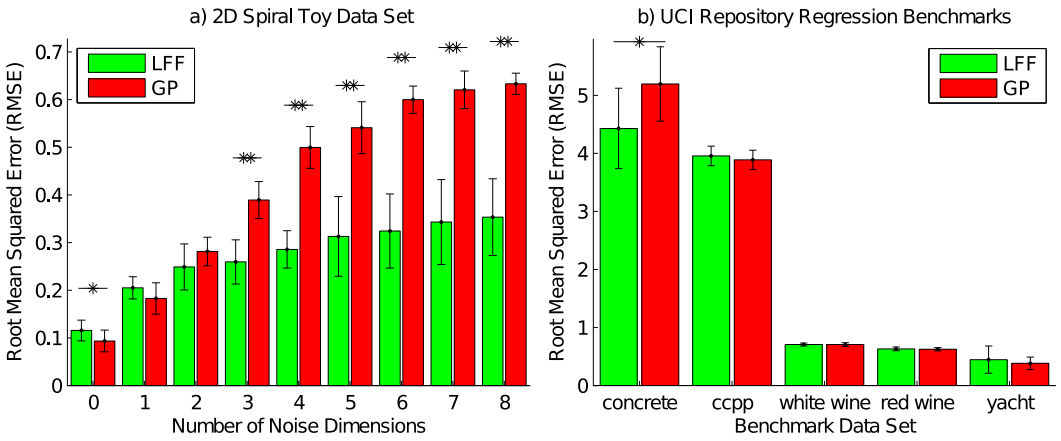


Fig. 3. Mean and standard deviation within a 10-fold cross-validation of a) the toy data set with additional independent noise input dimensions and b) all tested UCI benchmark data sets. The stars mark *significantly* different distribution of RMSE over all folds in both a *paired-sample t-test* and a *Wilcoxon signed rank test*. Significance levels are: one star $p < 0.05$, two stars $p < 0.005$.

mixture-components. The target variable is the real-valued compression strength of the mixture after it hardened.

- The *combined cycle power plant* data set (*ccpp*, Tüfekci, 2014) consists of $n = 9568$ samples with $d = 4$ dimensions describing 6 years worth of measurements from a combined gas and steam turbine. The real-valued target variable is the energy output of the system.
- The *wine quality* data set (Cortez et al., 2009) consists of two subsets with $d = 11$ dimensions each, which describe physical attributes of various white and red wines: the set contains $n = 4898$ samples of *white wine* and $n = 1599$ samples of *red wine*. The target variable is the estimated wine quality on a discrete scale from 0 to 10.
- The *yacht hydrodynamics* data set (*yacht*, Gerritsma et al., 1981) consists of $n = 308$ samples with $d = 6$ dimensions describing parameters of the *Delft yacht hull* ship-series. The real-valued target variable is the residuary resistance measured in full-scale experiments.

To demonstrate the advantage of factored basis functions, we also used the 2d-spiral toy-data set of the previous section with a varying number of additional input dimensions. Additional values were drawn i.i.d. from a Gaussian distribution and are thus independent of the target labels. As the input space $\mathcal{X}$ grows, kernel methods will increasingly face the curse of dimensionality during training.

Every data-dimension (except the labels) have been translated and scaled to zero mean and unit-variance before training. Hyper-parameters were chosen w.r.t. the mean of a 10-fold cross-validation. LFF-regression was tested for the uniform noise-parameters $\sigma_k^2 \in \{10^{-10}, 10^{-9.75}, 10^{-9.5}, \dots, 10^{10}\}, \forall k$, i.e. for 81 different hyper-parameters. GP were tested with Gaussian kernels $\kappa(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\frac{1}{2\bar{\sigma}^2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2)$ using kernel parameters $\bar{\sigma} \in \{10^{-1}, 10^{-3/4}, 10^{-1/2}, \dots, 3\}$ and prior-parameters $\beta \in \{10^{-2}, 10^{-1}, \dots, 10^{10}\}$ (see Bishop, 2006, for the def-

10

Table 1. 10-fold cross-validation RMSE for benchmark data sets with d dimensions and n samples, resulting in m *basis functions*. The cross-validation took h hours.

DATA SET	d	n	#SV	RMSE LFF	RMSE GP	m LFF	h LFF	h GP
Concrete	8	1030	927	$\mathbf{4.429 \pm 0.69}$	5.196 ± 0.64	4.2 ± 0.8	3.00	0.05
CCPP	4	9568	2000	3.957 ± 0.17	3.888 ± 0.17	8.8 ± 2.0	1.96	1.14
White Wine	11	4898	2000	0.707 ± 0.02	0.708 ± 0.03	4.2 ± 0.4	4.21	0.69
Red Wine	11	1599	1440	0.632 ± 0.03	0.625 ± 0.03	4.7 ± 0.7	3.25	0.13
Yacht	6	308	278	0.446 ± 0.23	0.383 ± 0.11	4.2 ± 0.6	0.43	0.005

inition), i.e. for 221 different hyper-parameter combinations. The number of support vectors in standard GP equals the number of training samples. As this is not feasible for larger data sets, we used the MP-MAH algorithm (Böhmer et al., 2012) to select a uniformly distributed subset of 2000 training samples for sparse GP (Rasmussen and Williams, 2006).

Figure 3a demonstrates the advantage of factored basis functions over kernel methods during training. The plot shows the *root mean squared errors*[10] (RMSE) of the two dimensional spiral toy-data set with an increasing number of independent noise dimensions. GP solves the initial task better, but clearly succumbs to the curse of dimensionality, as the size of the input space $\mathcal{X}$ grows. LFF, on the other hand, significantly overtake GP from 3 noise dimensions on, as the factored basis functions appear to be less affected by the curse. Another difference to GP is that decreasing performance automatically yields less factored basis functions (from 19.9 ± 2.18 with 0, to 6.3 ± 0.48 bases with 8 noise dimensions).

Figure 3b and Table 1 show that our LFF algorithm performs on all evaluated real-world benchmark data sets comparable to (sparse) GP. RMSE distributions over all folds were statistically indistinguishable, except for an advantage of LFF regression in the concrete compressive strength data set ($p < 0.01$ in a *t-test* and $p < 0.02$ in a *signed rank test*). As each basis function requries many iterations to converge, LFF regression runs considerably longer than standard approaches. However, LFF require between 3 and 12 factored basis functions to achieve the *same* performance as GP with 278-2000 kernel basis functions.

5 Discussion

We presented a novel algorithm for regression, which constructs factored basis functions during training. As *linear factored functions* (LFF) can in principle approximate any function in $L^2(\mathcal{X}, \vartheta)$, a regularization is necessary to avoid over-fitting. Here we rely on a regularization scheme that has been motivated by a Taylor approximation of the least-squares cost function with (an approximation

[10] RMSE are not a common performance metric for GP, which represent a *distribution* of solutions. However, RMSE reflect the objective of regression and are well suited to compare our algorithm with the *mean* of a GP.

of) virtual sample-noise. RMSE performance appears comparable to Gaussian processes on real-world benchmark data sets, but the factored representation is considerably more compact and seems to be less affected by distractors.

At the moment, LFF optimization faces two challenges. (i) The optimized cost function is not convex, but the local minimum of the solution may be controlled by selecting the next factor function to optimize. For example, MARS successively adds factor functions. Generalizing this will require further research, but may also allow some performance guarantees. (ii) The large number of inner-loop iterations make the algorithm slow. This problem should be mostly solved by addressing (i), but finding a trade-off between approximation quality and runtime may also provide a less compact shortcut with similar performance.

Preliminary experiments also demonstrated the viability of LFF in a *sparse regression* approach. Sparsity refers here to a limited number of input-dimensions that affect the prediction, which can be implemented by adjusting the sample-noise parameters σ_k^2 during training for each variable $\mathcal{X}_k$ individually. This is of particular interest, as factored functions are ideally suited to represent sparse functions and are in principle *unaffected* by the curse of dimensionality in function representation. Our approach modified the cost function to enforce LFF functions that were constant in all noise-dimensions. We did not include our results in this paper, as choosing the first updated factor functions g^k poorly resulted in basis functions that rather fitted noise than predicted labels. When we enforce sparseness, this initial mistake can afterwards no longer be rectified by other basis functions, in difference to the presented Algorithm 1. However, if this can be controlled by a sensible order in the updates, the resulting algorithm should be much faster and more robust than the presented version.

There are many application areas that may exploit the structural advantages of LLF. In *reinforcement learning* (Kaelbling et al., 1996), one can exploit the factorizing inner products to break the curse of dimensionality of the state space (Böhmer and Obermayer, 2013). Factored transition models also need to be learned from experience, which is essentially a sparse regression task. Another possible field of application are *junction trees* (for Bayesian inference, see e.g. Bishop, 2006) over continuous variables, where sparse regression may estimate the conditional probabilities. In each node one must also marginalize out variables, or calculate the point-wise product over multiple functions. Both operations can be performed analytically with LFF, the latter at the expense of more basis functions in the resulting LFF. However, one can use our framework to *compress* these functions after multiplication. This would allow junction-tree inference over mixed continuous and discrete variables.

In summary, we believe our approach to approximate functions by constructing non-linear factored basis functions (LFF) to be very promising. The presented algorithm performs comparable with Gaussian processes, but appears less sensitive to large input spaces than kernel methods. We also discussed some potential extensions for sparse regression that should improve upon that, in particular on runtime, and gave some fields of application that would benefit greatly from the algebraic structure of LFF.

Acknowledgments The authors thank Yun Shen and the anonymous reviewers for their helpful comments. This work was funded by the *German science foundation* (DFG) within SPP 1527 *autonomous learning*.

Appendix A LFF definition and properties

Let $\mathcal{X}_k$ denote the subset of $\mathbb{R}$ associated with the k'th variable of input space $\mathcal{X} \subset \mathbb{R}^d$, such that $\mathcal{X} := \mathcal{X}_1 \times \ldots \times \mathcal{X}_d$. To avoid the curse of dimensionality in this space, one can integrate w.r.t. a *factored probability measure* ϑ, i.e. $\vartheta(d\boldsymbol{x}) = \prod_{k=1}^d \vartheta^k(dx_k), \int \vartheta^k(dx_k) = 1, \forall k$. For example, ϑ^k could be uniform or Gaussian distributions over $\mathcal{X}_k$ and the resulting ϑ would be a uniform or Gaussian distribution over the input space $\mathcal{X}$.

A function $g : \mathcal{X} \to \mathbb{R}$ is called a *factored function* if it can be written as a product of one-dimensional *factor functions* $g^k : \mathcal{X}_k \to \mathbb{R}$, i.e. $g(\boldsymbol{x}) = \prod_{k=1}^d g^k(x_k)$. We only consider factored functions g that are twice integrable w.r.t. measure ϑ, i.e. $g \in L^2(\mathcal{X}, \vartheta)$. Note that not all functions $f \in L^2(\mathcal{X}, \vartheta)$ are factored, though. Due to *Fubini's theorem* the d-dimensional inner product between two factored functions $g, g' \in L^2(\mathcal{X}, \vartheta)$ can be written as the product of d one-dimensional inner products:

$$\langle g, g' \rangle_\vartheta = \int \vartheta(d\boldsymbol{x})\, g(\boldsymbol{x})\, g'(\boldsymbol{x}) = \int \prod_{k=1}^d \vartheta^k(dx_k)\, g^k(dx_k)\, g'^k(dx_k) = \prod_{k=1}^d \langle g^k, g'^k \rangle_{\vartheta^k} .$$

This trick can be used to solve the integrals at the heart of many least-squares algorithms. Our aim is to *learn* factored basis functions ψ_i. To this end, let $\{\phi_j^k : \mathcal{X}_k \to \mathbb{R}\}_{j=1}^{m_k}$ be a well-chosen[11] (i.e. universal) basis on $\mathcal{X}_k$, with the space of linear combinations denoted by $\mathcal{L}_\phi^k := \{\boldsymbol{b}^\top \boldsymbol{\phi}^k | \boldsymbol{b} \in \mathbb{R}^{m_k}\}$. One can thus approximate factor functions of ψ_i in $\mathcal{L}_\phi^k$, i.e., as linear functions

$$\psi_i^k(x_k) \quad := \quad \sum_{j=1}^{m_k} B_{ji}^k\, \phi_j^k(x_k) \quad \in \quad \mathcal{L}_\phi^k\,, \qquad\qquad \mathbf{B}^k \;\in\; \mathbb{R}^{m_k \times m}\,. \tag{8}$$

Let $\mathcal{F}$ be the space of all factored basis functions ψ_i defined by the factor functions ψ_i^k above, and $\mathcal{F}^m$ be the space of all linear combinations of those m factored basis functions (Equation 2).

Marginalization of LFF can be performed analytically with Fourier bases ϕ_j^k and uniform distribution ϑ (many other bases can be analytically solved as well):

$$\int \vartheta^l(dx_l)\, f(\boldsymbol{x}) = \sum_{i=1}^m \Big(a_i \sum_{j=1}^{m_l} B_{ji}^l \underbrace{\langle \phi_j^l, 1 \rangle_{\vartheta^l}}_{\text{mean of } \phi_j^l} \Big) \Big[\prod_{k \neq l}^d \psi_i^k \Big] \overset{\text{Fourier}}{=} \sum_{i=1}^m \underbrace{a_i B_{1i}^l}_{\text{new } a_i} \Big[\prod_{k \neq l}^d \psi_i^k \Big]\,. \tag{9}$$

[11] Examples for continuous variables $\mathcal{X}_k$ are Fourier cosine bases $\phi_j^k(x_k) \sim \cos\big((j-1)\,\pi\, x_k\big)$, and Gaussian bases $\phi_j^k(x_k) = \exp\big(\frac{1}{2\sigma^2}(x_k - s_{kj})^2\big)$. Discrete variables may be represented with Kronecker-delta bases $\phi_j^k(x_k = i) = \delta_{ij}$.

Using the trigonometric *product-to-sum* identity $\cos(x)\cdot\cos(y) = \frac{1}{2}\big(\cos(x-y)+\cos(x+y)\big)$, one can also compute the point-wise product between two LFF f and $\bar{f}$ with cosine-Fourier base (solutions to other Fourier bases are less elegant):

$$\tilde{f}(\boldsymbol{x}) \;:=\; f(\boldsymbol{x})\cdot\bar{f}(\boldsymbol{x}) \overset{\text{Fourier}}{=} \sum_{i,j=1}^{m\bar{m}} \underbrace{a_i\,\bar{a}_j}_{\text{new } \tilde{a}_t} \prod_{k=1}^{d}\sum_{l=1}^{2m_k} \Big(\overbrace{\tfrac{1}{2}\sum_{q=1}^{l-1} B^k_{qi}\,\bar{B}^k_{(l-q)j} + \tfrac{1}{2}\sum_{q=l+1}^{m_k} B^k_{qi}\,\bar{B}^k_{(q-l)j}}^{\text{new } \tilde{B}^k_{lt}}\Big)\,\phi^k_l(x_k)\,, \quad (10)$$

where $t := (i-1)\,\bar{m}+j$, and $B^k_{ji} := 0, \forall j > m_k$, for both f and $\bar{f}$. Note that this increases the number of basis functions $\tilde{m} = m\bar{m}$, and the number of bases $\tilde{m}_k = 2m_k$ for each respective input dimension. The latter can be counteracted by *low-pass filtering*, i.e., by setting $\tilde{B}^k_{ji} := 0, \forall j > m_k$.

Appendix B Inner loop derivation

Here we will optimize the problem in Equation 6 for one variable $\mathcal{X}_k$ at a time, by describing the update step $g^k \leftarrow g'^k$. This is repeated with randomly chosen variables k, until convergence of the cost $\tilde{\mathcal{C}}[g]$, that is, until all possible updates decrease the cost less than some small ϵ.

Let in the following $\mathbf{C}^k := \langle\boldsymbol{\phi}^k, \boldsymbol{\phi}^{k\top}\rangle_{\vartheta^k}$ and $\dot{\mathbf{C}}^k := \langle\nabla_k\boldsymbol{\phi}^k, \nabla_k\boldsymbol{\phi}^{k\top}\rangle_{\vartheta^k}$ denote covariance matrices, and $\boldsymbol{R}^k_l := \frac{\partial}{\partial \boldsymbol{b}^k}\langle\nabla_l g, \nabla_l f\rangle_\vartheta$ denote the derivative of one regularization term. Note that for some choices of bases $\{\phi^k_j\}_{j=1}^{m_k}$, one can compute the covariance matrices analytically before the main algorithm starts, e.g. Fourier cosine bases have $C^k_{ij} = \delta_{ij}$ and $\dot{C}^k_{ij} = (i-1)^2\,\pi^2\,\delta_{ij}$.
The approximated cost function in Equation 6 is

$$\tilde{\mathcal{C}}[g] \;=\; \|g\|^2_\xi - 2\langle g, \mu - f\rangle_\xi + \|\mu - f\|^2_\xi + \sum_{k=1}^{d}\sigma_k^2\Big(\|\nabla_k g\|^2_\vartheta + 2\langle\nabla_k g, \nabla_k f\rangle_\vartheta + \|\nabla_k f\|^2_\vartheta\Big)\,.$$

The non-zero gradients of all inner products of this equation w.r.t. parameter vector $\boldsymbol{b}^k \in \mathbb{R}^{m_k}$ are

$$\begin{aligned}
\tfrac{\partial}{\partial \boldsymbol{b}^k}\langle g, g\rangle_\xi &= 2\,\langle\boldsymbol{\phi}^k\cdot\prod_{l\neq k} g^l, \prod_{l\neq k} g^l\cdot\boldsymbol{\phi}^{k\top}\rangle_\xi \boldsymbol{b}^k\,,\\
\tfrac{\partial}{\partial \boldsymbol{b}^k}\langle g, \mu - f\rangle_\xi &= \langle\boldsymbol{\phi}^k\cdot\prod_{l\neq k} g^l, \mu - f\rangle_\xi\,,\\
\tfrac{\partial}{\partial \boldsymbol{b}^k}\langle\nabla_l g, \nabla_l g\rangle_\vartheta &= \tfrac{\partial}{\partial \boldsymbol{b}^k}\langle\nabla_l g^l, \nabla_l g^l\rangle_{\vartheta^l}\prod_{s\neq l}\overbrace{\langle g^s, g^s\rangle_{\vartheta^s}}^{1} \;=\; 2\,\delta_{kl}\,\dot{\mathbf{C}}^k\boldsymbol{b}^k\,,\\
\boldsymbol{R}^k_l \;:=\; \tfrac{\partial}{\partial \boldsymbol{b}^k}\langle\nabla_l g, \nabla_l f\rangle_\vartheta &= \begin{cases} \dot{\mathbf{C}}^k\mathbf{B}^k\Big[\boldsymbol{a}\cdot\prod_{s\neq k}\mathbf{B}^{s\top}\mathbf{C}^s\boldsymbol{b}^s\Big] & \text{, if } \; k = l\\ \mathbf{C}^k\mathbf{B}^k\Big[\boldsymbol{a}\cdot\mathbf{B}^{l\top}\dot{\mathbf{C}}^l\boldsymbol{b}^l\cdot\prod_{s\neq k\neq l}\mathbf{B}^{s\top}\mathbf{C}^s\boldsymbol{b}^s\Big] & \text{, if } \; k \neq l \end{cases}.
\end{aligned}$$

Setting this to zero yields the unconstrained solution g^k_{uc},

$$\boldsymbol{b}^k_{uc} = \Big(\overbrace{\langle\boldsymbol{\phi}^k\cdot\prod_{l\neq k} g^l, \prod_{l\neq k} g^l\cdot\boldsymbol{\phi}^{k\top}\rangle_\xi + \sigma_k^2\dot{\mathbf{C}}^k}^{\text{regularized covariance matrix } \bar{\mathbf{C}}^k}\Big)^{-1}\Big(\langle\boldsymbol{\phi}^k\cdot\prod_{l\neq k} g^l, \mu - f\rangle_\xi - \sum_{l=1}^{d}\boldsymbol{R}^k_l\,\sigma_l^2\Big)\,. \quad (11)$$

14

However, these parameters do not satisfy to the constraint $\|g'^k\|_{\vartheta^k} \stackrel{!}{=} 1$, and have to be normalized:

$$\boldsymbol{b}'^k \quad := \quad \frac{\boldsymbol{b}^k_{uc}}{\|g^k_{uc}\|_{\vartheta^k}} \quad = \quad \frac{\boldsymbol{b}^k_{uc}}{\sqrt{\boldsymbol{b}^{k\top}_{uc}\mathbf{C}^k\boldsymbol{b}^k_{uc}}}\,. \tag{12}$$

The inner loop finishes when for all k the improvement[12] from g^k to g'^k drops below some very small threshold ϵ, i.e. $\tilde{\mathcal{C}}[g] - \tilde{\mathcal{C}}[g'] < \epsilon$. Using $g'^l = g^l, \forall l \neq k$, one can calculate the left hand side:

$$\begin{aligned}\tilde{\mathcal{C}}[g] - \tilde{\mathcal{C}}[g'] &= \|g\|^2_\xi - \|g'\|^2_\xi - 2\langle g - g', \mu - f\rangle_\xi \\ &\quad + \sum_{l=1}^d \sigma_l^2 \Big[\underbrace{\|\nabla_l g\|^2_\vartheta}_{\boldsymbol{b}^{l\top}\dot{\mathbf{C}}^l\boldsymbol{b}^l} - \underbrace{\|\nabla_l g'\|^2_\vartheta}_{\boldsymbol{b}'^{l\top}\dot{\mathbf{C}}^l\boldsymbol{b}'^l} - 2\underbrace{\langle \nabla_l g - \nabla_l g', \nabla_l f\rangle_\vartheta}_{(\boldsymbol{b}^k - \boldsymbol{b}'^k)^\top \boldsymbol{R}^k_l} \Big] \qquad (13) \\ &= 2\langle g - g', \mu - f\rangle_\xi + \boldsymbol{b}^{k\top}\bar{\mathbf{C}}^k\boldsymbol{b}^{k\top} - \boldsymbol{b}'^{k\top}\bar{\mathbf{C}}^k\boldsymbol{b}'^{k\top} - 2(\boldsymbol{b}^k - \boldsymbol{b}'^k)^\top \Big(\sum_{l=1}^d \boldsymbol{R}^k_l \sigma_l^2\Big)\,.\end{aligned}$$

Appendix C Proofs of the propositions

Proof of Proposition 2: The 1st order Taylor approximation of any $g, f \in L^2(\mathcal{X}, \xi\chi)$ around $\boldsymbol{z} \in \mathcal{X}$ is $f(\boldsymbol{x}) = f(\boldsymbol{z} + \boldsymbol{x} - \boldsymbol{z}) \approx f(\boldsymbol{z}) + (\boldsymbol{x} - \boldsymbol{z})^\top \boldsymbol{\nabla} f(\boldsymbol{z})$. For the Hilbert space $L^2(\mathcal{X}, \xi\chi)$ we can thus approximate:

$$\begin{aligned}\langle g, f\rangle_{\xi\chi} &= \int \xi(d\boldsymbol{z}) \int \chi(d\boldsymbol{x}|\boldsymbol{z})\, g(\boldsymbol{x})\, f(\boldsymbol{x}) \\ &\approx \int \xi(d\boldsymbol{z}) \Big(g(\boldsymbol{z})\, f(\boldsymbol{z}) \int \overbrace{\chi(d\boldsymbol{x}|\boldsymbol{z})}^{1} + g(\boldsymbol{z}) \int \overbrace{\chi(d\boldsymbol{x}|\boldsymbol{z})\,(\boldsymbol{x} - \boldsymbol{z})}^{\mathbf{0} \text{ due to (eq.5)}}{}^\top \boldsymbol{\nabla} f(\boldsymbol{z}) \\ &\quad + \int \underbrace{\chi(d\boldsymbol{x}|\boldsymbol{z})\,(\boldsymbol{x} - \boldsymbol{z})}_{\mathbf{0} \text{ due to (eq.5)}}{}^\top \boldsymbol{\nabla} g(\boldsymbol{z})\, f(\boldsymbol{z}) + \boldsymbol{\nabla} g(\boldsymbol{z})^\top \int \underbrace{\chi(d\boldsymbol{x}|\boldsymbol{z})\,(\boldsymbol{x} - \boldsymbol{z})(\boldsymbol{x} - \boldsymbol{z})^\top}_{\frac{\vartheta}{\xi}(\boldsymbol{z})\cdot\boldsymbol{\Sigma} \text{ due to (eq.5)}} \boldsymbol{\nabla} f(\boldsymbol{z}) \Big) \\ &= \langle g, f\rangle_\xi + \sum_{k=1}^d \sigma_k^2 \,\langle \nabla_k g, \nabla_k f\rangle_\vartheta\,.\end{aligned}$$

Using this twice and the zero mean assumption (Eq. 5), we can derive:

$$\begin{aligned}\inf_{g\in\mathcal{F}} \mathcal{C}[f + g|\chi, \mu] &\equiv \inf_{g\in\mathcal{F}} \iint \xi(d\boldsymbol{z})\, \chi(d\boldsymbol{x}|\boldsymbol{z}) \Big(g^2(\boldsymbol{x}) - 2\, g(\boldsymbol{x}) \,\big(\mu(\boldsymbol{z}) - f(\boldsymbol{x})\big)\Big) \\ &= \inf_{g\in\mathcal{F}} \langle g, g\rangle_{\xi\chi} + 2\langle g, f\rangle_{\xi\chi} - 2\int \xi(d\boldsymbol{z})\, \mu(\boldsymbol{z}) \int \chi(d\boldsymbol{x}|\boldsymbol{z})\, g(\boldsymbol{x}) \\ &\approx \inf_{g\in\mathcal{F}} \langle g, g\rangle_\xi - 2\langle g, \mu - f\rangle_\xi + \sum_{k=1}^d \sigma_k^2 \Big(\langle \nabla_k g, \nabla_k g\rangle_\vartheta + 2\langle \nabla_k g, \nabla_k f\rangle_\vartheta \Big) \\ &\equiv \inf_{g\in\mathcal{F}} \|g - (\mu - f)\|^2_\xi + \sum_{k=1}^d \sigma_k^2 \|\nabla_k g + \nabla_k f\|^2_\vartheta \quad = \quad \tilde{\mathcal{C}}[g]\,. \qquad \square\end{aligned}$$

[12] Anything simpler does not converge, as the parameter vectors often evolve along chaotic orbits in $\mathbb{R}^{m_k}$.

Proof of Proposition 3: The analytical solution to the optimization problem in Equation 6 is derived in Appendix B and has a unique solution if the matrix $\bar{\mathbf{C}}^k$, defined in Equation 11, is of full rank:

$$\bar{\mathbf{C}}^k \quad := \quad \langle \boldsymbol{\phi}^k \cdot \prod_{l \neq k} g^l, \prod_{l \neq k} g^l \cdot \boldsymbol{\phi}^{k\top} \rangle_\xi \quad + \quad \sigma_k^2 \dot{\mathbf{C}}^k \,.$$

For Fourier bases the matrix $\dot{\mathbf{C}}^k$ is diagonal, with $\dot{C}^k_{11}$ being the only zero entry. $\bar{\mathbf{C}}^k$ is therfore full rank if $\sigma_k^2 > 0$ and $\bar{C}^k_{11} > 0$. Because ϑ is *absolutely continuous* w.r.t. ξ, the constraint $\|g^l\|_\vartheta = 1, \forall l$, implies that there exist no g^l that is zero on *all* training samples. As the first Fourier base is a constant, $\langle \phi^k_1 \cdot \prod_{l \neq k} g^l, \prod_{l \neq k} g^l \cdot \phi^k_1 \rangle_\xi > 0$ and the matrix $\bar{\mathbf{C}}^k$ is therefore of full rank. □

Bibliography

R. E. Bellman. *Dynamic programming*. Princeton University Press, 1957.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738.

W. Böhmer and K. Obermayer. Towards structural generalization: Factored approximate planning. ICRA Workshop on Autonomous Learning, 2013. URL `http://autonomous-learning.org/wp-content/uploads/13-ALW/paper_1.pdf`.

W. Böhmer, S. Grünewälder, H. Nickisch, and K. Obermayer. Generating feature spaces for linear algorithms with regularized sparse kernel slow feature analysis. *Machine Learning*, 89(1-2):67–86, 2012.

W. Böhmer, S. Grünewälder, Y. Shen, M. Musial, and K. Obermayer. Construction of approximation spaces for reinforcement learning. *Journal of Machine Learning Research*, 14:2067–2118, July 2013.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, 1992.

P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4): 547–553, 2009.

L. Csató and M. Opper. Sparse on-line Gaussian processes. *Neural Computation*, 14 (3):641–668, 2002.

J. H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19 (1):1–67, 1991.

J. Gerritsma, R. Onnink, and A Versluis. Geometry, resistance and stability of the delft systematic yacht hull series. *Int. Shipbuilding Progress*, 28:276–297, 1981.

M. Gönen and E. Alpaydın. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12:2211–2268, July 2011. ISSN 1532-4435.

S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2nd edition, 1998. ISBN 978-0132733502.

L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

J. A. Nelder and R. W. M. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society, Series A, General*, 135:370–384, 1972.

J. Pearl. *Probabilistic reasoning in intelligent systems*. Morgan Kaufmann, 1988.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

M. E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, September 2001. ISSN 1532-4435.

P. Tüfekci. Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. *International Journal of Electrical Power & Energy Systems*, 60:126–140, 2014.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

Z. Wang, K. Crammer, and S. Vucetic. Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale svm training. *Journal of Machine Learning Research*, 13(1):3103–3131, October 2012. ISSN 1532-4435.

I-C. Yeh. Modeling of strength of high performance concrete using artificial neural networks. *Cement and Concrete Research*, 28(12):1797–1808, 1998.

Algorithm 1 (detailed) – LFF-Regression

Input: $\mathbf{X} \in \mathbb{R}^{d \times n}$, $\boldsymbol{y} \in \mathbb{R}^n$, $\boldsymbol{\sigma}^2 \in \mathbb{R}^d$ $\epsilon, \varepsilon \in \mathbb{R}$

$\mathbf{C}^k := \langle \boldsymbol{\phi}^k, \boldsymbol{\phi}^k \rangle_{\vartheta^k}$, $\dot{\mathbf{C}}^k := \langle \nabla \boldsymbol{\phi}^k, \nabla \boldsymbol{\phi}^k \rangle_{\vartheta^k}$, $\forall k$ // analytical covariance matrices

$\Phi^k_{jt} := \boldsymbol{\phi}^k_j(X_{kt})$, $\forall k$, $\forall j$, $\forall t$ // optional cache of sample-expansion

$\boldsymbol{f} := \mathbf{0} \in \mathbb{R}^n$; $\boldsymbol{a} := \emptyset$; $\mathbf{B}^k := \emptyset$, $\forall k$; $\boldsymbol{\Psi} := \infty$ // initialize empty $f \in \mathcal{F}^0$

while $\det\left(\frac{1}{n}\boldsymbol{\Psi}\boldsymbol{\Psi}^\top\right) > \varepsilon$ **do**

  $\boldsymbol{b}^k := \mathbf{1}^k \in \mathbb{R}^{m_k}$, $\forall k$; $\boldsymbol{g}^k := \mathbf{1} \in \mathbb{R}^n$, $\forall k$ // initialize all g^k as constant

  $\boldsymbol{h} := \boldsymbol{\infty} \in \mathbb{R}^d$ // initialize estimated improvement

  while $\max(\boldsymbol{h}) > \epsilon$ **do**

    for k **in** randperm$(1, \ldots, d)$ **do**

      $\boldsymbol{R}_k := \dot{\mathbf{C}}^k \mathbf{B}^k [\boldsymbol{a} \cdot \prod_{s \neq k} \mathbf{B}^{s\top} \mathbf{C}^s \boldsymbol{b}^s]$ // $\boldsymbol{R}_k = \frac{\partial}{\partial \boldsymbol{b}^k} \langle \nabla_k g, \nabla_k f \rangle_\vartheta$

      $\boldsymbol{R}_l := \mathbf{C}^k \mathbf{B}^k [\boldsymbol{a} \cdot \mathbf{B}^{l\top} \dot{\mathbf{C}}^l \boldsymbol{b}^l \cdot \prod_{s \neq k \neq l} \mathbf{B}^{s\top} \mathbf{C}^s \boldsymbol{b}^s]$, $\forall l \neq k$ // $\boldsymbol{R}_l = \frac{\partial}{\partial \boldsymbol{b}^k} \langle \nabla_l g, \nabla_l f \rangle_\vartheta$

      $\bar{\mathbf{C}} := \boldsymbol{\Phi}^k \left[\boldsymbol{\Phi}^{k\top} \cdot \prod_{l \neq k} (\boldsymbol{g}^l)^2 \, \mathbf{1}^\top \right] + \sigma_k^2 \, \dot{\mathbf{C}}^k$ // regularized cov. matrix (eq. 11)

      $\boldsymbol{b}' := \bar{\mathbf{C}}^{-1} \left(\boldsymbol{\Phi}^k \left[(\boldsymbol{y} - \boldsymbol{f}) \cdot \prod_{l \neq k} \boldsymbol{g}^l \right] - \mathbf{R}\boldsymbol{\sigma}^2 \right)$ // unconstrained g^k_{uc} (eq. 11)

      $\boldsymbol{b}' := \boldsymbol{b}' \,/\, \sqrt{\boldsymbol{b}'^\top \mathbf{C}^k \boldsymbol{b}'}$ // enforce constraints (eq. 12)

      $h_k := \frac{2}{n} (\boldsymbol{b}^k - \boldsymbol{b}')^\top \left(\boldsymbol{\Phi}^k \left[(\boldsymbol{y} - \boldsymbol{f}) \cdot \prod_{l \neq k} \boldsymbol{g}^l \right] \right)$ // approximate $2\langle g - g', \mu - f \rangle_\xi$

      $h_k := h_k + \boldsymbol{b}^k \bar{\mathbf{C}} \boldsymbol{b}^k - \boldsymbol{b}' \bar{\mathbf{C}} \boldsymbol{b}' - 2(\boldsymbol{b}^k - \boldsymbol{b}')^\top \mathbf{R}\boldsymbol{\sigma}^2$ // cost improvement (eq. 13)

      $\boldsymbol{b}^k := \boldsymbol{b}'$; $\boldsymbol{g}^k := \boldsymbol{\Phi}^{k\top} \boldsymbol{b}^k$ // update factor function g^k

    end for // end function g^k update

  end while // end inner loop: cost function converged and thus g optimized

  $\mathbf{B}^k := [\mathbf{B}^k, \boldsymbol{b}^k]$, $\forall k$; $\boldsymbol{\Psi} := \left[\prod_{k=1}^{d} \mathbf{B}^{k\top} \boldsymbol{\Phi}^k \right]$ // adding g to the bases functions of f

  $\boldsymbol{a} := \left(\boldsymbol{\Psi}\boldsymbol{\Psi}^\top\right)^{-1} \boldsymbol{\Psi}\boldsymbol{y}$; $\boldsymbol{f} := \boldsymbol{\Psi}^\top \boldsymbol{a}$ // project μ onto new bases

end while // end outer loop: new g no longer linear independent, thus $f \approx \mu$

Output: $\boldsymbol{a} \in \mathbb{R}^m$, $\{\mathbf{B}^k \in \mathbb{R}^{m_k \times m}\}_{k=1}^d$ // return parameters of $f \in \mathcal{F}^m$

Appendix C

Proofs and Lemmas

C.1 Technical lemmas

Lemma C.1 *The supremum-operator norm of any transition operator* $\hat{\mathrm{P}}$ *from* $L^\infty(\mathcal{Y})$ *to* $L^\infty(\mathcal{X})$*, i.e.* $P : \mathcal{X} \times \mathcal{B}(\mathcal{Y}) \to [0,1]$*,* $\int_{\mathcal{Y}} P(dy|x) = 1, \forall x \in \mathcal{X}$*, is* $\|\hat{\mathrm{P}}\|_\infty = 1$*.*

Proof: In the following we use $\|\cdot\|_\infty^{\mathcal{X}}$ and $\|\cdot\|_\infty^{\mathcal{Y}}$ to indicate the supremum norm over $L^\infty(\mathcal{X})$ and $L^\infty(\mathcal{Y})$, respectively. The operator norm is defined as

$$\|\hat{\mathrm{P}}\|_\infty = \sup_{f \in L^\infty(\mathcal{Y})} \{\|\hat{\mathrm{P}}[f]\|_\infty^{\mathcal{X}} | \, \|f\|_\infty^{\mathcal{Y}} = 1\} .$$

The involved function norm $\|f\|_\infty^{\mathcal{Y}} = 1$ implies that $|f(y)| \leq 1, \forall y \in \mathcal{Y}$, and therefore $|\hat{\mathrm{P}}[f](x)| = |\int P(dy|x)\, f(y)| \leq \int P(dy|x)\, |f(y)| \leq \int P(dy|x) = 1, \forall x \in \mathcal{X}$. The inequality becomes an equality for $f(y) = 1, \forall y \in \mathcal{Y}$, and thus $\|\hat{\mathrm{P}}\|_\infty = 1$. □

Lemma C.2 *The operator norm of transition operator* $\hat{\mathrm{P}}$ *is* $\|\hat{\mathrm{P}}\|_{\xi\pi} = 1$*, when its conditional measure* $P : \mathcal{X} \times \mathcal{A} \times \mathcal{B}(\mathcal{X}) \to [0,1]$*,* $\int_{\mathcal{X}} P(dy|x,a) = 1, \forall x \in \mathcal{X}, \forall a \in \mathcal{A}$*, is ergodic with policy* π*, yielding steady state distribution* ξ*.*

Proof: The operator norm of $\hat{\mathrm{P}}$ is defined as

$$\|\hat{\mathrm{P}}\|_{\xi\pi} = \sup_{f \in L^2(\mathcal{X},\xi)} \Big\{ \|\hat{\mathrm{P}}[f]\|_{\xi\pi} \,\big|\, \|f\|_\xi = 1 \Big\} .$$

Using $\big(\int P(dy|x,a)\, f(y)\big)^2 \leq \int P(dy|x,a)\, f^2(y), \forall x \in \mathcal{X}, \forall a \in \mathcal{A}, \forall f \in L^2(\mathcal{X},\xi)$ (called *Jensens inequality*, see e.g. Boyd and Vandenberghe, 2004), we first show:

$$\begin{aligned} \|\hat{\mathrm{P}}[f]\|_{\xi\pi}^2 &= \int \xi(dx)\, \pi(da|x) \Big(\int P(dy|x,a)\, f(y) \Big)^2 \\ &\leq \int \underbrace{\iint \xi(dx)\, \pi(da|x) P(dy|x,a)}_{\xi(dy) \text{ due to ergodicity}} \, f^2(y) \quad = \quad \|f\|_\xi^2 \quad = \quad 1 . \end{aligned}$$

Now we prove that $\|\hat{\mathrm{P}}[f]\|_{\xi\pi} = 1$ at the example of function $f(y) = 1, \forall y \in \mathcal{X}$:

$$\|\hat{\mathrm{P}}[f]\|_{\xi\pi} \quad = \quad \iint \xi(dx)\, \pi(da|x) \Big(\underbrace{\int P(dy|x,a)\, \overbrace{f(y)}^{1}}_{1} \Big)^2 \quad = \quad 1 \quad = \quad \|f\|_\xi . \qquad \square$$

Lemma C.3 (similar to Lemma 4.3 in Williams and Baird, 1993)
For any $f, g \in L^\infty(\mathcal{X})$ *holds:* $\sup_{x\in\mathcal{X}} \big(f(x) - g(x)\big) \geq \sup_{x\in\mathcal{X}} f(x) - \sup_{y\in\mathcal{X}} g(y)$.

Proof: $\sup_{x\in\mathcal{X}} \big(f(x) - g(x)\big) \geq \sup_{x\in\mathcal{X}} \big(f(x) - \sup_{y\in\mathcal{X}} g(y)\big) = \sup_{x\in\mathcal{X}} f(x) - \sup_{y\in\mathcal{X}} g(y)\,.$ □

C.2 Proofs from Chapter 2

Proposition 2.1 (repeated from Page 12) (see e.g. Bertsekas, 2007) *The value iteration operator* $\hat{\Gamma}_\pi[\hat{\mathrm{B}}[v]] = \hat{\Gamma}_\pi[r] + \gamma\hat{\Gamma}_\pi[\hat{\mathrm{P}}^\pi[v]]$, $\forall v \in L^\infty(\mathcal{X})$, *is for* $\gamma < 1$ *a contraction mapping under the supremum norm.*

Proof: $\|\hat{\Gamma}_\pi[\hat{\mathrm{B}}[v]] - \hat{\Gamma}_\pi[\hat{\mathrm{B}}[v']]\|_\infty = \gamma\|\hat{\Gamma}_\pi[\hat{\mathrm{P}}[v-v']]\|_\infty \leq \gamma\|\hat{\Gamma}_\pi[\hat{\mathrm{P}}]\|_\infty\|v-v'\|_\infty = \gamma\|v-v'\|_\infty < \|v-v'\|_\infty$, where the last equation used Lemma C.1, as $\hat{\Gamma}_\pi[\hat{\mathrm{P}}]$ is a transition operator, which can be seen by $\iint \pi(da|x)P(dy|x,a) = 1$. □

Proposition 2.2 (repeated from Page 12) (see e.g. Bertsekas, 2007) *The fix point of value iteration operator* $\hat{\Gamma}_\pi[\hat{\mathrm{B}}]$ *in* $L^\infty(\mathcal{X})$ *is*

$$v^\pi \quad = \quad \hat{\Gamma}_\pi[\hat{\mathrm{B}}[v^\pi]] \quad = \quad \Big(\hat{\mathrm{I}} - \gamma\hat{\Gamma}_\pi[\hat{\mathrm{P}}]\Big)^{-1}[\hat{\Gamma}_\pi[r]] \quad \in \quad L^\infty(\mathcal{X})\,,$$

where $(\hat{\mathrm{A}})^{-1}$ *denotes the inverse operator of* $\hat{\mathrm{A}} : L^\infty(\mathcal{X}) \to L^\infty(\mathcal{X})$ *with* $\hat{\mathrm{A}}[(\hat{\mathrm{A}})^{-1}] = (\hat{\mathrm{A}})^{-1}[\hat{\mathrm{A}}] = \hat{\mathrm{I}}$, *and* $\hat{\mathrm{I}}$ *denotes the identity operator* $\hat{\mathrm{I}}[f] = f, \forall f \in L^\infty(\mathcal{X})$.

Proof: Let $(\hat{\mathrm{A}})^t$ denote t applications of operator $\hat{\mathrm{A}} : L^\infty(\mathcal{X}) \to L^\infty(\mathcal{X})$. Unfolding the recursive definition of the fix point yields the infinite sum

$$\hat{\Gamma}_\pi[\hat{\mathrm{B}}[v^\pi]] \quad = \quad \hat{\Gamma}_\pi[r] + \gamma\hat{\Gamma}_\pi[\hat{\mathrm{P}}[v^\pi]] \quad = \quad \lim_{n\to\infty} \sum_{t=0}^{n-1} \gamma^t \Big(\hat{\Gamma}_\pi[\hat{\mathrm{P}}]\Big)^t [\hat{\Gamma}_\pi[r]]\,.$$

For any invertible linear operator $\hat{\mathrm{A}}$ holds $\hat{\mathrm{A}}^{-1}[\hat{\mathrm{A}}[f]] = f, \forall f \in L^\infty(\mathcal{X})$, and we show this identity with $\|\hat{\mathrm{A}}^{-1}[\hat{\mathrm{A}}[f]] - f\|_\infty = 0, \forall f \in L^\infty(\mathcal{X})$:

$$\begin{aligned}\lim_{n\to\infty} \Big\| \Big(\hat{\mathrm{I}} - \gamma\hat{\mathrm{P}}^\pi\Big)^{-1}[(\hat{\mathrm{I}} - \gamma\hat{\mathrm{P}}^\pi)[f]] - f\Big\|_\infty \quad &= \quad \lim_{n\to\infty} \Big\| \sum_{t=0}^{n-1} \gamma^t (\hat{\mathrm{P}}^\pi)^t [(\hat{\mathrm{I}} - \gamma\hat{\mathrm{P}}^\pi)[f]] - f\Big\|_\infty \\ = \quad \lim_{n\to\infty} \gamma^n \Big\|(\hat{\mathrm{P}}^\pi)^n[f]\Big\|_\infty \quad &\leq \quad \lim_{n\to\infty} \gamma^n \|f\|_\infty \quad = \quad 0\,.\end{aligned}$$

The inequality is based on Lemma C.1 ($\|\hat{\mathrm{P}}^\pi\|_\infty = 1$) and the last equality on the fact that all $f \in L^\infty(\mathcal{X})$ are bounded ($\|f\|_\infty < \infty$) and $\lim_{n\to\infty} \gamma^n = 0$. □

Lemma 2.3 (repeated from Page 14) *For the bias-free online learning rate* $\alpha_t = \frac{\alpha}{1-(1-\alpha)^t}$ *holds* $\lim_{\alpha\to 0} \alpha_t = \frac{1}{t}$.

Proof: We will use the *binomial identity* $(x+y)^t = \sum_{k=0}^{t} \binom{t}{k} x^k y^{t-k}$.

$$\begin{aligned}
\lim_{\alpha\to 0} \alpha_t &= \lim_{\alpha\to 0} \tfrac{\alpha}{1-(1-\alpha)^t} = \lim_{\alpha\to 0} \left[\alpha^{-1} - (\alpha^{-\frac{1}{t}} - \alpha^{\frac{t-1}{t}})^t\right]^{-1} \\
&= \lim_{\alpha\to 0} \left[\alpha^{-1} - \sum_{k=0}^{t} \binom{t}{k} (-1)^{t-k} \alpha^{-\frac{k}{t}} \alpha^{\frac{(t-k)(t-1)}{t}}\right]^{-1} \\
&= \lim_{\alpha\to 0} \left[\alpha^{-1} - \sum_{k=0}^{t-2} \binom{t}{k} (-1)^{t-k} \alpha^{t-k-1} + \binom{t}{t-1} \alpha^0 - \binom{t}{t} \alpha^{-1}\right]^{-1} \\
&= \lim_{\alpha\to 0} \left[t - \sum_{k=0}^{t-2} \binom{t}{k} (-1)^{t-k} \alpha^{t-k-1}\right]^{-1} = \frac{1}{t}\,. \qquad \square
\end{aligned}$$

Proposition 2.4 (repeated from Page 14) *Online learning, with initial value f_0 and adaptable learning rate $\alpha_t = \frac{1}{t+\lambda}$, $\lambda \in [0,\infty)$, is equivalent to calculating the regularized empirical mean over the training set $\{y_t\}_{t=1}^n$, that is,*

$$f_n = f_{n-1} + \alpha_n(y_n - f_{n-1}) = \tfrac{1}{n+\lambda} \sum_{t=1}^{n} y_t + \tfrac{\lambda}{n+\lambda} f_0\,.$$

Proof: By expanding the recursive definition of f_n, one can write

$$f_n = \sum_{t=0}^{n-1} y_{n-t}\, \alpha_{n-t} \prod_{s=0}^{t-1} (1-\alpha_{n-s}) + f_0 \prod_{s=0}^{n-1} (1-\alpha_{n-s})\,.$$

Using the induction step

$$(1-\alpha_{n-s})\alpha_{n-s-1} = \tfrac{1}{n-s-1+\lambda} - \tfrac{1}{n-s+\lambda}\tfrac{1}{n-s-1+\lambda} = \tfrac{n-s-1+\lambda}{(n-s-1+\lambda)(n-s+\lambda)} = \alpha_{n-s}\,.$$

one can show that for all $t \in \{0,\ldots,n\}$ that $\alpha_{n-t} \prod_{s=0}^{t-1}(1-\alpha_{n-s}) = \alpha_n$. Therefore

$$f_n = \alpha_n \sum_{t=1}^{n} y_n + \tfrac{\alpha_n}{\alpha_0} f_0 = \tfrac{1}{n+\lambda} y_n + \tfrac{\lambda}{n+\lambda} f_0\,. \qquad \square$$

Proposition 2.5 (repeated from Page 15) *Given a converged value function $v^i \in L^\infty(\mathcal{X})$, $\|v^i - v^{i-1}\|_\infty \le \epsilon$, batch learning (Equation 2.11) converges in at most k value iterations after observing a new transition with TD-error $TD_n := r_n + \gamma v^i(x_{n+1}) - v^i(x_n)$:*

$$k \le 1 - \log_\gamma \left(\gamma + \tfrac{1}{\epsilon} \tfrac{|TD_n|}{\eta_n(x_n)}\right).$$

Proof: Using Proposition 2.4, one can write the estimate in Equation 2.11 using the Kronecker delta function $\delta_t := \delta_{(x_t - x)}$, i.e. $v^i(x) = \frac{1}{\eta_{n-1}(x)} \sum_{t=1}^{n-1} \delta_t (r_t + \gamma v^{i-1}(x_{t+1}))$ and $\eta_{n-1}(x) = \sum_{t=1}^{n-1} \delta_t$. Note that the next iteration's value function v^{i+1} depends on one sample more, i.e. with the abbreviation $b_t^i := r_t - \gamma v^i(x_{t+1})$:

$$\begin{aligned}
\|v^{i+1} - v^i\|_\infty &= \sup_{x\in\mathcal{X}} \frac{\left|\eta_{n-1}(x) \sum_{t=1}^{n} \delta_t b_t^i - (\eta_{n-1}(x) + \delta_n) \sum_{t=1}^{n-1} \delta_t b_t^{i-1}\right|}{\eta_n(x)\,\eta_{n-1}(x)} \\
&= \sup_{x\in\mathcal{X}} \frac{\left|\eta_{n-1}(x) \sum_{t=1}^{n-1} \delta_t (b_t^i - b_t^{i-1}) + \eta_{n-1}(x)\delta_n b_n^i - \delta_n \sum_{t=1}^{n-1} \delta_t b_t^{i-1}\right|}{\eta_n(x)\,\eta_{n-1}(x)} \\
&\le \gamma \sup_{x\in\mathcal{X}} \frac{\left|\sum_{t=1}^{n-1} \delta_t (\overbrace{v^i(x_{t+1}) - v^{i-1}(x_{t+1})}^{\le \epsilon})\right|}{\eta_n(x)} + \frac{r_n + \gamma v^i(x_{n+1}) - v^i(x_n)}{\eta_n(x_n)} \\
&\le \gamma\,\epsilon + \frac{r_n + \gamma v^i(x_{n+1}) - v^i(x_n)}{\eta_n(x_n)} = \gamma\,\epsilon + \tfrac{|TD_n|}{\eta_n(x_n)}\,.
\end{aligned}$$

After $k-1$ additional value iteration steps, this error will be $\|v^{i+k} - v^{i+k-1}\|_\infty \leq \gamma^{k-1}\|v^{i+1} - v^i\|_\infty$, which follows recursively from:

$$\begin{aligned}\|v^{i+2} - v^{i+1}\|_\infty &= \sup_{x\in\mathcal{X}} \left|\tfrac{1}{\eta_n(x)}\sum_{t=1}^{n}\delta_t\gamma(v^{i+1}(x_{t+1}) - v^i(x_{t+1}))\right| \\ &\leq \sup_{x\in\mathcal{X}} \tfrac{1}{\eta_n(x)}\sum_{t=1}^{n}\delta_t\gamma \underbrace{\left|v^{i+1}(x_{t+1}) - v^i(x_{t+1})\right|}_{\leq \|v^{i+1}-v^i\|_\infty} \\ &\leq \gamma\,\|v^{i+1} - v^i\|_\infty\,.\end{aligned}$$

After k steps, value iteration is converged, and we have

$$\gamma^{k-1}\left(\gamma\,\epsilon + \tfrac{|TD_n|}{\eta_n(x_n)}\right) \overset{!}{\leq} \epsilon \quad \Rightarrow \quad k \leq 1 - \log_\gamma\left(\gamma + \tfrac{1}{\epsilon}\tfrac{|TD_n|}{\eta_n(x_n)}\right).$$

□

Proposition 2.6 (repeated from Page 16) (see e.g. Watkins and Dayan, 1992) *The Q-learning operator* $\hat{\mathrm{B}}[\hat{\Gamma}_*[q]] = r + \gamma\hat{\mathrm{P}}[\hat{\Gamma}_*[q]]$, $\forall q \in L^\infty(\mathcal{X}\times\mathcal{A})$, *is for* $\gamma < 1$ *a contraction mapping under the supremum norm.*

Proof:

$$\begin{aligned}\left\|\hat{\mathrm{B}}[\hat{\Gamma}_*[q]] - \hat{\mathrm{B}}[\hat{\Gamma}_*[q']]\right\|_\infty &\leq \gamma\,\|\hat{\mathrm{P}}\|_\infty\|\hat{\Gamma}_*[q] - \hat{\Gamma}_*[q']\|_\infty \\ \text{(Lemma C.1)} \qquad &= \gamma \sup_{x\in\mathcal{X}}\left|\sup_{a\in\mathcal{A}} q(x,a) - \sup_{b\in\mathcal{A}} q'(x,b)\right| \\ \text{(Lemma C.3)} \qquad &\leq \gamma \sup_{x\in\mathcal{X}}\left|\sup_{a\in\mathcal{A}}\left(q(x,a) - q'(x,a)\right)\right| \\ &\leq \gamma \sup_{x\in\mathcal{X}}\left|\sup_{a\in\mathcal{A}}|q(x,a) - q'(x,a)|\right| \\ &= \gamma\,\|q - q'\|_\infty \quad < \quad \|q - q'\|_\infty \qquad \square\end{aligned}$$

C.3 Proofs from Chapter 3

Lemma 3.1 (repeated from Page 25) (see e.g. Schölkopf and Smola, 2001) *RKHS* $\mathcal{H}_\kappa$ *induced by positive-semi-definite kernel* $\kappa : L^\infty(\mathcal{X}\times\mathcal{X})$ *(i) contains all functions* $k_x(y) := \kappa(x,y), \forall x,y \in \mathcal{X}$, *and (ii) has the "reproducing property"* $\langle f, k_x\rangle_{\mathcal{H}_\kappa} = f(x)$, $\forall x \in \mathcal{X}$, $\forall f \in \mathcal{H}_\kappa$.

Proof: (i) By the definition of RKHS in Equation 3.7 on Page 25, a parameterized kernel function $k_x \in \mathcal{H}_\kappa$ if $\|k_x\|^2_{\mathcal{H}_\kappa} < \infty$. Here I use $\langle k_x, \varphi_i\rangle_\xi = \hat{\mathrm{K}}[\varphi_i](x) = \lambda_i\varphi_i(x)$:

$$\langle k_x, k_x\rangle_{\mathcal{H}_\kappa} \quad = \quad \sum_{i=0}^{\infty}\lambda_i^{-1}\langle k_x, \varphi_i\rangle_\xi^2 \quad = \quad \sum_{i=0}^{\infty}\lambda_i\varphi_i^2(x) \quad = \quad \kappa(x,x)\,.$$

As $\kappa \in L^\infty(\mathcal{X}\times\mathcal{X})$, the kernel is point-wise bounded $\kappa(x,x) < \infty, \forall x \in \mathcal{X}$.
(ii) The same property of eigenfunctions ensures the reproducing property:

$$\langle f, k_x\rangle_{\mathcal{H}_\kappa} \quad = \quad \sum_{i=0}^{\infty}\langle f, \varphi_i\rangle_\xi\,\lambda_i^{-1}\,\langle k_x, \varphi_i\rangle_\xi \quad = \quad \sum_{i=0}^{\infty}\langle f, \varphi_i\rangle_\xi\varphi_i(x) \quad = \quad f(x)\,. \qquad \square$$

Theorem 3.2 (repeated from Page 26) (from Schölkopf and Smola, 2001) *Let $\kappa \in L^\infty(\mathcal{X} \times \mathcal{X})$ be a positive-definite kernel on the non-empty set $\mathcal{X}$ with corresponding RKHS $\mathcal{H}_\kappa$, and let $\hat{\mathrm{C}}$ be any empirical cost based only on training set $\{x_t, y_t\}_{t=1}^n \subset \mathcal{X} \times \mathbb{R}$ with a strictly monotonically increasing real-valued regularization $g : [0, \infty) \to \mathbb{R}$. Any solution $f^* \in \mathcal{H}_\kappa$ can be represented as linear combination of parameterized kernel functions,*

$$f^* \in \operatorname*{arg\,inf}_{f \in \mathcal{H}_\kappa} \hat{\mathrm{C}}[f](\{x_t, y_t\}_{t=1}^n) + g(\|f\|_{\mathcal{H}_\kappa}) \quad \Rightarrow \quad f^*(\cdot) = \sum_{t=1}^n a_t\, \kappa(\cdot, x_t)\,, \;\; \exists \boldsymbol{a} \in \mathbb{R}^n\,.$$

Proof: Assume there exists a solution $f^*(\cdot) = \sum_{t=1}^n a_i\, \kappa(\cdot, x_t) + v(\cdot)$, where $v \in \mathcal{H}_\kappa$ can not be expressed as linear combination of parameterized kernel functions $k_{x_t}(\cdot) := \kappa(\cdot, x_t) \in \mathcal{H}_\kappa$, i.e. v is orthogonal to the subspace and thus $\langle v, k_{x_t}\rangle_{\mathcal{H}_\kappa} = 0, \forall t \in \{1, \ldots, n\}$. Due to the *reproducing property* (Lemma 3.1) one can write

$$f(x_t) \quad = \quad \langle f, k_{x_t}\rangle_{\mathcal{H}_\kappa} \quad = \quad \sum_{i=1}^n a_i \underbrace{\langle k_{x_i}, k_{x_t}\rangle_{\mathcal{H}_\kappa}}_{\kappa(x_i, x_t)} + \underbrace{\langle v, k_{x_t}\rangle_{\mathcal{H}_\kappa}}_{0}\,,$$

which implies that any cost function depending only on training samples is not influenced by the orthogonal function v. Furthermore, $\|f^*\|_{\mathcal{H}_\kappa}^2 = \|\sum_{i=1}^n a_i k_{x_t}\|_{\mathcal{H}_\kappa}^2 + \|v\|_{\mathcal{H}_\kappa}^2 \geq \|\sum_{i=1}^n a_i k_{x_t}\|_{\mathcal{H}_\kappa}^2$ and the regularization term $g(\|f\|_{\mathcal{H}_\kappa})$ therefore forces v in any solution to zero, due the strictly monotonically increase of g. □

Lemma 3.3 (repeated from Page 27) *Let $\{r_t\}_{t=1}^n$ be i.i.d. normal distributed with means $\{r(z_t)\}_{t=1}^n$ and variance σ^2. For the least squares operator $\hat{\mathrm{L}}[f] := \frac{1}{2n}\sum_{t=1}^n \big(r_t - f(z_t)\big)^2$ holds:*

$$\mathbb{E}\big[\hat{\mathrm{L}}[f]\big] = \tfrac{1}{2}\|r - f\|_\zeta^2 + \tfrac{\sigma^2}{2} \qquad \text{and} \qquad \mathbb{E}\Big[\big(\hat{\mathrm{L}}[f] - \mathbb{E}[\hat{\mathrm{L}}[f]]\big)^2\Big] = \frac{\sigma^2}{n}\Big(\|r - f\|_\zeta^2 + \tfrac{\sigma^2}{2}\Big)\,.$$

Proof: Let each label r_t be the sum of the mean and a normal distributed random variable ϵ_t, i.e. $r_t = r(z_t) + \epsilon_t$. This variable has zero mean $\mathbb{E}[\epsilon_i] = 0$, co-variance $\mathbb{E}[\epsilon_i \epsilon_j] = \sigma^2\, \delta_{ij}$, 3rd moment $\mathbb{E}[\epsilon_i \epsilon_j \epsilon_k] = 0$ and 4th moment $\mathbb{E}[\epsilon_i \epsilon_j \epsilon_k \epsilon_l] = 3\sigma^4\, \delta_{ijkl}$.

$$\begin{aligned}
\mathbb{E}\big[\hat{\mathrm{L}}[f]\big] &= \tfrac{1}{2n}\sum_{t=1}^n \mathbb{E}\Big[\big(r(z_t) - f(z_t)\big)^2 + 2\epsilon_t\big(r(z_t) - f(z_t)\big) + \epsilon_t^2\Big] \\
&= \tfrac{1}{2}\big(\|r - f\|_\zeta^2 + \sigma^2\big) \\
\mathbb{E}\big[\hat{\mathrm{L}}[f]^2\big] &= \tfrac{1}{4}\|r - f\|_\zeta^4 + \tfrac{n+2}{2n}\sigma^2\|r - f\|_\zeta^2 + \tfrac{n+2}{4n}\sigma^4 \\
\mathbb{E}\big[\hat{\mathrm{L}}[f]\big]^2 &= \tfrac{1}{4}\|r - f\|_\zeta^4 + \tfrac{1}{2}\sigma^2\|r - f\|_\zeta^2 + \tfrac{1}{4}\sigma^4
\end{aligned}$$

For $\mathbb{E}\big[\hat{\mathrm{L}}[f]^2\big]$ I used the above moments, in particular: $\frac{1}{n^2}\sum_{ij}\mathbb{E}[\epsilon_i\epsilon_j] = \frac{\sigma^2}{n}$ and $\frac{1}{4n^2}\sum_{ij}\mathbb{E}[\epsilon_i^2\epsilon_j^2] = \frac{1}{4n^2}\big(\sum_{i=j}\mathbb{E}[\epsilon_i^4] + \sum_{i\neq j}\mathbb{E}[\epsilon_i^2]\mathbb{E}[\epsilon_j^2]\big) = \frac{3}{4n}\sigma^4 + \frac{n-1}{4n}\sigma^4 = \frac{n+2}{4n}\sigma^4$. □

Lemma 3.4 (repeated from Page 29) *For any set of orthonormal basis functions* $\{\phi_i\}_{i=1}^p \subset L^2(\mathcal{Z},\zeta)$ *holds*

$$|f(x)-f(y)| \quad \leq \quad \|\boldsymbol{\phi}(x)-\boldsymbol{\phi}(y)\|_2 \; \|f\|_\zeta\,, \qquad \forall x,y\in\mathcal{Z}\,, \quad \forall f\in\mathcal{F}_\phi\,,$$

where $\mathcal{F}_\phi := \{\boldsymbol{a}^\top\boldsymbol{\phi}\}$ *denotes the space of linear combinations of basis functions.*

Proof: Using the *Cauchy-Schwarz inequality* for the inner product space $\mathbb{R}^p$, i.e. $|\boldsymbol{u}^\top\boldsymbol{v}| \leq \|\boldsymbol{u}\|_2\,\|\boldsymbol{v}\|_2$, and $\langle\boldsymbol{\phi},\boldsymbol{\phi}^\top\rangle_\zeta = \mathbf{I}$, one can show:

$$\begin{aligned} |f(x)-f(y)| &= |\boldsymbol{a}^\top(\boldsymbol{\phi}(x)-\boldsymbol{\phi}(y))| \quad \leq \quad \|\boldsymbol{\phi}(x)-\boldsymbol{\phi}(y)\|_2\,\|\boldsymbol{a}\|_2 \\ &= \|\boldsymbol{\phi}(x)-\boldsymbol{\phi}(y)\|_2\,\sqrt{\boldsymbol{a}^\top\langle\boldsymbol{\phi},\boldsymbol{\phi}^\top\rangle_\zeta\boldsymbol{a}} \quad = \quad \|\boldsymbol{\phi}(x)-\boldsymbol{\phi}(y)\|_2\,\|f\|_\zeta \quad \square \end{aligned}$$

Lemma 3.5 (repeated from Page 29) *Let* $d_\kappa(x,y) = \sqrt{\kappa(x,x)-2\kappa(x,y)+\kappa(y,y)}$, $\forall x,y\in\mathcal{Z}$, *denote the (semi-)distance induced by positive (semi-)definite kernel* $\kappa\in L^\infty(\mathcal{Z}\times\mathcal{Z})$, *then*

$$|f(x)-f(y)| \quad \leq \quad d_\kappa(x,y)\;\|f\|_{\mathcal{H}_\kappa}\,, \qquad \forall x,y\in\mathcal{Z}\,, \quad \forall f\in\mathcal{H}_\kappa\,.$$

Proof: Let $\lambda_i\in\mathbb{R}$ and $\varphi_i\in L^2(\mathcal{Z},\zeta)$ denote the eigenvalues and orthonormal eigenfunctions of kernel κ. Using the *Cauchy-Schwarz inequality* on the representation $f(z)=\sum_{i=0}^\infty\langle f,\varphi_i\rangle_\zeta\,\varphi_i(z), \forall z\in\mathcal{Z}$, one can show:

$$\begin{aligned} |f(x)-f(y)| &= \Big|\sum_{i=0}^\infty\langle f,\varphi_i\rangle_\zeta\,\lambda_i^{-\frac{1}{2}}\,\lambda_i^{\frac{1}{2}}\,(\varphi_i(x)-\varphi_i(y))\Big| \\ &\leq \underbrace{\sqrt{\textstyle\sum_{i=0}^\infty\lambda_i^{-1}\langle f,\varphi_i\rangle_\zeta^2}}_{\|f\|_{\mathcal{H}_\kappa}}\;\underbrace{\sqrt{\textstyle\sum_{i=0}^\infty\lambda_i(\varphi_i(x)-\varphi_i(y))^2}}_{\sqrt{\kappa(x,x)-2\kappa(x,y)+\kappa(y,y)}}\,. \end{aligned}$$

$\square$

Proposition 3.6 (repeated from Page 35) (see e.g. Bertsekas, 2007) *Both projected (Q-)value iteration operators* $\hat{\Pi}_\xi^\phi[\hat{\Gamma}_\pi[\hat{\mathrm{B}}[\cdot]]]$ *and* $\hat{\Pi}_{\xi\pi}^\phi[\hat{\mathrm{B}}[\hat{\Gamma}_\pi[\cdot]]]$ *are for* $\gamma<1$ *contraction mappings under the norms of* $L^2(\mathcal{X},\xi)$ *and* $L^2(\mathcal{X}\times\mathcal{A},\xi\pi)$, *respectively.*

Proof: First I show that the projection operator is a *non-expansion* under the considered norm in $L^2(\mathcal{Z},\zeta)$. Note that due to *Pythagoras theorem*, for two orthogonal functions, like $u := \hat{\Pi}_\zeta^\phi[f]$ and $v := (\hat{\mathrm{I}}-\hat{\Pi}_\zeta^\phi)[f]$, holds $\|u+v\|_\zeta^2 = \|u\|_\zeta^2+\|v\|_\zeta^2$.

$$\|\hat{\Pi}_\zeta^\phi[f]-\hat{\Pi}_\zeta^\phi[g]\|_\zeta^2 \quad \leq \quad \|\hat{\Pi}_\zeta^\phi[f-g]\|_\zeta^2 \;+\; \|(\hat{\mathrm{I}}-\hat{\Pi}_\zeta^\phi)[f-g]\|_\zeta^2 \quad = \quad \|f-g\|_\zeta^2\,.$$

Using (i) the non-expansion property, (ii) Lemma C.2 and (iii) Jensen's inequality $\int\xi(dx)(\int\pi(da|x)\,f(x,a))^2 \leq \iint\xi(dx)\,\pi(da|x)\,f^2(x,a)$, one can show

$$\begin{aligned} \Big\|\hat{\Pi}_\xi^\phi\Big[\hat{\Gamma}_\pi\big[\hat{\mathrm{B}}[f]\big]\Big]-\hat{\Pi}_\xi^\phi\Big[\hat{\Gamma}_\pi\big[\hat{\mathrm{B}}[g]\big]\Big]\Big\|_\xi^2 &\overset{\text{(i)}}{\leq} \gamma^2\Big\|\hat{\Gamma}_\pi\big[\hat{\mathrm{P}}[f-g]\big]\Big\|_\xi^2 \overset{\text{(iii)}}{\leq} \gamma^2\Big\|\hat{\mathrm{P}}[f-g]\Big\|_{\xi\pi}^2 \\ &\overset{\text{(ii)}}{\leq} \gamma^2\|f-g\|_\xi^2 \overset{(\gamma<1)}{<} \|f-g\|_\xi^2\,, \end{aligned}$$

and the equivalent for the Q-value iteration operator,

$$\left\|\hat{\Pi}^{\phi}_{\xi\pi}\left[\hat{\mathrm{B}}[\hat{\Gamma}_{\pi}[f]]\right]-\hat{\Pi}^{\phi}_{\xi\pi}\left[\hat{\mathrm{B}}[\hat{\Gamma}_{\pi}[g]]\right]\right\|^2_{\xi\pi} \overset{\text{(i)}}{\leq} \gamma^2\left\|\hat{\mathrm{P}}[\hat{\Gamma}_{\pi}[f-g]]\right\|^2_{\xi\pi} \overset{\text{(ii)}}{\leq} \gamma^2\left\|\hat{\Gamma}_{\pi}[f-g]\right\|^2_{\xi} \overset{\text{(iii)}}{\leq} \gamma^2\|f-g\|^2_{\xi\pi} \overset{(\gamma<1)}{<} \|f-g\|^2_{\xi\pi}\,. \qquad \square$$

Proposition 3.7 (repeated from Page 35) (from Tsitsiklis and Van Roy, 1997) *The difference between the LSTD fix-point solution $f \in \mathcal{F}_{\phi}$ and the true fix point $v = \hat{\Gamma}_{\pi}[\hat{\mathrm{B}}[v]] \in L^2(\mathcal{X},\xi)$ or $v = \hat{\mathrm{B}}[\hat{\Gamma}_{\pi}[v]] \in L^2(\mathcal{X}\times\mathcal{A},\xi\pi)$, with $\hat{\mathrm{B}}[\cdot] := r + \gamma\hat{\mathrm{P}}[\cdot]$ and ξ as the steady state distribution of P with policy π, is bounded from above by*

$$\left\|v - f^*\right\|_{\zeta} \leq \frac{1}{\sqrt{1-\gamma^2}}\left\|v - \hat{\Pi}^{\phi}_{\zeta}[v]\right\|_{\zeta}, \qquad \textit{with} \quad \zeta = \xi \quad \textit{or} \quad \zeta = \xi\pi\,.$$

Proof: I will use the linear operator $\hat{\mathrm{T}} : L^2(\mathcal{Z},\zeta) \to L^2(\mathcal{Z},\zeta)$ as place-holder to prove the proposition for both $\hat{\Gamma}_{\pi}[\hat{\mathrm{B}}]$ and $\hat{\mathrm{B}}[\hat{\Gamma}_{\pi}]$. Note that from the proof of Proposition 3.6 follows that $\|\hat{\Pi}^{\phi}_{\zeta}[\hat{\mathrm{T}}[v]] - \hat{\Pi}^{\phi}_{\zeta}[\hat{\mathrm{T}}[f^*]]\|_{\zeta} \leq \gamma\|v - f^*\|_{\zeta}$.

$$\begin{aligned}\|v-f^*\|^2_{\zeta} &= \|(\hat{\mathrm{I}}-\hat{\Pi}^{\phi}_{\zeta})[v] + \hat{\Pi}^{\phi}_{\zeta}[v] - f^*\|^2_{\zeta} = \|v-\hat{\Pi}^{\phi}_{\zeta}[v]\|^2_{\zeta} + \|\hat{\Pi}^{\phi}_{\zeta}[v]-f^*\|^2_{\zeta}\\ &= \|v-\hat{\Pi}^{\phi}_{\zeta}[v]\|^2_{\zeta} + \|\hat{\Pi}^{\phi}_{\zeta}[\hat{\mathrm{T}}[v]] - \hat{\Pi}^{\phi}_{\zeta}[\hat{\mathrm{T}}[f^*]]\|^2_{\zeta}\\ &\leq \|v-\hat{\Pi}^{\phi}_{\zeta}[v]\|^2_{\zeta} + \gamma^2\|v-f^*\|^2_{\zeta} = \frac{1}{1-\gamma^2}\,\|v-\hat{\Pi}^{\phi}_{\zeta}[v]\|^2_{\zeta}\end{aligned}$$

The second equality stems from *Pythagoras theorem* and the fact that both $\hat{\Pi}^{\phi}_{\zeta}[v]$ and f^* are in the space spanned by the basis functions $\{\phi_i\}_{i=1}^{p}$ and $(\hat{\mathrm{I}} - \hat{\Pi}^{\phi}_{\zeta})[v]$ is orthogonal to this subspace. The last equality stems from the subtraction of $\gamma^2\|v-f^*\|^2_{\zeta}$ and the division by $1-\gamma^2$ of the inequality equation. $\square$

C.4 Proofs from Chapter 4

Lemma 4.2 (repeated from Page 51) *For transition model $P^{\pi} : \mathcal{Z}\times\mathcal{B}(\mathcal{Z}) \to [0,1]$ and any distribution $\zeta : \mathcal{Z} \to [0,1]$, with respect to which the uniform distribution ϑ in $\mathcal{Z}$ is absolutely continuous, i.e. $\vartheta \ll \zeta$, holds under Assumption 4.1, $\forall x,y \in \mathcal{Z}, \forall t \in \mathbb{N}\setminus\{0\}$:*

$$\int \vartheta(dz)\,\frac{d\zeta}{d\vartheta}^{-1}(z)\left(\frac{d(P^{\pi})^t(\cdot|x)}{d\vartheta}(z) - \frac{d(P^{\pi})^t(\cdot|y)}{d\vartheta}(z)\right)^2 = \left\|\frac{d(P^{\pi})^t(\cdot|x)}{d\zeta} - \frac{d(P^{\pi})^t(\cdot|y)}{d\zeta}\right\|^2_{\zeta}.$$

Proof: Note that every distribution is absolutely continuous w.r.t. uniform distribution ϑ. We have thus $\zeta \ll \vartheta \ll \zeta$ and due to Assumption 4.1 one can guarantee that $\frac{d(P^{\pi})^t(\cdot|x)}{d\zeta} \in L^2(\mathcal{Z},\zeta), \forall x \in \mathcal{Z}, \forall t \in \mathbb{N}\setminus\{0\}$. This allows to use in the following properties of Radon-Nikodym derivatives: (i) $\frac{d\zeta}{d\vartheta}^{-1} = \frac{d\vartheta}{d\zeta}$, (ii) $\frac{d\mu}{d\zeta} = \frac{d\mu}{d\vartheta}\frac{d\vartheta}{d\zeta}$ and (iii)

$\int \zeta(dz)\, \frac{d\vartheta}{d\zeta}(z)\, f(z) = \int \vartheta(z)\, f(z), \forall f \in L^2(\mathcal{Z}, \zeta)$.

$$\begin{aligned} d_t(x,y) &= \int \vartheta(dz)\, \tfrac{d\zeta}{d\vartheta}^{-1}(z) \Big(\tfrac{d(P^\pi)^t(\cdot|x)}{d\vartheta}(z) - \tfrac{d(P^\pi)^t(\cdot|y)}{d\vartheta}(z) \Big)^2 \\ &\overset{\text{(i)}}{=} \int \vartheta(dz)\, \tfrac{d\vartheta}{d\zeta}(z) \Big(\tfrac{d(P^\pi)^t(\cdot|x)}{d\vartheta}(z) - \tfrac{d(P^\pi)^t(\cdot|y)}{d\vartheta}(z) \Big)^2 \\ &\overset{\text{(iii)}}{=} \int \zeta(dz)\, \tfrac{d\vartheta}{d\zeta}^2(z) \Big(\tfrac{d(P^\pi)^t(\cdot|x)}{d\vartheta}(z) - \tfrac{d(P^\pi)^t(\cdot|y)}{d\vartheta}(z) \Big)^2 \\ &\overset{\text{(ii)}}{=} \int \zeta(dz) \Big(\tfrac{d(P^\pi)^t(\cdot|x)}{d\zeta}(z) - \tfrac{d(P^\pi)^t(\cdot|y)}{d\zeta}(z) \Big)^2 \\ &\overset{(4.1)}{=} \Big\| \tfrac{d(P^\pi)^t(\cdot|x)}{d\zeta} - \tfrac{d(P^\pi)^t(\cdot|y)}{d\zeta} \Big\|_\zeta^2 . \end{aligned}$$

□

Lemma 4.4 (repeated from Page 51) *Let $\mathcal{Z}$ be the state or state-action space of an MDP with reward function $r \in L^2(\mathcal{Z}, \zeta)$, ergodic transition model $P^\pi : \mathcal{Z} \times \mathcal{B}(\mathcal{Z}) \to [0,1]$ and steady state(-action) distribution ζ. For (Q-)value fix-point $v = r + \gamma \hat{\mathrm{P}}^\pi[v]$ holds*

$$|v(x) - v(y)| \quad \leq \quad \underbrace{\Big(\sum_{t=0}^\infty \gamma^t\, d_t(x,y) \Big)}_{d_\gamma^\pi(x,y)} \; \|r\|_\zeta , \qquad \forall x, y \in \mathcal{Z} .$$

Proof: The equivalent value-definition is $v(x) = \mathbb{E}[\sum_{t=0}^\infty \gamma^t r_t \mid x_0 = x]$. Recall (i) the definitions of the Radon-Nikodym derivative $\int \mu(dz)\, f(z) = \int \zeta(dz)\, \frac{d\mu}{d\zeta}(z)\, f(z)$, $\forall f \in L^2(\mathcal{Z}, \zeta)$, (ii) of state(-action) distribution $(P^\pi)^t(\cdot|x) : \mathcal{B}(\mathcal{Z}) \to [0,1]$ after t steps (starting from state(-action) $x \in \mathcal{Z}$) and (iii) of the diffusion distance $d_t(\cdot,\cdot)$ (last two from Definition 4.3 on Page 51). Using (iv) the inequality $|a+b| \leq |a|+|b|$ recursively and (v) the Cauchy-Schwarz inequality for $L^2(\mathcal{Z}, \zeta)$, i.e. $|\langle f, g \rangle_\zeta| \leq \|f\|_\zeta \, \|g\|_\zeta, \forall f, g \in L^2(\mathcal{Z}, \zeta)$, one can show:

$$\begin{aligned} |v(x) - v(y)| &\overset{\text{(iv)}}{\leq} \sum_{t=0}^\infty \gamma^t \Big| \mathbb{E}[r_t \,|\, x_0 = x] - \mathbb{E}[r_t \,|\, x_0 = y] \Big| \\ &\overset{\text{(ii)}}{=} \sum_{t=0}^\infty \gamma^t \Big| \int (P^\pi)^t(dz|x)\, r(z) - \int (P^\pi)^t(dz|y)\, r(z) \Big| \\ &\overset{\text{(i)}}{=} \sum_{t=0}^\infty \gamma^t \Big| \langle r, \tfrac{d(P^\pi)^t(\cdot|x)}{d\zeta} - \tfrac{d(P^\pi)^t(\cdot|y)}{d\zeta} \rangle_\zeta \Big| \\ &\overset{\text{(v)}}{\leq} \sum_{t=0}^\infty \gamma^t \, \| \tfrac{d(P^\pi)^t(\cdot|x)}{d\zeta} - \tfrac{d(P^\pi)^t(\cdot|y)}{d\zeta} \|_\zeta \, \|r\|_\zeta \\ &\overset{\text{(iii)}}{=} \sum_{t=0}^\infty \gamma^t \; d_t(x,y) \; \|r\|_\zeta . \end{aligned}$$

□

Lemma 4.5 (repeated from Page 52) *Given the eigenfunctions $\varphi_i \in L^2(\mathcal{Z}, \zeta)$ and eigenvalues $\lambda_i \in \mathbb{R}$ of a self-adjoint transition operator in $L^2(\mathcal{Z}, \zeta)$, the distance $d_\gamma^\pi(x,y) := \sum_{t=0}^\infty \gamma^t d_t(x,y)$, $\forall x, y \in \mathcal{Z}$, is bounded by the Euclidean distance between scaled eigenfunctions, i.e.*

$$d_\gamma^\pi(x,y) \;\leq\; \tfrac{1}{\sqrt{1-\gamma}} \|\tilde{\boldsymbol{\psi}}(x) - \tilde{\boldsymbol{\psi}}(y)\|_2 , \quad \tilde{\psi}_i(x) := \tfrac{1}{\sqrt{1-\gamma\lambda_i^2}}\, \varphi_i(x) , \quad \forall i \in \mathbb{N}, \;\; \forall x, y \in \mathcal{Z} .$$

Proof: In the following I will use *Jensen's inequality*: $(\sum_{t=0}^{\infty} p_t x_t)^2 \leq \sum_{t=0}^{\infty} p_t x_t^2$ for $\sum_{t=0}^{\infty} p_t = 1$ and $p_t \geq 0, \forall t \in \mathbb{N}$ (see e.g. Boyd and Vandenberghe, 2004). I will also employ the geometric series $\sum_{t=0}^{\infty} a^t = \frac{1}{1-a}, \forall |a| \in [0,1)$, e.g. $\sum_{t=0}^{\infty}(1-\gamma)\,\gamma^t = 1$.

$$\begin{aligned}\Big(d_\gamma^\pi(x,y)\Big)^2 &= \Big(\sum_{t=0}^{\infty}(1-\gamma)\,\gamma^t\,\tfrac{d_t(x,y)}{1-\gamma}\Big)^2 \quad\leq\quad \sum_{t=0}^{\infty}(1-\gamma)\,\gamma^t\Big(\tfrac{d_t(x,y)}{1-\gamma}\Big)^2 \\ &\overset{(*)}{=} \tfrac{1}{1-\gamma}\sum_{t=0}^{\infty}\gamma^t\sum_{i=0}^{\infty}\lambda_i^{2t}\Big(\varphi_i(x)-\varphi_i(y)\Big)^2 \\ &= \tfrac{1}{1-\gamma}\sum_{i=0}^{\infty}\tfrac{1}{1-\gamma\lambda_i^2}\Big(\varphi_i(x)-\varphi_i(y)\Big)^2 \quad=\quad \tfrac{1}{1-\gamma}\|\tilde{\boldsymbol{\psi}}(x)-\tilde{\boldsymbol{\psi}}(y)\|_2^2\,. \quad \square\end{aligned}$$

The equality marked (*) uses $\big(d_t(x,y)\big)^2 = \sum_{i=0}^{\infty}\lambda_i^{2t}\big(\varphi_i(x)-\varphi_j(y)\big)^2$ from Lemma 3 in Böhmer et al. (2013, Page 2108), which can be found on Page 187.

Proposition 4.6 (repeated from Page 52) *In the limit of an infinite ergodic Markov chain in state or state-action space $\mathcal{Z}$, following transition model P^π with steady state distribution ζ, the first p SFA features in $L^2(\mathcal{Z},\zeta)$ are the eigenfunctions φ_i of the symmetrized transition operator $\hat{\mathrm{P}}_s^\pi := \frac{1}{2}\hat{\mathrm{P}}^\pi + \frac{1}{2}(\hat{\mathrm{P}}^\pi)^*$ in $L^2(\mathcal{Z},\zeta)$, corresponding to the p largest eigenvalues $\lambda_i = 1 - \frac{1}{2}\hat{\mathcal{S}}[\varphi_i]$, except the first constant eigenfunction $\varphi_0(x) = 1\,, \forall x \in \mathcal{Z}$.*

Proof: The eigenfunction to the largest eigenvalue of any transition operator $\hat{\mathrm{P}}^\pi$ is a constant 1, as $\hat{\mathrm{P}}^\pi[1](x) = \int P^\pi(dz|x) = 1, \forall x \in \mathcal{Z}$. From Lemma 4 in Böhmer et al. (2013, see Page 188 of this thesis) follows that the adjoint transition operator $(\hat{\mathrm{P}}^\pi)^* : L^2(\mathcal{Z},\zeta) \to L^2(\mathcal{Z},\zeta)$ is ζ-almost-everywhere a transition operator as well. The first eigenfunction $\varphi_0(x) = 1, \forall x \in \mathcal{Z}$, of the symmetrized operator P_s^π is therefore a constant 1 everywhere, and has the corresponding eigenvalue $\lambda_0 = 1$.

From Lemma 5 in Böhmer et al. (2013, see Page 188 of this thesis) follows that $\hat{\mathcal{S}}[\phi_i] = 2\langle\phi_i, (\hat{\mathrm{I}}-\hat{\mathrm{P}}^\pi[\phi_i]\rangle_\zeta$, and one can therefore transform the optimization problem

$$\inf_\phi \hat{\mathcal{S}}[\phi] \quad=\quad \inf_\phi 2\langle\phi,(\hat{\mathrm{I}}-\hat{\mathrm{P}}^\pi)[\phi]\rangle_\zeta \quad\equiv\quad \sup_\phi 2\langle\phi,\hat{\mathrm{P}}^\pi[\phi]\rangle_\zeta \quad=\quad \sup_\phi 2\langle\phi,\hat{\mathrm{P}}_s^\pi[\phi]\rangle_\zeta\,.$$

This problem is solved by the eigenfunctions φ_i with the largest positive eigenvalues λ_i, i.e. $\inf_i \hat{\mathcal{S}}[\varphi_i] \equiv \sup_i 2\lambda_i$. Due to the *Hilbert-Schmidt theorem* (e.g. Theorem 4.2.23 in Davies, 2007), eigenfunctions of self-adjoint operators are orthogonal in $L^2(\mathcal{Z},\zeta)$ and fulfill therefore the constrains. The only exception is the constant eigenfunction φ_0, which is excluded by the zero mean constraint $\mathbb{E}[\phi_i] = 0$. The slowness of these features is in the limit $\hat{\mathcal{S}}[\varphi_i] = 2(1-\lambda_i)$. $\square$

Proposition 4.10 (repeated from Page 56) *γ-SFA and slow feature analysis (Wiskott and Sejnowski, 2002) have the same solutions for all $\gamma \in (0,1]$, if (i) the infinite training set is drawn from an ergodic Markov chain and (ii) the kernel operator $\hat{\mathrm{K}}$ is the identity operator $\hat{\mathrm{I}}$.*

Proof: In the limit of an infinite ergodic Markov chain drawn by transition model $P^\pi : \mathcal{Z}\times\mathcal{B}(\mathcal{Z}) \to [0,1]$ with steady state(-action) distribution ζ, the constrains are

$$\begin{aligned}\lim_{n\to\infty} \tfrac{1}{n^2}\sum_{s,t=1}^{n}\phi_i(x_s)\,K(x_s,x_t)\,\phi_j(x_t) &= \delta_{ij} \\ \equiv \int\zeta(dx)\,\phi_i(x)\underbrace{\int\zeta(dy)\,K(x,y)\,\phi_j(y)}_{\hat{\mathrm{K}}[\phi_j](x)} &= \delta_{ij}\,.\end{aligned}$$

Using the identity operator $\hat{\mathrm{I}}$ as $\hat{\mathrm{K}}$, the constrains are therefore $\langle \phi_i, \phi_j \rangle_\zeta = \delta_{ij}, \forall i, j$. The constraints guarantee $\|\phi_k\|_\zeta^2 = 1, \forall k$, and the objective of γ-SFA is therefore

$$
\begin{aligned}
& \inf_{\{\phi_k\}_{k=1}^p} \quad \lim_{n\to\infty} \quad \sum_{k=1}^{p} \tfrac{1}{n} \sum_{t=1}^{n} \Big(\phi_k(x_t) - \gamma\, \phi_k(x_{t+1})\Big)^2 \\
\equiv \quad & \inf_{\{\phi_k\}_{k=1}^p} \quad \sum_{k=1}^{p} \iint \zeta(dx)\, P^\pi(dy|x) \Big(\phi_k(x) - \gamma \phi_k(y)\Big)^2 \\
\equiv \quad & \inf_{\{\phi_k\}_{k=1}^p} \quad \sum_{k=1}^{p} \Big(\|\phi_k\|_\zeta^2 - 2\gamma \langle \phi_k, \hat{\mathrm{P}}^\pi[\phi_k] \rangle_\zeta + \gamma^2 \int \underbrace{\textstyle\int \zeta(dx) P^\pi(dy|x)}_{\zeta(dy) \text{ (ergodicity)}} \big(\phi_k(y)\big)^2 \Big) \\
\equiv \quad & \inf_{\{\phi_k\}_{k=1}^p} \quad \sum_{k=1}^{p} \Big((1+\gamma^2) \overbrace{\|\phi_k\|_\zeta^2}^{1} - 2\,\gamma \langle \phi_k, \hat{\mathrm{P}}^\pi[\phi_k] \rangle_\zeta \Big) \\
\equiv \quad & \sup_{\{\phi_k\}_{k=1}^p} \quad \sum_{k=1}^{p} \langle \phi_k, \hat{\mathrm{P}}^\pi[\phi_k] \rangle_\zeta \quad \equiv \quad \sup_{\{\phi_k\}_{k=1}^p} \quad \sum_{k=1}^{p} \langle \phi_k, \hat{\mathrm{P}}_s^\pi[\phi_k] \rangle_\zeta \,,
\end{aligned}
$$

where the $\hat{\mathrm{P}}_s^\pi := \frac{1}{2}\hat{\mathrm{P}}^\pi + \frac{1}{2}(\hat{\mathrm{P}}^\pi)^*$ is the symmetrized transition operator, which is self-adjoint and can therefore be written as $\hat{\mathrm{P}}_s^\pi[\cdot] = (\hat{\mathrm{P}}_s^\pi)^{\frac{1}{2}}[(\hat{\mathrm{P}}_s^\pi)^{\frac{1}{2}}[\cdot]]$. The problem

$$
\sup_{\{\phi_k\}_{k=1}^p} \quad \sum_{k=1}^{p} \Big\| (\hat{\mathrm{P}}_s^\pi)^{\frac{1}{2}}[\phi_k] \Big\|_\zeta^2 \qquad \text{s.t.} \quad \langle \phi_i, \phi_j \rangle_\zeta = \delta_{ij} \,, \quad \forall i, j \in \{1, \ldots, p\} \,,
$$

is the eigenvalue problem of operator $(\hat{\mathrm{P}}_s^\pi)^{\frac{1}{2}}$, which has the same eigenfunctions as $\hat{\mathrm{P}}_s^\pi$. Proposition 4.6 proves that these eigenfunctions are solutions to SFA. □

Proposition 4.11 (repeated from Page 56) *γ-SFA optimizes an* UPPER BOUND *on the objective in Definition 4.8 under the assumption that: (i) the infinite training set in state(-action) space $\mathcal{Z}$ is drawn by an ergodic Markov chain with steady state(-action) distribution ζ, (ii) the set of anticipated policies ω contains only the sampling policy π and (iii) MDP in ρ have the same transition model P and reward functions are distributed with zero mean and positive-definite covariance kernel function $K \in L^\infty(\mathcal{Z} \times \mathcal{Z})$.*

Proof: The optimization problem for optimal features in Definition 4.8 uses an expectation over policies and MDP. According to the assumptions, the policy π is the sampling policy and the MDP varies only their reward functions $r \in L^2(\mathcal{Z}, \zeta)$. These have zero mean and kernel K as covariance, i.e. $\mathbb{E}[r(x)\, r(y)] = K(x, y), \forall x, y \in \mathcal{Z}$.

The average projection error over values $v = (\hat{\mathrm{I}} - \gamma\hat{\mathrm{P}}^\pi)^{-1}[r] \in L^2(\mathcal{Z}, \zeta)$ is therefore:

$$
\begin{aligned}
& \inf_{\{\phi_i\}_{i=1}^p} && \mathbb{E}\Big[\big\|v - \hat{\Pi}_\zeta^\phi[v]\big\|_\zeta^2\Big] && && \\
\overset{(a)}{\equiv} & \sup_{\{\phi_i\}_{i=1}^p} && \mathbb{E}\Big[\langle v, \hat{\Pi}_\zeta^\phi[v]\rangle_\zeta\Big] && && \\
\overset{(b)}{\equiv} & \sup_{\{\phi_i\}_{i=1}^p} && \sum_{i=1}^p \mathbb{E}\Big[\langle \phi_i, (\hat{\mathrm{I}} - \gamma\hat{\mathrm{P}}^\pi)^{-1}[r]\rangle_\zeta^2\Big] && \text{s.t.} \quad & \langle \phi_i, \phi_j\rangle_\zeta = \delta_{ij}, \forall i, j\,, \\
\overset{(c)}{\equiv} & \sup_{\{\phi_i\}_{i=1}^p} && \sum_{i=1}^p \Big\|\hat{\mathrm{K}}^{\frac{1}{2}}\big[(\hat{\mathrm{I}} - (\hat{\mathrm{P}}^\pi)^*)^{-1}[\phi_i]\big]\Big\|_\zeta^2 && \text{s.t.} \quad & \langle \phi_i, \phi_j\rangle_\zeta = \delta_{ij}, \forall i, j\,, \\
\overset{(d)}{\equiv} & \inf_{\{\phi_i\}_{i=1}^p} && \sum_{i=1}^p \Big\|\phi_i - \gamma(\hat{\mathrm{P}}^\pi)^*[\phi_i]\Big\|_\zeta^2 && \text{s.t.} \quad & \langle \phi_i, \hat{\mathrm{K}}[\phi_j]\rangle_\zeta = \delta_{ij}, \forall i, j\,, \\
\overset{(e)}{\leq} & \inf_{\{\phi_i\}_{i=1}^p} && \sum_{i=1}^p \Big((1+\gamma^2)\|\phi_i\|_\zeta^2 - 2\gamma\langle \phi_i, \hat{\mathrm{P}}^\pi[\phi_i]\rangle_\zeta\Big) && \text{s.t.} \quad & \langle \phi_i, \hat{\mathrm{K}}[\phi_j]\rangle_\zeta = \delta_{ij}, \forall i, j\,, \\
\overset{(f)}{\equiv} & \inf_{\{\phi_i\}_{i=1}^p} && \sum_{i=1}^p \hat{\mathcal{S}}_\gamma[\phi_i] && \text{s.t.} \quad & \langle \phi_i, \hat{\mathrm{K}}[\phi_j]\rangle_\zeta = \delta_{ij}, \forall i, j\,.
\end{aligned}
$$

The above equivalences and inequality are in the following explained in detail:

(a) Firstly, note that $\|v - \hat{\Pi}_\zeta^\phi[v]\|_\zeta^2 = \|v\|_\zeta^2 - \langle v, \hat{\Pi}_\zeta^\phi[v]\rangle_\zeta$. Secondly, $\|v\|_\zeta^2$ is constant w.r.t. $\phi_i, \forall i$, and therefore holds $\inf_\phi \|v - \hat{\Pi}_\zeta^\phi[v]\|_\zeta^2 \equiv \sup_\phi \langle v, \hat{\Pi}_\zeta^\phi[v]\rangle_\zeta$.

(b) The basis functions $\{\phi_i\}_{i=1}^p$ describe a *subspace* of $L^2(\mathcal{Z}, \zeta)$. The subspace remains the same, independent of the features scaling or correlation. I introduce the constrains $\langle \phi_i, \phi_j\rangle_\zeta = \delta_{ij}, \forall i, j$, and the value function $v = (\hat{\mathrm{I}} - \gamma\hat{\mathrm{P}}^\pi)^{-1}[r]$.

(c) The positive semi-definite kernel K denotes the covariance between the anticipated reward functions, i.e. $\mathbb{E}[r(x)\, r(y)] = K(x, y), \forall x, y \in \mathcal{Z}$. K induces the linear operator $\hat{\mathrm{K}}[f](x) := \int \zeta(dy)\, K(x, y)\, f(y),\ \forall y \in \mathcal{Z}, \forall f \in L^2(\mathcal{Z}, \zeta)$, which is self-adjoint and according Section 3.1.2 can be written as two operators $\hat{\mathrm{K}}^{\frac{1}{2}}$ with the same eigenfunctions and square root eigenvalues: $\hat{\mathrm{K}}[f] = \hat{\mathrm{K}}^{\frac{1}{2}}[\hat{\mathrm{K}}^{\frac{1}{2}}[f]], \forall f \in L^2(\mathcal{Z}, \zeta)$. Using the shorthand $f := (\hat{\mathrm{I}} - \gamma(\hat{\mathrm{P}}^\pi)^*)^{-1}[\phi_i] \in L^2(\mathcal{Z}, \zeta)$, one can therefore write

$$
\mathbb{E}[\langle f, r\rangle_\zeta^2] = \iint \zeta(dx)\, \zeta(dy)\, f(x)\, f(y)\, \mathbb{E}[r(x)\, r(y)] = \langle f, \hat{\mathrm{K}}[f]\rangle_\zeta = \|\hat{\mathrm{K}}^{\frac{1}{2}}[f]\|_\zeta^2 .
$$

(d) Let $\hat{\mathrm{A}}$ denote the operator $\hat{\mathrm{I}} - (\hat{\mathrm{P}}^\pi)^*$, which is invertible due to Lemmas 14 and 4 in Böhmer et al. (2013, proofs can be found on Pages 191 and 188, respectively, of this thesis). The optimization problem is equivalent to the operator norm

$$
\begin{aligned}
\|\hat{\mathrm{K}}^{\frac{1}{2}}[\hat{\mathrm{A}}^{-1}]\|_\zeta \quad &:= \quad \sup_{f \in L^2(\mathcal{Z}, \zeta)} \Big\{ \frac{\|\hat{\mathrm{K}}^{\frac{1}{2}}[\hat{\mathrm{A}}^{-1}[f]]\|_\zeta}{\|f\|_\zeta} \,\Big|\, f \neq 0 \Big\} \\
&\equiv \quad \sup_{f \in L^2(\mathcal{Z}, \zeta)} \Big\{ \frac{\|\hat{\mathrm{K}}^{\frac{1}{2}}[f]\|_\zeta}{\|\hat{\mathrm{A}}[f]\|_\zeta} \,\Big|\, f \neq 0 \Big\} \\
&\equiv \quad \inf_{f \in L^2(\mathcal{Z}, \zeta)} \Big\{ \frac{\|\hat{\mathrm{A}}[f]\|_\zeta}{\|\hat{\mathrm{K}}^{\frac{1}{2}}[f]\|_\zeta} \,\Big|\, f \neq 0 \Big\} ,
\end{aligned}
$$

which is in turn equivalent to the problem after the transformation.

(e) $\|\phi_i - \gamma(\hat{\mathrm{P}}^\pi)^*[\phi_i]\|_\zeta^2 = \|\phi_i\|_\zeta^2 - 2\gamma\langle\phi_i, \hat{\mathrm{P}}^\pi[\phi_i]\rangle_\zeta + \gamma^2\|(\hat{\mathrm{P}}^\pi)^*[\phi_i]\|_\zeta^2$. Lemma 4 of Böhmer et al. (2013) shows that $(\hat{\mathrm{P}}^\pi)^*$ is a transition operator with steady state distribution ζ. One can show with Lemma C.2 that $\|(\hat{\mathrm{P}}^\pi)^*[\phi_i]\|_\zeta^2 \leq \|\phi_i\|_\zeta^2$.

(f) The slowness $\hat{\mathcal{S}}_\gamma$ of γ-SFA is in the limit of an infinite ergodic Markov chain

$$\begin{aligned}
\hat{\mathcal{S}}_\gamma[\phi_i] &:= \lim_{n\to\infty} \tfrac{1}{n}\sum_{t=1}^{n}\Big(\phi_i(x_t) - \gamma\,\phi_i(x_{t+1})\Big)^2 \\
&= \int \zeta(dx)\Big(\phi_i(x)\Big)^2 - 2\,\gamma \iint \zeta(dx)\,P(dy|x)\,\phi_i(x)\,\phi_i(y) \\
&\quad + \gamma^2 \int \underbrace{\textstyle\int \zeta(dx)\,P^\pi(dy|x)}_{\text{ergodicity: } \zeta(dy)} \Big(\phi(y)\Big)^2 \\
&= (1+\gamma^2)\,\|\phi_i\|_\zeta^2 \;-\; 2\,\gamma\,\langle\phi_i, \hat{\mathrm{P}}^\pi[\phi_i]\rangle_\zeta\,.
\end{aligned}$$

The corresponding γ-SFA constraints can, in the limit of an infinite ergodic Markov chain, be expressed as $\langle\phi_i, \hat{\mathrm{K}}[\phi_j]\rangle_\zeta = \delta_{ij}$, using the kernel operator $\hat{\mathrm{K}}$. □

Bibliography

Abbeel, P. and Ng, A. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pages 1–8, 2004.

Abbeel, P.; Coates, A.; Quigley, M., and Ng, A. Y. An application of reinforcement learning to aerobatic helicopter flight. In *Advances in Neural Information Processing Systems*, pages 1–8, 2007.

Alissandrakis, A.; Nehaniv, C. L., and Dautenhahn, K. Correspondence mapping induced state and action metrics for robotic imitation. *IEEE Transactions on systems, man, and cybernetics, part B: cybernetics*, 37(2):299–307, 2007.

Allender, E.; Arora, S.; Kearns, M.; Moore, C., and Russell, A. A note on the representational incompatibility of function approximation and factored dynamics. In *Advances in Neural Information Processing Systems 15*, pages 431–437, 2002.

Anderson, M. L. Embodied cognition: A field guide. *Artificial Intelligence*, 149: 91–130, 2003.

Aravkin, A.; Burke, J. V.; Chiuso, A., and Pillonetto, G. Convex vs non-convex estimators for regression and sparse estimation: The mean squared error properties of ard and glasso. *Journal of Machine Learning Research*, 15(1):217–252, January 2014.

Argall, B. D.; Chernova, S.; Veloso, M., and Browning, B. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, May 2009.

Bach, F. R. and Jordan, M. I. Learning spectral clustering. In *Advances In Neural Information Processing Systems 16*, 2003.

Baird, L. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 30–37. Morgan Kaufmann, 1995.

Barto, A. and Sutton, R. Simulation of anticipatory responses in classical conditioning by a neuron-like adaptive element. *Behavioral Brain Research*, 4:221–235, 1982.

Barto, A.; Sutton, R., and Watkins, C. *Learning and sequential decision making*, pages 539–602. MIT Press, 1990.

Bear, M.; Connors, B., and Paradiso, M. *Neuroscience: Exploring the Brain*. Lippincott Williams & Wilkins, 2001. ISBN 9780781732550.

Bellemare, M.; Veness, J., and Bowling, M. Bayesian learning of recursively factored environments. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 1211–1219, 2013. URL http://jmlr.org/proceedings/papers/v28/bellemare13.pdf.

Bellman, R. E. *Dynamic programming*. Princeton University Press, 1957.

Bertsekas, D. P. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, 3rd edition, 2007.

Biggs, M.; Ghodsi, A.; Wilkinson, D., and Bowling, M. Scalable action respecting embedding. In *Proceedings of the 10th International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, 2008. URL https://webdocs.cs.ualberta.ca/~bowling/papers/08isaim-are.pdf.

Bishop, C. *Pattern Recognition and Machine Learning*. Springer Verlag, 2006. ISBN 978-0-387-31073-2.

Bitzer, S.; Howard, M., and Vijayakumar, S. Using dimensionality reduction to exploit constraints in reinforcement learning. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pages 3219–3225, 2010.

Böhmer, W. and Obermayer, K. Towards structural generalization: Factored approximate planning. ICRA Workshop on Autonomous Learning, 2013. URL http://autonomous-learning.org/wp-content/uploads/13-ALW/paper_1.pdf.

Böhmer, W. and Obermayer, K. Regression with linear factored functions. In *Machine Learning and Knowledge Discovery in Databases*, volume 9284 of *Lecture Notes in Computer Science*, pages 119–134. Springer International Publishing, 2015.

Böhmer, W.; Grünewälder, S.; Nickisch, H., and Obermayer, K. Regularized sparse kernel slow feature analysis. In *Machine Learning and Knowledge Discovery in Databases*, volume I of *LNAI 6911*, pages 235–248, 2011.

Böhmer, W.; Grünewälder, S.; Nickisch, H., and Obermayer, K. Generating feature spaces for linear algorithms with regularized sparse kernel slow feature analysis. *Machine Learning*, 89(1-2):67–86, 2012.

Böhmer, W.; Grünewälder, S.; Shen, Y.; Musial, M., and Obermayer, K. Construction of approximation spaces for reinforcement learning. *Journal of Machine Learning Research*, 14:2067–2118, July 2013.

Böhmer, W.; Springenberg, J. T.; Boedecker, J.; Riedmiller, M., and Obermayer, K. Autonomous learning of state representations for control: An emerging field aims to autonomously learn state representations for reinforcement learning agents from their real-world sensor observations. *KI - Künstliche Intelligenz*, 29(4): 353–362, 2015. URL http://www.redaktion.tu-berlin.de/fileadmin/fg215/articles/boehmer15b.pdf.

Boser, B. E.; Guyon, I. M., and Vapnik, V. N. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, 1992.

Bostan, I.; Gheorghe, A.; Dulgheru, V.; Sobor, I.; Bostan, V., and Sochirean, A. *Resilient Energy Systems: Renewables: Wind, Solar, Hydro*. Topics in Safety, Risk, Reliability and Quality. Springer Netherlands, 2012. ISBN 9789400741898.

Boutilier, C.; Dean, T., and Hanks, S. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

Bowling, M.; Ghodsi, A., and Wilkinson, D. Action respecting embedding. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 65–72, 2005.

Bowling, M.; Wilkinson, D., and Ghodsi, A. Subjective mapping. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 1569–1572, 2006.

Boyan, J. A. and Moore, A. W. Generalization in reinforcement learning: safely approximating the value function. In *Advances in Neural Information Processing Systems*, pages 369–376, 1995.

Boyd, S. and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, 2004.

Bradtke, S. J. and Barto, A. G. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1/2/3):33–57, 1996.

Bray, A. and Martinez, D. Kernel-based extraction of Slow features: Complex cells learn disparity and translation invariance from natural images. In *Advances in Neural Information Processing Systems*, volume 15, pages 253–260, 2002.

Buswell, G. T. *How People Look at Pictures*. University of Chicago Press, 1935.

Chakraborty, D. and Stone, P. Structure learning in ergodic factored MDPs without knowledge of the transition function's in-degree. In *Proceedings of the 28th International Conference on Machine Learning*, pages 737–744, 2011.

Champandard, A. J. Monte-Carlo tree search in TOTAL WAR: ROME II's campaign AI. URL: `http://aigamedev.com/open/coverage/mcts-rome-ii`, 2014. Webpage accessed 2015-05-22.

Choi, D. and Roy, B. A generalized kalman filter for fixed point approximation and efficient temporal-difference learning. *Discrete Event Dynamic Systems*, 16 (2):207–239, 2006.

Coifman, R.; Lafon, S.; Lee, A.; Maggioni, M.; Nadler, B.; Warner, F., and Zucker, S. Geometric diffusions as a tool for harmonic analysis and structure definition of data. Part I: diffusion maps. *Proceedings of the National Academy of Science*, 102(21):7426 – 7431, May 2005.

Coradeschi, S.; Loutfi, A., and Wrede, B. A short review of symbol grounding in robotic and intelligent systems. *KI – Künstliche Intelligenz*, 27(2):129–136, 2013.

Coulom, R. Efficient selectivity and backup operators in Monte-Carlo tree search. In *5th International Conference of Computers and Games*, pages 72–83, 2006.

Cover, T. and Hart, P. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

Csató, L. and Opper, M. Sparse on-line Gaussian processes. *Neural Computation*, 14(3):641–668, 2002.

Cully, A. and Mouret, J.-B. Evolving a behavioral repertoire for a walking robot. *Evolutionary Computation Journal*, 1:1–33, 2015.

Dagum, P.; Galper, A., and Horvitz, E. Dynamic network models for forecasting. In *Proceedings of the Eighth International Conference on Uncertainty in Artificial Intelligence*, UAI'92, pages 41–48, 1992.

Dann, C.; Neumann, G., and Peters, J. Policy evaluation with temporal differences: A survey and comparison. *Journal of Machine Learning Research*, 15:809–883, 2014.

Davies, E. B. *Linear Operators and their Spectra*. Cambridge University Press, 2007.

Davison, A. J. Real-time simultaneous localization and mapping with a single camera. In *IEEE International Conference on Computer Vision*, volume 2, page 1403, 2003.

Dayan, P. and Abbott, L. F. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT Press, 2005. ISBN 0262541858.

Degris, T.; Sigaud, O., and Wuillemin, P.-H. Learning the structure of factored Markov decision processes in reinforcement learning problems. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 257–264, 2006.

Deisenroth, M. P.; Neumann, G., and Peters, J. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142, 2013.

Delgado, K.; Sanner, S.; de Barros, L., and Cozman, F. Efficient solutions to factored mdps with imprecise transition probabilities. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling*, 2009.

Dietterich, T.; Domingos, P.; Getoor, L.; Muggleton, S., and Tadepalli, P. Structured machine learning: the next ten years. *Machine Learning*, 73(1):3–23, 2008. ISSN 0885-6125.

Duda, R. and Hart, P. *Pattern Classification and Scene Analysis*. Wiley, 1973.

Džeroski, S.; Raedt, L. D., and Drissens, K. Relational reinforcement learning. *Machine Learning*, 43:7–52, 2001.

Engel, Y.; Mannor, S., and Meir, R. Bayes meets Bellman: the Gaussian process approach to temporal difference learning. In *International Conference on Machine Learning*, pages 154–161, 2003.

Englert, P.; Paraschos, A.; Peters, J., and Deisenroth, M. P. Model-based imitation learning by probabilistic trajectory matching. In *IEEE International Conference on Robotics and Automation*, 2013.

Ferguson, K. and Mahadevan, S. Proto-transfer learning in Markov decision processes using spectral methods. In *ICML Workshop on Transfer Learning*, 2006.

Ferrante, E.; Lazaric, A., and Restelli, M. Transfer of task representation in reinforcement learning using policy-based proto-value functions. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, 2008.

Ferrucci, D. A. Introduction to "This is Watson". *IBM Journal of Research and Development*, 56(3):235–249, 2012. ISSN 0018-8646.

Fikes, R. E. and Nilsson, N. J. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(2–4):189–208, 1971.

Fisher, R. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

Floreano, D. and Wood, R. J. Science, technology and the future of small autonomous drones. *Nature*, 521(7553):460–466, May 2015.

Foster, D. J. and Dayan, P. Structure in the space of value functions. *Machine Learning*, 49(2–3):325–346, 2002.

Franzius, M.; Sprekeler, H., and Wiskott, L. Slowness and sparseness leads to place, head-direction, and spatial-view cells. *PLoS Computational Biology*, 3(8):e166, 2007.

Fukumizu, K.; Bach, F., and Gretton, A. Statistical consistency of kernel canonical correlation analysis. *Journal of Machine Learning Research*, 8:361–383, 2007.

Geist, M. and Pietquin, O. Kalman temporal differences. *Journal of Artificial Intelligence Research*, 39(1):483–532, 2010.

Gens, R. and Domingos, P. M. Learning the structure of sum-product networks. In *Proceedings of the 30th International Conference on Machine Learning*, pages 873–880, 2013.

Glynn, P. W. and Iglehart, D. L. Importance sampling for stochastic simulations. *Management Science*, 35(11):1367–1392, 1989.

Guestrin, C.; Koller, D., and Parr, R. Max-norm projections for factored MDPs. In *International Joint Conference on Artificial Intelligence*, pages 673–682, 2001a.

Guestrin, C.; Hauskrecht, M., and Kveton, B. Solving factored MDPs with continuous and discrete variables. In *Uncertainty in Artificial Intelligence*, pages 235–242, 2004.

Guestrin, C.; Koller, D., and Parr, R. Multiagent planning with factored MDPs. In *Advances in Neural Information Processing Systems 14*, pages 1523–1530, 2001b.

Guestrin, C.; Koller, D., and Parr, R. Solving factored POMDPs with linear value functions. In *In IJCAI-01 workshop on Planning under Uncertainty and Incomplete Information*, 2001c.

Guestrin, C.; Koller, D.; Gearhart, C., and Kanodia, N. Generalizing plans to new environments in relational MDPs. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 1003–1010, 2003a.

Guestrin, C.; Koller, D.; Parr, R., and Venkataraman, S. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research (JAIR)*, 19: 399–468, 2003b.

Hauskrecht, M. and Kveton, B. Linear program approximations for factored continuous-state Markov decision processes. In *Advances in Neural Information Processing Systems*, pages 895–902, 2003.

Hauskrecht, M. and Kveton, B. Approximate linear programming for solving hybrid factored MDPs. In *Proceedings of the 9th International Symposium on Artificial Intelligence and Mathematics*, 2006.

Haykin, S. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2nd edition, 1998. ISBN 978-0132733502.

Hebb, D. O. *The Organization of Behavior*. John Wiley, New York, 1949.

Hinton, G. E. and Osindero, S. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

Hochreiter, S.; Bengio, Y.; Frasconi, P., and Schmidhuber, J. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In Kremer and Kolen, editors, *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press, 2001.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Hoerl, A. E. and Kennard, R. W. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67, 1970.

Höfer, S.; Hild, M., and Kubisch, M. Using slow feature analysis to extract behavioural manifolds related to humanoid robot postures. In *Tenth International Conference on Epigenetic Robotics*, pages 43–50, 2010.

Höfer, S.; Lang, T., and Brock, O. Extracting kinematic background knowledge from interactions using task-sensitive relational learning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2014.

Houillon, A.; Lorenz, R.; Böhmer, W.; Rapp, M.; Heinz, A.; Klaus, J. G., and Obermayer. The effect of novelty on reinforcement learning. *Progress in Brain Research*, 202:415–439, 2013.

Hsu, F.-h. *Behind Deep Blue: Building the Computer that Defeated the World Chess Champion*. Princeton University Press, 2002. ISBN 0-691-09065-3.

Ivan, V.; Zarubin, D.; Toussaint, M.; Komura, T., and Vijayakumar, S. Topology-based representations for motion planning and generalisation in dynamic environments with interactions. *International Journal of Robotics Research*, 32:1151–1163, 2013.

Jakab, H. and Csató, L. Manifold-based non-parametric learning of action-value functions. In *20th European Symposium on Artificial Neural Networks, ESANN 2012*, 2012. URL https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2012-179.pdf.

Jenkins, O. C. and Matarić, M. J. A spatio-temporal extension to isomap nonlinear dimension reduction. In *The International Conference on Machine Learning*, pages 441–448, 2004.

Jetchev, N. and Toussaint, M. Discovering relevant task spaces using inverse feedback control. *Autonomous Robots*, 37:169–189, 2014.

Jetchev, N.; Lang, T., and Toussaint, M. Learning grounded relational symbols from continuous data for abstract reasoning. ICRA Workshop on Autonomous Learning, 2013. URL http://autonomous-learning.org/wp-content/uploads/13-ALW/paper_5.pdf.

Jodogne, S. and Piater, J. H. Closed-loop learning of visual control policies. *Journal of Artificial Intelligence Research*, 28:349–391, 2007.

Johns, J. and Mahadevan, S. Constructing basis functions from directed graphs for value function approximation. In *International Conference on Machine Learning*, 2007.

Johns, J.; Painter-Wakefield, C., and Parr, R. Linear complementarity for regularized policy evaluation and improvement. In *Advances in Neural Information Processing Systems*, pages 1009–1017. Curran Associates, Inc., 2010.

Jonschkowski, R. and Brock, O. State representation learning in robotics: using prior knowledge about physical interaction. In *Proceedings of Robotics: Science and Systems*, 2014.

Kaelbling, L. P.; Littman, M. L., and Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

Kalman, R. E. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960. doi: 10.1115/1.3662552.

Karakovskiy, S. and Togelius, J. The Mario AI benchmark and competitions. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):55 – 67, 2012.

Kearns, M. and Singh, S. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49:209–232, 2002.

Keller, P. W.; Mannor, S., and Precup, D. Automatic basis function construction for approximate dynamic programming and reinforcement learning. In *International Conference on Machine Learning*, 2007.

Kersting, K.; Otterlo, M. V., and Raedt, L. D. Bellman goes relational. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 465–472, 2004.

Kim, S.; Shukla, A., and Billard, A. Catching objects in flight. *IEEE Transactions on Robotics*, 30(5):1049–1065, 2014.

Klein, E.; Geist, M., and Pietquin, O. Batch, off-policy and model-free apprenticeship learning. In *EWRL*, volume 7188 of *Lecture Notes in Computer Science*, pages 285–296. Springer, 2011.

Kober, J.; Bagnell, D., and Peters, J. Reinforcement learning in robotics: A survey. *International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Koller, D. and Parr, R. Computing factored value functions for policies in structured MDPs. In *International Joint Conference on Artificial Intelligence*, pages 1332–1339, 1999.

Koller, D. and Parr, R. Policy iteration for factored MDPs. In *Uncertainty in Artificial Intelligence*, pages 326–334, 2000.

Kolter, J. Z. and Ng, A. Y. Regularization and feature selection in least-squares temporal difference learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 521–528, 2009.

Konda, V. R. and Tsitsiklis, J. N. On actor-critic algorithms. *SIAM Journal on Control and Optimization*, 2003.

Konidaris, G. D.; Osentoski, S., and Thomas, P. Value function approximation in reinforcement learning using the Fourier basis. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence*, 2011.

Konidaris, G.; Kuindersma, S.; Barto, A. G., and Grupen, R. A. Constructing skill trees for reinforcement learning agents from demonstration trajectories. In *Advances in Neural Information Processing Systems 21*, pages 1162–1170, 2010.

Kveton, B. and Theocharous, G. Kernel-based reinforcement learning on representative states. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012. URL `http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/4967`.

Lagoudakis, M. G. and Parr, R. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

Land, M. F. Motion and vision: why animals move their eyes. *Journal of Comparative Physiology A*, 185(4):341–352, 1999.

Lang, T. and Toussaint, M. Planning with noisy probabilistic relational rules. *Journal of Artificial Intelligence Research*, 39:1–49, 2010.

Lang, T.; Toussaint, M., and Kersting, K. Exploration in relational domains for model-based reinforcement learning. *Journal of Machine Learning Research*, 13 (1):3725–3768, December 2012.

Lange, S. and Riedmiller, M. Deep auto-encoder neural networks in reinforcement learning. In *Proceedings of the International Joint Conference on Neural Networks*, 2010.

Lange, S.; Riedmiller, M., and Voigtlaender, A. Autonomous reinforcement learning on raw visual input data in a real world application. In *International Joint Conference on Neural Networks, Brisbane, Australia*, 2012.

Lee, C.-S.; Wang, M.-H.; Chaslot, G.; Hoock, J.-B.; Rimmel, A.; Teytaud, O.; Tsai, S.-R.; Hsu, S.-C., and Hong, T.-P. The computational intelligence of MoGo revealed in Taiwan's computer Go tournaments. *IEEE Trans. Comput. Intellig. and AI in Games*, 1(1):73–89, 2009.

Lee, H.; Battle, A.; Raina, R., and Ng, A. Y. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems*, pages 801–808, 2007.

Legenstein, R.; Wilbert, N., and Wiskott, L. Reinforcement learning on slow features of high-dimensional input streams. *PLoS Computational Biology*, 6(8):e1000894, 2010.

Li, Y.; Campbell, C., and Tipping, M. E. Bayesian automatic relevance determination algorithms for classifying gene expression data. *Bioinformatics*, 18(10): 1332–1339, 2002.

Littman, M. L.; Sutton, R. S., and Singh, S. Predictive representations of state. In *Advances In Neural Information Processing Systems 14*, pages 1555–1561. MIT Press, 2001.

Lowe, D. G. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, 1999.

Luciw, M. and Schmidhuber, J. Low complexity proto-value function learning from sensory observations with incremental slow feature analysis. In *International Conference on Artificial Neural Networks and Machine Learning*, volume III, pages 279–287. Springer-Verlag, 2012.

Lücke, J. and Eggert, J. Expectation truncation and the benefits of preselection in training generative models. *Journal of Machine Learning Research*, 11:2855–2900, 2010.

Lukic, L.; Santos-Victor, J., and Billard, A. Learning robotic eye-arm-hand coordination from human demonstration: A coupled dynamical systems approach. *Biological Cybernetics*, 108(2):223–248, April 2014.

Mahadevan, S. and Liu, B. Basis construction from power series expansions of value functions. In *Advances in Neutral Information Processing Systems*, pages 1540–1548, 2010.

Mahadevan, S. and Maggioni, M. Proto-value functions: a Laplacian framework for learning representations and control in Markov decision processes. *Journal of Machine Learning Research*, 8:2169–2231, 2007.

Mahadevan, S.; Giguere, S., and Jacek, N. Basis adaptation for sparse nonlinear reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2013. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI13/paper/view/6430/7228.

Mairal, J.; Bach, F.; Ponce, J., and Sapiro, G. Online dictionary learning for sparse coding. In *Proceedings of the International Conference on Machine Learning*, pages 689–696, 2009.

Mattner, J.; Lange, S., and Riedmiller, M. A. Learn to swing up and balance a real pole based on raw visual input data. In *Proceedings to the 19th International Conference on Neural Information Processing, Part V*, pages 126–133, 2012.

McCulloch, W. and Pitts, W. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:127–147, 1943.

Menache, I.; Mannor, S., and Shimkin, N. Basis function adaptation in temporal difference reinforcement learning. *Annals of Operations Research*, 134:215–238, 2005.

Mercer, J. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 209(441-458):415–446, 1909. ISSN 0264-3952. doi: 10.1098/rsta.1909.0016.

Millington, I. and Funge, J. *Artificial Intelligence for Games, Second Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2009. ISBN 0123747317, 9780123747310.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

Nash, J. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.

Nath, A. and Domingos, P. M. Learning relational sum-product networks. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2878–2886, 2015.

Nedić, A. and Bertsekas, D. Least squares policy evaluation algorithms with linear function approximation. *Discrete Event Dynamic Systems*, 13(1-2):79–110, 2003.

Nehaniv, C. L. and Dautenhahn, K. Imitation in animals and artifacts. pages 41–61. MIT Press, Cambridge, MA, USA, 2002. ISBN 0-262-04203-7.

Neumann, J. V. and Morgenstern, O. *Theory of Games and Economic Behavior*. Princeton University Press, 1944. ISBN 0691119937.

Olshausen, B. A. and Field, D. J. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

Orthey, A.; Toussaint, M., and Jetchev, N. Optimizing motion primitives to make symbolic models more predictive. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2868–2873, 2013.

Osborne, M. J. and Rubinstein, A. *A course in game theory*. The MIT Press, Cambridge, USA, 1994. ISBN 0-262-65040-1.

Painter-Wakefield, C. and Parr, R. Greedy algorithms for sparse reinforcement learning. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 2012.

Parikh, D. and Grauman, K. Relative attributes. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 503–510, 2011.

Parr, R.; Painter-Wakefield, C.; Li, L., and Littman, M. Analyzing feature generation for value-function approximation. In *International Conference on Machine Learning*, 2007.

Parr, R.; Li, L.; Taylor, G.; Painter-Wakefiled, C., and Littman, M. L. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *International Conference on Machine Learning*, 2008.

Pasula, H. M.; Zettlemoyer, L. S., and Kaelbling, L. P. Learning symbolic models of stochastic domains. *Journal of Artificial Intelligence Research*, 29:309–352, 2007.

Pearson, K. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series 6*, 2(11):559 – 572, 1901.

Peharz, R.; Tschiatschek, S.; Pernkopf, F., and Domingos, P. M. On theoretical properties of sum-product networks. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, 2015.

Perkins, T. J. and Precup, D. A convergent form of approximate policy iteration. In Becker, S.; Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 1595–1602. MIT Press, 2002.

Petrik, M. An analysis of Laplacian methods for value function approximation in MDPs. In *International Joint Conference on Artificial Intelligence*, pages 2574–2579, 2007.

Petrik, M. and Zilberstein, S. Robust approximate bilinear programming for value function approximation. *Journal of Machine Learning Research*, 12:3027–3063, 2011.

Petrik, M. and Zilberstein, S. A bilinear programming approach for multiagent planning. *Journalof Artificial Intelligence Research*, 35:235–274, 2009.

Poon, H. and Domingos, P. M. Sum-product networks: A new deep architecture. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 337–346, 2011.

Poupart, P.; Boutilier, C.; Patrascu, R., and Schuurmans, D. Piecewise linear value function approximation for factored MDPs. In *AAAI Conference on Innovative Applications of Artificial Intelligence*, pages 292–299, 2002.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Rescorla, R. A. and Wagner, A. W. *A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement*, chapter 3, pages 64–99. Appleton-Century-Crofts, New York, 1972.

rethink robotics. The Baxter industrial robot. URL: `http://www.rethinkrobotics.com/baxter`, 2008. Webpage accessed 2015-10-27.

Ribas-Fernandes, J.; Solway, A.; Diuk, C.; McGuire, J.; Barto, A.; Niv, Y., and Botvinick, M. A neural signature of hierarchical reinforcement learning. *Neuron*, 71(2):370–379, 2011. ISSN 0896-6273.

Riedmiller, M.; Gabel, T.; Hafner, R., and Lange, S. Reinforcement learning for robot soccer. *Autonomous Robots*, 27(1):55–74, 2009.

Riedmiller, M. Neural fitted q iteration - first experiences with a data efficient neural reinforcement learning method. In *16th European Conference on Machine Learning*, pages 317–328. Springer, 2005.

Riedmiller, M. and Braun, H. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *IEEE International Conference on Neural Networks*, pages 586–591, 1993.

Rosenblatt, F. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan, 1962.

Rubin, D. Iteratively reweighted least squares. *Encyclopedia of Statistical Sciences*, 4:272–275, 1983.

Rumelhart, D. E.; Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *Nature*, 323(Oct):533–536, 1986.

Rus, D. and Tolley, M. T. Design, fabrication and control of soft robots. *Nature*, 521(7553):467–475, May 2015.

Russell, S. and Norvig, P. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, 3rd edition, 2009. ISBN 0136042597, 9780136042594.

Sallans, B. and Hinton, G. E. Reinforcement learning with factored states and actions. *Journal of Machine Learning Research*, 5:1063–1088, 2004.

Sanner, S. and Boutilier, C. Practical solution techniques for first-order mdps. *Artificial Intelligence*, 173:748–788, Apr 2009.

Schapiro, A. C.; Rogers, T. T.; Cordova, N. I.; Turk-Browne, N. B., and Botvinick, M. M. Neural representations of events arise from temporal community structure. *Nature neuroscience*, 16(4):486–492, April 2013. ISSN 1546-1726.

Schölkopf, B. and Smola, A. J. *Learning with Kernels*. MIT Press, 2001. ISBN 978-0262194754.

Schölkopf, B.; Smola, A., and Müller, K. Kernel principal component analysis. In *Artificial Neural Networks ICANN*, pages 583–588, 1997.

Schölkopf, B.; Smola, A., and Müller, K.-R. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

Schultz, W.; Dayan, P., and Montague, P. R. A Neural Substrate of Prediction and Reward. *Science*, 275(5306):1593–1599, March 1997. ISSN 1095-9203.

Schweitzer, P. J. and Seidmann, A. Generalized polynomial approximations in Markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110(2):568–582, 1985.

Seeger, M. W. Bayesian inference and optimal design for the sparse linear model. *Journal of Machine Learning Research*, 9:759–813, 2008.

Shatkay, H. and Kaelbling, L. P. Learning geometrically-constrained hidden markov models for robot navigation: Bridging the topological-geometrical gap. *Journal of Artificial Intelligence Research (JAIR)*, 16:167–207, 2002.

Shawe-Taylor, J. and Cristianini, N. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

Shen, Y.; Stannat, W., and Obermayer, K. Risk-sensitive Markov control processes. *SIAM Journal on Control and Optimization*, 51(5):3652–3672, 2013.

Simsek, Ö. and Barto, A. G. Skill characterization based on betweenness. In *Advances in Neural Information Processing Systems 21*, pages 1497–1504, 2008.

Smart, W. D. Explicit manifold representations for value-function approximation in reinforcement learning. In *International Symposium on Artificial Intelligence and Mathematics (AI&M 2004)*, 2004. URL `http://rutcor.rutgers.edu/~amai/aimath04/AcceptedPapers/Smart-aimath04.pdf`.

Smith, R.; Slef, M., and Cheeseman, P. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*. Springer-Verlag, 1990.

Smola, A. and Schölkopf, B. Sparse greedy matrix approximation for machine learning. In *Proceedings to the 17th International Conference Machine Learning*, pages 911–918, 2000.

Solway, A.; Diuk, C.; Córdova, N.; Yee, D.; Barto, A. G.; Niv, Y., and Botvinick, M. M. Optimal behavioral hierarchy. *PLoS Computational Biology*, 10(8): e1003779, 2014. doi: 10.1371/journal.pcbi.1003779.

Sprague, N. Predictive projections. In *International Joint Conference on Artificial Intelligence*, pages 1223–1229, 2009.

Sprekeler, H. On the relationship of slow feature analysis and Laplacian eigenmaps. *Neural Computation*, 23(12):3287–3302, 2011.

Sussmann, H. J. and Willems, J. C. 300 years of optimal control: from the brachystochrone to the maximum principle. *IEEE Control Systems Magazine*, 17(3): 32–44, June 1997. ISSN 02721708.

Sutton, R. S. Generalization in reinforcement learning: successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems*, pages 1038–1044, 1996.

Sutton, R. S.; Szepesvari, C., and Maei, H. R. A convergent $o(n)$ algorithm for off-policy temporal-difference learning with linear function approximation. In *Advances in Neural Information Processing Systems*, volume 21, pages 1609–1616. MIT Press, 2009.

Sutton, R.; Precup, D., and Singh, S. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112: 181–211, 1999.

Sutton, R. S. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh Intternational Conference on Machine Learning*, pages 216–224, 1990.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

Sutton, R. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

Sutton, R. and Barto, A. *Time-derivative models of pavlovian reinforcement*, pages 497–537. MIT Press, 1990.

Tenenbaum, J. B.; de Silva, V., and Langford, J. C. A global framework for nonlinear dimensionality reduction. *Science*, 290:2319 – 2323, 2000.

Tesauro, G. J. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6, 1994.

Tobia, M.; Guo, R.; Schwarze, U.; Böhmer, W.; Gläscher, J.; Finckh, B.; Marschner, A.; Büchel, C.; Obermayer, K., and Sommer, T. Neural systems for choice and valuation with counterfactual learning signals. *NeuroImage*, 89:57–69, 2014.

Tsitsiklis, J. and Van Roy, B. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, 1997.

Vapnik, V. N. *The Nature of Statistical Learning Theory*. Springer, 1995.

Venter, C. and Cohen, D. The century of biology. *New Perspectives Quarterly*, 21 (4):73–77, 2004.

Wahba, G. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, 1990.

Watkins, C. and Dayan, P. Q-learning. *Machine Learning*, 8:279–292, 1992.

Watter, M.; Springenberg, J. T.; Boedecker, J., and Riedmiller, M. A. Embed to control: A locally linear latent dynamics model for control from raw images. *CoRR*, abs/1506.07365, 2015. URL http://arxiv.org/abs/1506.07365.

Williams, R. J. and Baird, L. C. Tight performance bounds on greedy policies based on imperfect value functions. Technical Report NU–CCS-93-14, Northeastern University, Nov 1993. URL http://leemon.com/papers/1993wb2.pdf.

Wingate, D. Predictively defined representations of state. In Wiering, M. and van Otterlo, M., editors, *Reinforcement Learning: State-of-the-Art*, pages 415–439. Springer-Verlag Berlin Heidelberg, 2012.

Wipf, D. and Nagarajan, S. A new view of automatic relevance determination. In *Advances in Neural Information Processing Systems 20*, pages 1625–1632. MIT Press, 2008.

Wiskott, L. Slow feature analysis: a theoretical analysis of optimal free responses. *Neural Computation*, 15(9):2147–2177, 2003.

Wiskott, L. and Sejnowski, T. Slow feature analysis: unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, 2002.

Wookey, D. S. and Konidaris, G. D. Regularized feature selection in reinforcement learning. *Machine Learning*, 100(2-3):655–676, 2015.

Wright, S. J. Coordinate descent algorithms. *Mathematical Programming*, 151(1): 3–34, June 2015.

Xing, E. P.; Jordan, M. I.; Russell, S., and Ng, A. Y. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 505–512, 2002.

Xu, X.; Hu, D., and Lu, X. Kernel-based least squares policy iteration for reinforcement learning. *IEEE Transactions on Neural Networks*, 18(4):973–992, 2007.

Yasuda, T. and Ohkura, K. A reinforcement learning technique with an adaptive action generator for a multi-robot system. In *International Conference on Simulation of Adaptive Behavior*, pages 250–259, 2008.

Yu, L.; Jiang, Z., and Liu, K. Research on task decomposition and state abstraction in reinforcement learning. *Artificial Intelligence Review*, 38(2):119–127, 2012.

Yuan, M. and Lin, Y. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, series B*, 68:49–67, 2006.

Index

This thesis contains the following publications:

- Böhmer, W.; Grünewälder, S.; Nickisch, H., and Obermayer, K. Generating feature spaces for linear algorithms with regularized sparse kernel slow feature analysis. *Machine Learning*, 89(1-2):67–86, 2012.
  The final publication is available at Springer via
  http://dx.doi.org/10.1007/s10994-012-5300-0.

- Böhmer, W.; Grünewälder, S.; Shen, Y.; Musial, M., and Obermayer, K. Construction of approximation spaces for reinforcement learning. *Journal of Machine Learning Research*, 14:2067–2118, July 2013.
  The original article can be downloaded from
  http://jmlr.org/papers/v14/boehmer13a.html.

- Böhmer, W. and Obermayer, K. Towards structural generalization: Factored approximate planning. ICRA Workshop on Autonomous Learning, 2013. URL http://autonomous-learning.org/wp-content/uploads/13-ALW/paper_1.pdf.

- Böhmer, W.; Springenberg, J. T.; Boedecker, J.; Riedmiller, M., and Obermayer, K. Autonomous learning of state representations for control: An emerging field aims to autonomously learn state representations for reinforcement learning agents from their real-world sensor observations. *KI - Künstliche Intelligenz*, 29(4):353–362, 2015.
  The final publication is available at Springer via
  http://dx.doi.org/10.1007/s13218-015-0356-1.

- Böhmer, W. and Obermayer, K. Regression with linear factored functions. In *Machine Learning and Knowledge Discovery in Databases*, volume 9284 of Lecture Notes in Computer Science, pages 119–134. Springer International Publishing, 2015.
  The final publication is available at Springer via
  http://dx.doi.org/10.1007/978-3-319-23528-8_8.

Eidesstattliche Erklährung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Ausführungen, die anderen veröffentlichten oder nicht veröffentlichten Schriften wörtlich oder sinngemäss entnommen wurden, habe ich kenntlich gemacht.

Die Arbeit hat in gleicher oder ähnlicher Fassung noch keiner anderen Prüfbehörde vorgelegen.

Unterschrift: _____

Datum: _____

Statement of Authorship

This dissertation is the result of my own work and includes nothing, which is the outcome of work done in collaboration except where specifically indicated in the text. It has not been previously submitted, in part or whole, to any university or institution for any degree, diploma or other qualification.

Signature: _____

Date: _____

Wendelin Böhmer

Berlin, 2016