# Application of Statistical Estimation Theory, Adaptive Sensory Systems and Time Series Processing to Reinforcement Learning

vorgelegt von
Diplom Informatiker
**Steffen Grünewälder**
aus Berlin

Von der Fakultät IV-Elektrotechnik und Informatik
der Technischen Universität Berlin
zum Erlangen des akademischen Grades
**Doktor der Naturwissenschaften**
**- Dr. rer. nat. -**

genehmigte Dissertation

Promotionsausschuss:

| | |
|---|---|
| Vorsitzender: | Prof. Dr. Reinhold Orglmeister |
| Berichter: | Prof. Dr. Klaus Obermayer |
| Berichter: | Prof. Dr. Manfred Opper |

Tag der wissenschaftlichen Aussprache: 28.1.2009

Berlin 2009

D-83

## Acknowledgements

I wish to express my gratitude to Klaus Obermayer, who is the head of the NI group at TU Berlin. In his group I had a great freedom in choosing my research topics and the scientific environment he set up supported the approaching of new topics. While my approaches were often "naive" in the beginning the discussions with NI members and the "scientific approaches" that I learnt from them helped a lot to make the works successful in the end.

I want to express my special thanks to Sepp Hochreiter, who was member of the NI group during my first 2-3 years. From Sepp I learnt the most about research and scientific approaches. At that time we had daily coffee breaks and the majority of discussions took place at these breaks. The discussions and the cooperation did not had a direct impact onto publications, but they were invaluable to get on the right track.

Furthermore, I want to thank Wendelin Böhmer for his work enthusiasm with our robotics project. Wendelin worked with me at the NI student project and afterwards at the robotics project (now about 2 1/2 years !). I think our cooperation was much more on a PhD student - PhD student level and not a supervisor-student level. Very often we had rough ideas how one can approach a problem and Wendelin was able to realize the ideas with very few supervision.

Next, I would like to thank Arno Onken with whom I spent most of the time in my final year. I assisted Arno in his Copula project and tried to provide him with some background in "scientific working". Yet, I learnt a lot on my own following Arno's journey from his start to the 08 NIPS submission and decision. More than once, we were astonished by strange coincidences on this journey.

I also want to thank Yun Shen and Aki Naito. Yun worked with us at the robotics project and handled a lot of practical problems with the robot. The timetable for the robotics project was very tight, as two of three years had already been spent without much success when we took over. So both Wendelin and Yun worked with a not negligible amount of pressure. Yun, for example, was recording robot videos at night or on Sundays, when that was needed. And that just for a student job! Aki worked with me in the final year. In this year she was in a tough situation as she also needed to work roughly 5 days a week in a company. Therefore, the progress of our project was slow. Yet, Aki is working well and I have no doubts that we will successfully complete the project.

From my colleagues I want to thank first of all my roommates Claudia Sannelli and Michal Natora. I enjoyed sharing the office with both of them. Claudia switched to Klaus-Robert Müller's group, but we are still in close contact. Michal is also a great roommate and we always have a lot of fun in office.

From TU Berlin I would like to thank Marek Musial for the joint work on the robotics project and Manfred Opper for numerous discussions.

Finally, I would like to thank in particular the NI members for coffee breaks, discussions and all the other things of the daily life in a research group: Klaus Wimmer, Kamil Adiloglu, Stephan Schmitt, Roland Vollgraf, Oliver Beck, Peter Wiesing, Thomas Hoch, Falk Henrich, Johannes Jain, Michael Sibila, Daniela Pelz, Philipp Kallerhoff, Robert Martin, Felix Franke, Susanne Schönknecht, Nicolas Neubauer, Robert Annies and Marcel Stimberg.

Last I want to especially thank my parents and my wife for their great support. In particular, at the beginning of my work things were really difficult and I would not have been able to proceed without their help.

# Contents

# List of Figures

# List of Symbols

| | |
|---|---|
| $\mu_{ss'}$ | The number of direct transitions from state $s$ to $s'$. |
| $\pi$ | Path. |
| $\pi_i$ | ith state in the path. |
| $\Pi_{ss'}$ | Set of all paths from state $s$ to $s'$. |
| $\mathbf{\Pi}(\mathcal{S})$ | Set of paths that are consistent with $\mathcal{S}$. |
| $\mathbb{P}, \mathbb{E}, \mathbb{V}$ | Probability measure, expectation and variance. |
| $H_s$ | Sum of the reward received through transitions from state $s$. |
| $K_s$ | Number of visits of state $s$. |
| $\bar{p}_{ss'}$ | Estimate of the probability for a direct transition from state $s$ to $s'$. |
| $\bar{P}_{ss'}$ | Estimate of the transition probability from state $s$ to $s'$. |
| $\bar{R}_s$ | Estimator of the reward received through transitions from state $s$. |
| $\mathbb{S}$ | State space. |
| $\mathbb{A}$ | Action space. |
| $\mathbb{O}$ | Observation space. |
| $\mathcal{S}, \mathcal{T}$ | Sufficient Statistics. |
| $V_s$ | True value of state $s$. |
| $\bar{V}_s$ | Estimated value of state $s$. Concrete estimator is section dependent. |
| $\bar{V}_s^{(i)}, \bar{P}_{ss'}^{(i)}, \ldots$ | Superscripts denote values after the ith run. |
| $\mathbf{V}, \mathbf{P}, \ldots$ | Vectors and Matrices. |
| $\boldsymbol{\phi}, \boldsymbol{\psi}$ | Elements of feature space. |
| $\mathbf{\Phi}$ | Data matrix. |
| $\dot{\mathbf{\Phi}}$ | Discrete derivative of the data matrix. |

# CHAPTER 1

# Introduction

In this thesis three major topics of Reinforcement Learning (RL) are addressed: (1) The classical control and estimation problem, (2) the control task for the case that only sensory information are available and no state space representation and (3) a special non-Markov control problem, where the system needs to memorize important events. These three topics address main parts of a robotic control system. In a robotic setting no state space is available, but only sensory information and a control system needs to be able to deal with these (point 2). Furthermore, in real world setting the sensory information alone are not enough. The system needs to identify and memorize important information and actions. For example, a robot that works in a household and uses a camera for navigation will be unable to derive the "state of the house" out of the current image. He needs to remember what he did and what he observed before (point 3). Based on the sensory processing and possibly memorized information the system needs to derive a reasonable control (point 1). We treat all three topics in the simplest setting, where the topic makes "sense": Finite state space Markov Decision Processes (MDPs) for topic 1, a camera based robotic task where the system is Markovian and the sensory information are sufficient, i.e. no unobserved environmental states, for topic 2 and finite state space Partially Observable Markov Decision Processes (POMDPs) for topic 3.

In the following we provide some background information to the studies and introduce basic concepts.

## 1.1 Value Estimation for MDPs

An important task in Reinforcement Learning is the estimation of state values by means of different estimators. Natural questions which arise are: (1) Does the estimator converge to the correct value with an increasing number of examples? (2) Which estimator is better for a fixed number of examples? These are standard questions in statistical estimation theory (for example [63], [75], [48]) and they are typically addressed in a two step procedure. First, a measure for

the distance between an estimator and the true value is defined (error measure). Second, statements about the distance are derived. In this sense convergence means that the distance converges to zero with an increasing number of examples. While an estimator $A$ is better than an estimator $B$ if it has a smaller distance for a fixed number of examples. The following two subsections deal with the convergence and the finite example case.

### 1.1.1 Convergence

An important property of estimators is the convergence to the correct value with an increasing number of examples. Different kinds of convergence exist, for example point-wise, norm-wise and so on. Three important kinds of convergence in probability and estimation theory are: *convergence in probability*, *almost sure convergence (a.s.)* and *convergence in the average ($L^1$-convergence)*. A series of estimators $(A_n)$, with n being the number of examples, converges in probability to $\theta$ if

$$\lim_{n\to\infty} \mathbb{P}[|A_n - \theta| > \epsilon] = 0$$

for all $\epsilon > 0$. This means that the distance between an estimator $A_n$ and the true value will be arbitrary small for a sufficient number of training examples with probability one. An estimator $A_n$ converges almost surely if

$$\mathbb{P}[\lim_{n\to\infty} A_n = \theta] = 1.$$

That means the limes of the series of estimators differs from $\theta$ only on a set of probability 0. Almost sure convergence is similar to convergence in probability. This can be observed, by using the following alternative definition of almost sure convergence. $A_n$ converges almost surely, if

$$\lim_{n\to\infty} \mathbb{P}[\sup_{m\geq n} |A_m - \theta| > \epsilon] = 0$$

holds for all $\epsilon > 0$. Again the probability of a deviation tends to zero, however, the distance between the estimate and the true value is now measured with the supremums norm. The almost sure convergence is stronger and implies convergence in probability. The contrary does not hold in general. Intuitively the difference between these two types of convergence is that for a given $N$ for which $\mathbb{P}[|A_n - \theta| > \epsilon] < \delta$ holds it is likely to have "outliers".This is because the probability of a deviation of a single element of the series is bounded and not a set of elements, like for the almost sure convergence. Contrary, for the almost sure convergence the probability of finding an "outlier" after $N$ is bounded by $\delta$. In this sense the almost sure convergence is uniform, while the convergence in probability is not. Finally, the $L^1$-convergence is a convergence in the average. It is defined through

$$\lim_{n\to\infty} \mathbb{E}[|A_n - \theta|] = 0.$$

$\mathbb{E}$ denotes the expectation. This kind of convergence is different to the former two in the sense that we do not look at the probability of a deviation but we measure the distance directly on the space of the estimators with the $L^1$-norm. The convergence in probability follows from the $L^1$-convergence. Other implications, like $L^1 \leftrightarrow a.s.$, do not hold in general. $A.s. \rightarrow L^1$ can, however, often be

derived if a further "integrability" property is available using convergence theorems like the Lebesgue theorem. More general theorems for this direction use the concept of *uniform integrability*. A family of measurable functions $(f_i)_{i\in I}$ is *uniformly $\mu$-integrable* if for every $\epsilon > 0$ a $\mu$-integrable function $g$ exists with $\int_{|f|\geq g} |f| d\mu \leq \epsilon$ for all $f_i$. The index set must not be countable. In summary, these three kinds of convergence differ in the choise of distance measure. We will see a similar setup in the next section, where we measure the distance between an estimator and the true value in different ways (for example with the $L^2$-norm).

### 1.1.2 Comparison of Estimators

The convergence results tell us what happens for an infinite number of examples. However, in practice we want to choose the estimator which is best for a finite number of examples. So the first question which arises is: How can we say that an estimator $A$ is better than estimator $B$? One way would be to show that $A$ is always closer to the true value for any fixed number of examples. Again, this is a bit imprecise as an estimator is a random variable. To be more precise we could say that $A$ is better than $B$ if it has a higher probability to be close to the true value. Formalising this results in the following criterion: Whereas $A$ is better than $B$ if for all $\lambda_1, \lambda_2 > 0$ and parameter $\theta$

$$\mathbb{P}(\theta - \lambda_1 < A < \theta + \lambda_2) \geq \mathbb{P}(\theta - \lambda_1 < B < \theta + \lambda_2)$$

holds. $A$ is said to be optimal if this holds for all possible estimators $B$. This criterion is also called *largest possible concentration criterion* because $A$ has the highest concentration of probability mass around the true parameter [63]. Although the criterion is appealing from a theoretical point of view, it is not applicable in a practical setting [63]. Instead of this measurement for the distance one can use any of the distance measurements which are used for convergence, like $\mathbb{P}[|A - \theta| > \epsilon]$ or the distance implied by the $L^2$-norm. These distance measurements provide weaker statements then the largest possible concentration criterion but are often related to this criterion. For example, the distance implied by the squared $L^2$-norm is related in the sense that it is a necessary but not sufficient condition for the largest possible concentration criterion to hold that an optimal estimator has also the lowest squared $L^2$-norm. This distance is often used in statistical estimation theory and is named the *mean squared error (MSE)*.

$$\text{mse}(A) = \mathbb{E}[(A - \theta)^2]$$

The MSE is the expected squared-distance between the estimator and the parameter. The MSE can be decomposed in a bias and a variance part:

$$\text{mse}(A) = \mathbb{V}[A] + (\mathbb{E}[A] - \theta)^2. \tag{1.1}$$

The first term is the *variance* of the estimator and the second is the *bias*. $A$ is called *unbiased* if the bias term is zero. In the unbiased case the MSE equals the variance of the estimator and the MSE bounds the probability of a deviation of more than $\epsilon$ from the parameter:

$$\mathbb{P}(|A - \theta\,| \geq \epsilon) \leq \frac{1}{\epsilon^2}\text{mse}(A).$$

This follows from the Chebyshev inequality [7]. One should be aware of the difference between the MSE and the *empirical* or *sample mean squared error* which is often used as an error function. The MSE used here corresponds to the value obtained by averaging the empirical mean-squared-error values from an infinite number of learning tasks on a given problem. The MSE is often used for estimator comparison. See [63] for a discussion on estimator comparison. A special problem are counterintuitive effects like the following: Consider the class of constant estimators c (c does not change with data and always gives the same value). In this class an estimator with $\text{mse}(c) = 0$ exists and is hence optimal. This is the case because c has zero variance and one c equals $\theta$. However, one does not know the correct c. It is possible to overcome such strange effects by restricting the estimator class to unbiased estimators. This is a typical setup in statistics, whereas one is interested in finding the best estimator in this class (the minimum variance unbiased estimator MVU, [48]).

**Sufficient Statistics**

Naturally related to the estimation performance is the amount of information. Information about a sample is typically available through a *statistic* $\mathcal{S}$ of the data (for example $\mathcal{S} = \sum_i x_i$, where $x$ is a sample). Especially important are statistics which contain all information of a sample. Such statistics are called *sufficient*[1]. They are of high importance, as they allow us to derive the optimal unbiased estimator for a distribution if they are *minimal* and *complete*. Many different sufficient statistics exist, but the most interesting one is the one with the smallest dimension. This is the minimal sufficient statistics. Formally, suppose that a statistic $\mathcal{S}$ is sufficient for $\theta$. Then $\mathcal{S}$ is minimally sufficient if $\mathcal{S}$ is a function of any other statistic $\mathcal{T}$ that is sufficient for $\theta$. Completeness is a more abstract property, which is needed to guaranty optimallity. Formally, a statistic $\mathcal{S}$ is complete if $\mathbb{E}_\theta[h(\mathcal{S})] = 0$ for all $\theta$ implies $h = 0$ almost surely. The famous theorem from *Rao and Blackwell* [75] states that for a complete and minimal sufficient statistics $\mathcal{S}$ and any unbiased estimator $A$ of a parameter $\theta$ the estimator $\mathbb{E}[A|\mathcal{S}]$ is the optimal unbiased estimator with respect to any convex loss function and hence the optimal unbiased estimator for the MSE. The parameter can also be vector valued, in which case the minimal statistics typically has the same size as the parameter vector. Furthermore, the theorem holds, if a function of the parameters is estimated (for example $A$ estimates $f(\theta)$). We show the two main properties of the theorem (unbiased and that $\mathbb{E}[A|\mathcal{S}]$ has a lower loss than $A$), as they are short to derive, give some intuition and because the theorem is often stated only for the parameters, but not for functions of the parameters. The new estimator is unbiased because $A$ is unbiased,

$$\mathbb{E}[\mathbb{E}[A|\mathcal{S}]] = \mathbb{E}[A] = f(\theta).$$

The new estimator has a lower convex loss. This follows from the *Jensen*-inequality for the conditional expectation [7]. Let $l$ be a convex loss function, the loss of $\mathbb{E}[A|\mathcal{S}]$ is given by

$$\mathbb{E}[l(\mathbb{E}[A|\mathcal{S}]-f(\theta))] = \mathbb{E}[l(\mathbb{E}[A-f(\theta)|\mathcal{S}])] \overset{\text{Jensen}}{\leq} \mathbb{E}[\mathbb{E}[l(A-f(\theta))|\mathcal{S}]] = \mathbb{E}[l(A-f(\theta))].$$

[1] It is distribution dependent which statistics contain all information. A statistics might be sufficient for one distribution and not sufficient for another.

## 1.2 The Variance of a MDP and Variance based Cost Functions

An important measure of uncertainty and of risk is the variance of a MDP. An equation of the variance was first derived from Sobel [72]. I started to study the variance of a MDP in winter 2006. I wanted to submit a workshop paper to a PASCAL workshop about the exploration-exploitation problem. I thought the variance might be a "nice" measure for estimator uncertainty. I did not had too much time for the submission and, therefore, in my literature study I did not stumble over the work of Sobel and the follow ups. So I rederived the variance used it for the workshop paper and submitted a long version to ICML. There, one out of three reviewers was actually aware of the work of Sobel and told me to cite the paper instead of rederiving the equations. Yet, the other two reviewers did not know the work. I think this is a typical phenomena in Reinforcement Learning. There are many overlaps with other fields and the community is often unaware of important contributions. Respectively, for students like myself it is hard to have the needed broad overview. I think this problem applies in particular to Reinforcement Learning as a "standard text book" summarizing and unifying the field is missing.

Being unaware of the work of [72] was particularly annoying, as he did not only derived the variance but he also showed with an example that the variance is not useful for optimization as it does not fulfill the so called *monotonicity* property. Intuitively, monotonicity means that a policy $\pi$ which has a higher variance than a policy $\pi'$ for successor states of a state $s$ has also a higher variance for state $s$, given that in state $s$ for $\pi$ and $\pi'$ the same action is chosen. I found the example in Sobel not that intuitive and want to give here a different example, emphasising one of the factors that is responsible for this effect. Some basic properties of MDPs are used in this example. Readers unfamilar with MDPs may consult the preliminaries section from Chapter 2 (2.2).

**Monotonicity Example** We use a MDP with 5 states with transitions $1 \to 2, 1 \to 3, 2 \to 4, 2 \to 5, 3 \to 4, 3 \to 5$. We want to decompose the variance of state 1 into the variance parts of state 2 and 3 and a further "covariance" term. We simplify the analysis by using a reward of 0 for the transitions from state 1 to state 2 and 3 and a discount factor of $\gamma = 1$. In the following we denote the second moment with $M$ and the first moment (the value) with $V$. The variance $\mathbb{V}$ of state 1 is given by

$$\begin{aligned}\mathbb{V}_1 =& p_{12}(p_{24}r_{24}^2 + p_{25}r_{25}^2) + p_{13}(p_{34}r_{34}^2 + p_{35}r_{35}^2) \\ & - (p_{12}p_{24}r_{24} + p_{12}p_{25}r_{25} + p_{13}p_{34}r_{34} + p_{13}p_{35}r_{35})^2 \\ =& p_{12}M_2 + p_{13}M_3 - (p_{12}V_2)^2 + 2p_{12}p_{13}V_2V_3 + (p_{13}V_3)^2 \\ =& p_{12}M_2 + p_{13}M_3 - (p_{12}V_2 + p_{13}V_3)^2.\end{aligned}$$

Now the problem becomes obvious. The normalization is not the right one. Instead of $p_{12}V_2^2 + p_{13}V_3^2$ we have the term $(p_{12}V_2 + p_{13}V_3)^2$. If we reformulate this with variance terms, we get:

$$\mathbb{V}_1 = p_{12}\mathbb{V}_2 + p_{13}\mathbb{V}_3 + p_{12}p_{13}(V_2 - V_3)^2.$$

So the variance of state 1 does not only depend on the variance of state 2 and 3 but also on the difference in value of state 2 and 3. This effect is problematic as we cannot define optimization problems using directly the variance. For example, if we have a policy $\pi$ with $\mathbb{V}_2 = \mathbb{V}_3 = 0$, $V_2 = 100, V_3 = -100$ and a policy $\pi'$ with $\mathbb{V}_2 = \mathbb{V}_3 = 1$ and $V_2 = 0, V_3 = 0$ then policy $\pi$ has a higher variance for states 2 and 3 but a lower variance for state 1 than $\pi'$. If we want to maximize the variance in state 1 then we need to chose policies in state 2 and 3 that does not achieve the maximal variance in these states.

In operations research different approximations are known which replace the variance with alternative measures. We summarize different approximations and corresponding "Mean-Variance Tradeoffs" in Chapter 3.

## 1.3 Function Approximation and Control

Another major topic of my thesis is a camera based robotic control system. We addressed the control problem with Reinforcement Learning methods that are constructed for the function approximation setting, i.e. for the case that no discrete state space is given but only sensory signals. This setting is currently quiet popular in Reinforcement Learning and in my opinion will become even more popular in the future. The reason for this is that we are currently at the point where computers are fast enough to apply the Reinforcement Learning methods to "real-world" problems. The robotic task we are studying is still a simple one, but its a huge step forward compared to old Reinforcement Learning tasks of the 90's like for example pole balancing. Certainly, there were already good applications at that time like the TD-Gammon. Yet, I think a really huge field for applications can be explored in the future using robots. And our work is in this sense a first step in this direction.

Working on this robotic task yielded to us a lot of insight knowledge for the methods and the problem. One particular interesting insight was for me that in control tasks it is possible to construct "functions on the environment" with unsupervised procedures. Usually, one wants to have *independent and identical distributed (iid)* examples. In a control setting the examples are not iid. This certainly introduces problems, yet it also allows us to construct "more powerful" unsupervised algorithms. For example, the Slow Feature Analysis we used in our work constructs trigonometric functions "on the environment". Using these functions in combination with a linear value estimator is similar to performing a Fourier expansion. And the resulting method inherits all the nice approximation properties of the Fourier expansion. I think the Fourier expansion is just a first step towards a new set of unsupervised methods for control tasks. Approximation theory, for example, provides us with a broad number of "well" structured function spaces that are useful for approximating functions.

## 1.4 Vanishing Gradient and Reinforcement Learning

In the final two chapter of the thesis we discuss *vanishing gradient* related topics and an application to Reinforcement Learning. Both are smaller works, but in

my opinion the whole topic has a big future potential.

To explain here shortly the basic idea about the vanishing gradient problem: Assume that we have a dynamical (differentiable) system with a dynamics described by $f$ and a Jacobian $J$ of the dynamical system, where $J_{ij} := \frac{\partial s_{t,i}}{\partial s_{t-1,j}}$ and $s_t$ is the $n$-dimensional state of the system at time $t$. The Jacobian is a measure for the influence of the state of the system at time $t-1$ to the new state at time $t$. If the Jacobian is in a given matrix norm smaller than 1 for each possible state $s$, i.e. $\sup_s ||J_s|| \leq c < 1$ then the influence of the state $s_0$ onto the state $s_t$ vanishes exponentially fast due to the chain rule:

$$||D_{s_0} f^{(t)}|| = ||D_{f^{(t-1)}s_0} f D_{f^{(t-2)}s_0} f \dots D_{s_0} f|| \leq \sup_s ||D_s f||^n = \sup_s ||J_s||^n \leq c^n,$$

where $f$ describes the system evolution, $f^{(t)}$ denotes the $t$-times application of $f$ and $D_s$ denotes the differential at point $s$.

This exponential information loss has a tremendous practical impact. Learning systems with a norm smaller 1 are hardly able to relate signals that are more than a few steps apart. For a control setting this means, for example, that important old actions which does not influence the observable state but only the true (unobservable) system state are extremly hard to detect. We applied in our last chapter a special Hidden-Markov Model from Sepp Hochreiter which circumvents the problem. It is a smaller work, which is partly due to the fact that the model worked "out of the box". For me this was quiet surprising: Usually, if you want to apply a system to a new setting then at least a hand full of unexpected and not directly solvable problems appear for which you need to modify and adapt the original system. In this case the needed modifications were "minor".

## 1.5 Contributions and Outline

I separated the thesis into three parts. Each part corresponds to another stochastical model:

1. Discrete Markov Decision Processes (MDPs).
2. A continuous state space system with sensory observations.
3. Partially Observable Markov Decision Processes (POMDPs).

It might be more natural to first treat the discrete case (MDPs and POMDPs) and then move on to the continuous setting. My motivation was here the relevance of the different studies. In my opinion the MDP studies are the most significant with a couple of open question that can be further explored. After that I find the continuous setting very important. Currently, there is a lot of interest in this setting and I think our study provides an interesting new approach. Finally, the POMDP studies are smaller works and no main stream topics. However, I think the combination of methods from time-series analysis and control problems is a good and interesting approach. And I am quite sure that the problems considerd in part 3 will become popular in the near future.

In *Chapter 2* we study value estimation in MDPs. Most of the previous works in RL were concerned with asymptotic properties of estimators, i.e. does

an estimator converge to the value. In practice, however, only a finite number of samples is available and it is important to know which estimators are best in this situation. We addressed this question with methods from statistical estimation theory. We applied the Rao-Blackwell theorem to *derive the optimal unbiased value estimator (Minimum Variance Unbiased estimator, MVU)* and we compared it to three well known estimators from RL: Temporal Difference Learning (TD), Monte Carlo estimation (MC) and Least Squares Temporal Difference Learning (LSTD). In particular, we demonstrate that the *LSTD estimator is equivalent to the MVU for acyclic Markov Reward Processes (MRPs)* and show that both differ for cyclic Markov Reward Processes as *LSTD is then typically biased*. In general, estimators that fulfill the Bellman equation, such as LSTD, cannot be unbiased. *Two main factors that separate unbiased estimators and estimators that fulfill the Bellman equation, are identified: 1) the discount and 2) a normalization.* Furthermore, we show that *in the undiscounted case the MC estimator is equivalent to the MVU and to the LSTD estimator if the same amount of information is available to MC. In the discounted case this equivalence does not hold anymore.* For the case of TD we show that *TD gets unbiased with a minor modification for acyclic MRPs.* In general, the TD update rule "moves" the estimate towards the LSTD solution. Consequently, TD is biased for cyclic MRPs. Finally, counter-examples are presented to show that *no general ordering exists between the MVU and the LSTD estimator, between MC and LSTD and between TD and MC.* Theoretical results are supported by examples and an empirical evaluation.

In *Chapter 3* we addressed the cost function which is optimized. It is common in RL to choose strategies that maximize the mean without regarding the corresponding variance. This strategy, however, is undesirable in many cases as the derived policies are unreliable and the performance may have high positive and negative peaks causing fatal effects (e.g. the bankruptcy of a trading company). In operations research several variations of mean-variance tradeoffs are used to control the variance and thus, the risk. The objectives have the drawback that a policy iteration cannot be applied and the optimization is slow, leading to problems for large state spaces. We overcame this problem by using a *first order Taylor approximation to transform the objective to a value maximization objective with modified rewards.* Furthermore, *we derived estimators for the variance of a policy* which can be used for evaluation of the associated risk. We compared the objectives and the methods in different experimental settings. We used two tasks from the series of paper by D. J. White on MDPs with real-world applications for the evaluation.

In *Chapter 4* we present *adaptive control system for navigating a camera equipped robot.* The main feature of the system is the visual processing which generates a Fourier expansion on the area in which the robot acts. In our approach the system for visual processing is learned from examples of the concrete environment using a Slow Feature Analysis and not "hard-coded". The control is learned by applying the Least-Squares Policy Iteration and the Least-Squares Temporal Difference Value estimator onto the learned visual features. The combination of the linear value estimator and the features corresponding to the elements of the Fourier expansion turns out to be a powerful tool to approximate value functions. The navigation capability of the control system is demonstrated and analyzed on a real robot and in simulated environments. In addition, we derived a *Kernel Slow Feature Analysis* to increase the capability

of the preprocessing.

In two smaller studies (*Chapter 5 and 6*) we studied the so called *vanishing gradient* problem which hinders the detection of long term dependencies in time series data. Most systems are affected by this problem and are unable to detect dependencies which are more than a few steps apart. A system called *Long Short Term Memory (LSTM)* has been proposed to solve the problem. The key element of the LSTM is a "volume conserving" unit that prevents the system from losing information. In the first study we modified the LSTM with an attention mechanism. This study was done with an emphasis on cognitive science and the goal was to address critical points which make the processing of the structure differ from human processing. In particular, we were interested in the encoding and the storage: (1) The LSTM structure is built to massively store information instead of carefully selecting few input signals for storage in memory. (2) Reweighting of stored information due to changing constellations is not possible. In this work we tackled these points through introducing an attention mechanism which drives the encoding and the storage of the structure. The mechanism "weakens" the volume conservation in that way that it allows to reweight stored information. Yet, the vanishing gradient problem is circumvented by lower bounding the weighting factors.

In the second study we applied a state space based system called *Memory Input-Output Factorial Hidden Markov Model (MIOFHMM)* to POMDPs. Similarly to the LSTM the MIOFHMM solves the vanishing gradient problem. In particular, we addressed in this part POMDPs where observations are available, which contain information about the true MDP state. In simulations we showed that a MIOFHMM based system is able to resolve MDPs even if a critical observation is about 100 000 steps before a related effect (T-maze task). In comparison, other methods can at maximum handle up to 60-70 steps.

Each chapter starts with an abstract and a short introduction followed by chapter specific preliminaries.

# Part I

# Control in Markov Decision Processes

CHAPTER 2

# The Optimal Unbiased Value Estimator and its Relation to LSTD, TD and MC


## Abstract

In this analytical study we derive the optimal unbiased value estimator (Minimum Variance Unbiased estimator, MVU) and compare its statistical risk to three well known value estimators: Temporal Difference learning (TD), Monte Carlo estimation (MC) and Least-Squares Temporal Difference Learning (LSTD). In particular, we demonstrate that the LSTD estimator is equivalent to the MVU for acyclic Markov Reward Processes (MRPs) and show that both differ for cyclic MRPs as LSTD is then typically biased. In general, estimators that fulfill the Bellman equation, such as LSTD, cannot be unbiased. Two main factors that separate unbiased estimators and estimators that fulfill the Bellman equation, are identified: 1) the discount and 2) a normalization. Furthermore, we show that in the undiscounted case the MC estimator is equivalent to the MVU and to the LSTD estimator if the *same amount of information* is available to MC. In the discounted case this equivalence does not hold anymore. For the case of TD we show that TD gets unbiased with a minor modification for acyclic MRPs. In general, the TD update rule "moves" the estimate towards the LSTD solution. Consequently, TD is biased for cyclic MRPs. Finally, counter-examples are presented to show that no general ordering exists between the MVU and the LSTD estimator, between MC and LSTD and between TD and MC. Theoretical results are supported by examples and an empirical evaluation.


## 2.1 Introduction

One of the important theoretical issues in reinforcement learning are rigorous statements on convergence properties of so called *value estimators* (e.g. [77], [80], [44], [19]) which provide an empirical estimate of the expected future reward for every given state. So far most of these convergence results were restricted

to the asymptotic case and did not provide statements about the deviation of the estimate from the true value for the case of a finite number of observations. In practice, however, one wants to choose the estimator which yields the best result for a given number of examples or in the shortest time.

Current approaches to the finite example case are mostly empirical and few non-empirical approaches exist. [49] present upper bounds on the generalization error for *Temporal Difference estimators (TD)*. They use these bounds to formally verify the intuition that TD methods are subject to a "bias-variance" trade-off and to derive "schedules" for estimator parameters. Comparisons of different estimators with respect to the bounds were not performed. The issue of *bias and variance* in reinforcement learning is also addressed in other works ([70], [55]). [70] provide analytical expressions of the *mean squared error (MSE)* for various *Monte Carlo (MC)* and TD value estimators. They further provide a software that yields the exact mean squared error curves given a complete description of a *Markov Reward Process (MRP)*. The method can be used to compare different estimators for concrete MRPs and concrete parameter values. But it is not possible to prove general statements with this method. The most relevant works for our analysis are provided by [55] and by [69].

In [55] the bias and the variance in value function estimates is studied and closed-form approximations are provided for these terms. The approximations are used in a large sample approach to derive asymptotic confidence intervals. The underlying assumption of normally distributed estimates is tested empirically on a dataset of a mail-order catalog. In particular, a Kolmogorov-Smirnov test was unable to reject the hypothesis of normal distribution with a confidence of 0.05. The value function estimates are based on sample mean estimates of the MRP parameters. The parameter estimates are used in combination with the value equation to produce the value estimate. Different assumptions are made in the paper to simplify the analysis. A particularly important assumption is that the number of visits to a state is fixed. Under this assumption the sample mean parameter estimates are unbiased and the application of the value equation results in biased estimates. We show that without this assumption the sample mean estimates underestimate the parameters in the average and the value estimates can therefore be unbiased in special cases. We address this point in detail in Section 2.3.4.

In [69] different kinds of eligibility traces are introduced and analyzed. It is shown that TD(1) is unbiased if the *replace-trace* is used and that it is biased if the usual eligibility trace is used. What is particularly important for our work is one of their side findings: The Maximum Likelihood and the MC estimates are equivalent in a special case. We characterize this special case with Criterion 3 (p. 24) and we make frequent use of this property. We call the criterion the *Full Information Criterion* because all paths that are relevant for a value estimator in a state $s$ must hit this state (For details see p. 24).

In this work we follow a new approach to the finite example case using tools from statistical estimation theory (e.g. [75]). Rather than relying on bounds, on approximations, or on results to be recalculated for every specific MRP this approach allows us to derive general statements. Our main results are sketched in Figure 2.1. A major contribution is the derivation of the optimal unbiased value estimator (*Minimum Variance Unbiased estimator (MVU)*, Sec. 2.3.3). We show that the *Least-Squares Temporal Difference estimator (LSTD)* from [19] is equivalent to the *Maximum Likelihood value estimator (ML)* (Sec. 2.3.4)

and that both are equivalent to the MVU if the discount $\gamma = 1$ (undiscounted) and the Full Information Criterion is fulfilled or if an acyclic MRP is given (Sec. 2.3.4). In general the ML estimator differs from the MVU as estimators that fulfill the Bellman equation such as ML cannot be unbiased. The main reasons are the discount and a normalization (Sec. 2.3.1). In the undiscounted case the ML estimator approaches the MVU exponentially in the number of observed paths. In general, neither the ML nor the MVU estimator are superior to each other (Appendix 2.B.5). For the *first-visit MC* estimator it is known that the estimator is unbiased and the first-visit MC estimator is therefore inferior to the MVU. However, we show that for $\gamma = 1$ the MC estimator is equivalent to the MVU if the Full Information Criterion applies (Sec. 2.3.5). Finally, we compare these estimators to TD($\lambda$). We show that TD($\lambda$) gets unbiased through a minor modification for acyclic MDPs (Appendix 2.A) and is thus inferior to the MVU and to the ML estimator. In the cyclic case the bias of TD is related to the bias of ML (Sec. 2.3.6).

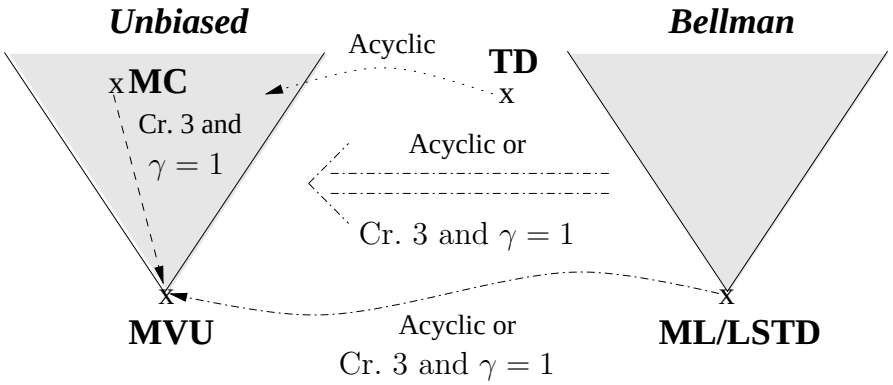


Figure 2.1: The figure shows two classes of estimators: Unbiased estimators and estimators that fulfill the Bellman equation. The best estimators (MVU and ML/LSTD) are drawn at the bottom. The arrows represent relations between the classes and between individual estimators. Different properties are needed for these relations. These properties are written above or below an arrow and they are highlighted with the line styles of the arrows.

For the sake of readability proofs are presented in Appendix 2.B.

## 2.2 Estimation in Reinforcement Learning

Reinforcement learning methods typically consist of a value estimation and a policy update step (value/policy iteration, [76]). A common assumption underlying the value estimation is that the environment can be described by a Markov Decision Process (MDP). This assumption allows us to improve the estimation performance beyond the performance of general estimators, such as the sample mean (Monte Carlo) estimator. The best known estimator, which uses the Markov structure to collect information that is "invisible" to general estimators,

is the temporal difference estimator. Despite this principal advantage of TD, simple cases exist where MC outperforms TD.

In our work, we focus on systems which are modeled as Markov Reward Processes (MRP). The difference between an MRP and an MDP is that there is only one action in MRPs. Therefore, there is only one policy for the MRP case and "learning" is thus restricted to the estimation of the value function. MDPs with fixed policies (e.g. no online update) can be described with MRPs as it is possible to account for the effect of nondeterministic policies through the transition distribution of the MRP.

### 2.2.1 Markov Reward Processes and Value Estimators

A Markov Reward Process (MRP) consists of a state space $\mathbb{S}$ (in the following we will consider a finite state space), starting probabilities $p_i$ for the initial states, transition probabilities $p_{ij}$ and a random reward $R_{ij}$ between states $i$ and $j$. In principle, the reward can have an arbitrary distribution such as multinomial or normal. The statements we derive are valid at least for deterministic rewards as well as for multinomial distributed rewards. For arbitrary reward distributions they do not hold. For example, distributions exist for which no unbiased estimators exists for the mean and thus no unbiased estimator exists for the corresponding value function. However, this is a property of the reward distribution and not of the MRP. Therefore, we will not address this topic any further.

Our goal is to estimate the value $V_i$ of each state $i$, i.e. the expected future reward received after visiting the state. This value function is given by

$$V_i = \sum_{j \in \mathbb{S}} p_{ij} \big( \mathbb{E}[R_{ij}] + \gamma V_j \big),$$

respectively in vector notation by

$$\mathbf{V} = \sum_{t=0}^{\infty} \gamma^t \mathbf{P}^t \mathbf{r} = (\mathbf{I} - \gamma \mathbf{P})^{-1} \mathbf{r},$$

where $\mathbf{P} = (p_{ij})$ is the transition matrix of the Markov process, $\mathbf{I}$ the identity matrix, $\gamma$ a discount factor and $\mathbf{r}$ is the vector of the expected one step reward ($\mathbf{r}_i = \sum_{j \in \mathbb{S}} p_{ij} \mathbb{E}[R_{ij}]$).

We compare different value estimators with respect to their risk (not the empirical risk)

$$\mathbb{E}[\mathcal{L}(\bar{V}_i, V_i)],$$

where $\bar{V}_i$ is the value estimator of state $i$ and $\mathcal{L}$ is the loss function, which penalizes the deviation from the true value $V_i$. We will mainly use the mean squared error

$$\mathrm{MSE}[\bar{V}_i] := \mathbb{E}[(\bar{V}_i - V_i)^2], \tag{2.1}$$

which can be split into a *bias* and a *variance* term

$$\mathrm{MSE}[\bar{V}_i] = \underbrace{\mathbb{V}[\bar{V}_i]}_{\text{Variance}} + (\underbrace{\mathbb{E}[\bar{V}_i - V_i]}_{\text{Bias}})^2.$$

The MSE used here corresponds to the value obtained by averaging the empirical mean-squared-error values from an infinite number of learning tasks on a given problem. An estimator is called *unbiased* if the bias term is zero.

In Reinforcement Learning it is not unbiasedness with respect to $\mathbb{E}[\cdot]$ that is important, but it is conditional unbiasedness on the event $\{N_i \geq 1\}$. Here, $N_i$ denotes the number of times state $i$ is visited. In other words, one is interested in estimators that are on average correct, given that at least one example is available for estimation.

We conclude this section by introducing a simple criterion for a class of estimators. With this criterion it is possible to verify unbiasedness and minimal MSE in this class. This criterion provides an intuitive interpretation of a weakness of the TD($\lambda$) estimator. Let $x_i, i = 1, \ldots, n$ be a sample consisting of $n \geq 1$ *independent and identically distributed (iid)* elements from an arbitrary distribution (e.g. reward sequences of an MRP). The estimator

$$\sum_{i=1}^{n} \alpha_i x_i, \quad \text{with} \quad 0 \leq \alpha_i \leq 1, \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i = 1, \tag{2.2}$$

is unbiased and has the lowest variance for $\alpha_i = 1/n$ [75]. This means that all examples should have an equal influence to achieve optimal performance.

### 2.2.2 Temporal Difference Learning

A commonly used value estimator for MRPs is the TD($\lambda$) estimator [77]. It converges on average ($L^1$-convergence, [77]) and it converges almost surely to the correct value ([80], [44]). In practical tasks it seems to outperform the MC estimator with respect to convergence speed and its computational costs are low. Analyses for the TD(0) estimator are often less technical. We therefore restrict some statements to the TD(0) case. The TD(0) estimator is given by

$$\bar{V}_s^{(i+1)} = \bar{V}_s^{(i)} + \alpha_{i+1}(R_{ss'}^{(i+1)} + \gamma \bar{V}_{s'}^{(i)} - \bar{V}_s^{(i)}), \tag{2.3}$$

where $\alpha_{i+1}$ is the learning rate, $\bar{V}_s^{(i)}$ is the estimated value for state $s$ after the $i$th transition, $s'$ is the successor state of $s$, $\gamma$ is the discount factor, and $R_{ss'}^{(i+1)}$ is the reward which occurred during the transition from $s$ to $s'$. The general TD($\lambda$) update equation is given by

$$\bar{V}_s^{(i+1)} = \bar{V}_s^{(i)} + \Delta\bar{V}_s^{(i+1)} \quad \text{and} \quad \Delta\bar{V}_s^{(i+1)} = \alpha_{i+1}(R_{ss'}^{(i+1)} + \gamma \bar{V}_{s'}^{(i)} - \bar{V}_s^{i}) e_s^{(i+1)},$$

where $\alpha_i$ is the learning rate in sample path $i$ (the learning rate might be defined differently) and $e_s^{(i+1)}$ is an *eligibility trace*. The update equation can be applied after each transition (*online*), when a terminal state is reached (*offline*) or after an entire set of paths has been observed (*batch update*). The eligibility trace can be defined in various ways. Two important definitions are the *accumulating trace* and the *replacing trace* [69]. In [69] it is shown that for $\lambda = 1$ the TD($\lambda$) estimator corresponding to the accumulating trace is biased while the one corresponding to the replacing trace is unbiased. The replacing trace is defined by

$$e_s^{(i+1)} = \begin{cases} 1 & \text{if } s = t \\ \gamma\lambda & \text{else .} \end{cases} \tag{2.4}$$

In the acyclic case both definitions are equivalent. For $\lambda < 1$ the estimators are biased towards their initialization value. However, a minor modification is sufficient to delete the bias for acyclic MRPs (Appendix 2.A). In the following we will use this modified version.

### 2.2.3 Monte Carlo Estimation

The Monte Carlo (MC) estimator is the sample mean estimator of the future reward. In [76] it is defined as $1/n \sum_{i=1}^{n} \text{Return}(i)$, where $n$ is the number of paths that have been observed and $\text{Return}(i)$ the cumulated future reward of run $i$. In the cyclic case there are two alternative MC estimators: First-visit MC and every-visit MC. First-visit MC makes one update per sampled path. It uses the part of the path which follows upon the first visit of the relevant state. Every-visit MC makes an update for each visit of the state. This leads to an overlap in the paths used for training. The every-visit MC estimator is equivalent to TD($\lambda$) for the accumulate trace and the first first-visit MC estimator for the replace trace if $\lambda = 1$ and $\alpha_i = 1/i$. The first-visit MC estimator is unbiased while the every-visit MC estimator is biased in the cyclic case [69]. Both estimators converge almost surely and on average to the correct value.

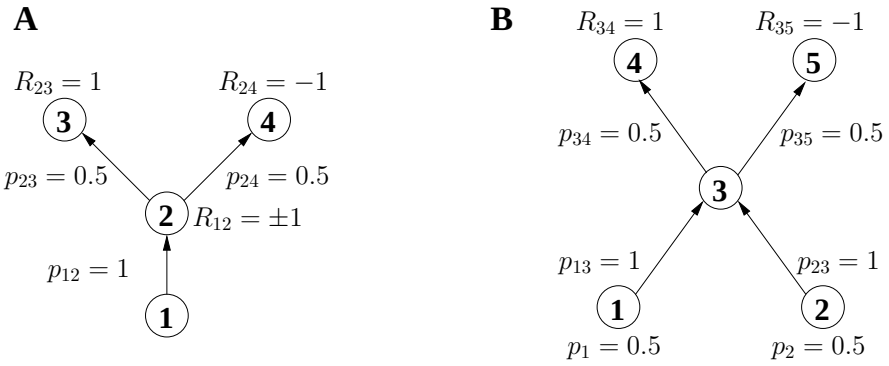


Figure 2.2: **A**: An MRP for which TD is inferior to MC. The transition from state 1 to state 2 is followed by a reward $R_{12} = +1$ and $R_{12} = -1$ with probability $p = 0.5$ each. **B**: An MRP for which MC is inferior to TD. No reward is received for transitions $1 \to 2$ and $1 \to 3$. $p_1$ and $p_2$ are the probabilities to start in state 1 and 2.

### 2.2.4 Comparison of TD and MC

Although TD is often preferred to the MC estimator, there is no guarantee that TD will converge faster. Using the mean squared error (eq. 2.1) as a performance criterion we will construct two MRPs: One for which $\text{MSE}[MC] < \text{MSE}[TD]$ (Section 2.2.4), and one for which $\text{MSE}[TD] < \text{MSE}[MC]$ (Section 2.2.4). Hence, cases exist in which estimators that use the Markov structure (here TD) are worse than estimators that do not use the Markov structure

(here MC) even if the learning rate is chosen optimally. This motivates the theoretical analysis of Section 2.3, where we derive the optimal unbiased value estimator which makes extensive use of the Markov structure.

### MSE of MC is Smaller than MSE of TD

Figure 2.2 (A) shows an example for which the MC estimator is superior. We assume that the learning rate $\alpha_i$ of TD(0) is between 0 and 1, that the learning rate in the first step is 1 ($\alpha_1 = 1$) and that the estimator is initialized to 0 (we use this assumption for readability, it is also possible to use the unbiased TD(0) estimator (Modification 2)). Let state 1 be the starting state, $n$ be the number of observed paths and let $\gamma = 1$ for simplicity.

The MC estimator for state 2 is $1/n \sum_{i=1}^{n} Y_i$, where $Y_i = R_{23}$ or $Y_i = R_{24}$ are the rewards received after a transition from state 2 to state 3 or 4. For state 1 we obtain $1/n \sum_{i=1}^{n} (Y_i + R_{12}^{(i)})$, where the $Y_i$ are the same as before and $R_{12}^{(i)}$ is the received reward after a transition to state 2. The MC estimator is a weighted average of the examples and it is the optimal unbiased linear estimator (eq. 2.2) as $\alpha_i = 1/n$ for all $i$.

We now analyze the TD(0) estimator. Consider two different sequences $\alpha_i$ and $\tilde{\alpha}_i$, $i = 1, \ldots, n$, of learning rates for the TD(0) estimators $\bar{V}_1$ and $\bar{V}_2$. The TD(0) estimator $\bar{V}_2$ can be written as (Lemma 12, Appendix 2.A)

$$\bar{V}_2^{(n)} = \sum_{i=1}^{n} \Big( \tilde{\alpha}_i \prod_{j=i+1}^{n} (1 - \tilde{\alpha}_j) \Big) Y_i =: \sum_{i=1}^{n} \tilde{\beta}_i Y_i.$$

The estimator is unbiased and has minimal variance if, and only if, $\tilde{\beta}_i = 1/n$. This can be enforced by choosing $\tilde{\alpha}_i = 1/i$. For state 1 we obtain

$$\begin{aligned} \bar{V}_1^{(n)} &= \sum_{i=1}^{n} \beta_i (\bar{V}_2^{(i-1)} + R_{12}^{(i)}) = \sum_{i=1}^{n} \beta_i \Big( \sum_{j=1}^{i-1} \tilde{\beta}_j Y_j + R_{12}^{(i)} \Big) \\ &= \Big( \sum_{i=1}^{n} \beta_i R_{12}^{(i)} \Big) + \Big( \sum_{i=1}^{n-1} (\tilde{\beta}_i \sum_{j=i+1}^{n} \beta_j) Y_i \Big) =: \Big( \sum_{i=1}^{n} \beta_i R_{12}^{(i)} \Big) + \Big( \sum_{i=1}^{n-1} \gamma_i Y_i \Big), \end{aligned} \tag{2.5}$$

where $\beta_i = \alpha_i \prod_{j=i+1}^{n} (1 - \alpha_j)$. Using the Bienaymé equality (e.g. [7]) the variance of the estimator takes the following form

$$\mathbb{V}(\bar{V}_1^{(n)}) \overset{ind}{=} \mathbb{V}(\sum_{i=1}^{n} \beta_i R_{12}^{(i)}) + \mathbb{V}(\sum_{i=1}^{n-1} \gamma_i Y_i) \overset{iid}{=} \mathbb{V}(R_{12}^{(1)}) \sum_{i=1}^{n} \beta_i^2 + \mathbb{V}(Y_1) \sum_{i=1}^{n-1} \gamma_i^2,$$

where "ind" abbreviates "independence". $Y_1$ and $R_{12}^{(1)}$ have the same variance. With $\gamma_n = 0$

$$\mathbb{V}(\bar{V}_1^{(n)}) = \mathbb{V}(Y_1)(\sum_{i=1}^{n} \beta_i^2 + \sum_{i=1}^{n} \gamma_i^2).$$

Because $0 \leq \beta_i, \gamma_i \leq 1$ and $\sum_{i=1}^{n} \beta_i = \sum_{i=1}^{n} \gamma_i = 1$ (see Appendix 2.A) this term would be minimal if, and only if, $\beta_i = \gamma_i = 1/n$. From $\beta_i = \tilde{\beta}_i = 1/n$, however, it follows that $\gamma_i = 1/n \sum_{i+1}^{n-1} 1/n = (n - i - 2)/n^2 \neq 1/n$. Hence

optimality cannot be achieved. Since both MC and TD are unbiased, we obtain $\text{MSE}[MC] < \text{MSE}[TD]$.

This example demonstrates a major weakness of TD, namely that it is impossible for TD to weight the observed paths equally, even for simple MRPs. We will show in Section 2.3.5 that MC is actually the optimal unbiased estimator for this MRP and thus it also holds that $\text{MSE}[MC] \leq \text{MSE}[TD(\lambda)]$ for each $\lambda$.

**MSE of TD is Smaller than MSE of MC**

Figure 2.2 (B) shows an example where TD(0) is superior. Let the number of observed paths be $n = 2$ and $\gamma = 1$. The value of all states is zero. TD(0) and MC are unbiased for this example. The variance of the MC estimator for states 1,2 and 3 is therefore given by

$$\begin{aligned}
\mathbb{E}[\bar{V}_3^2] =& \mathbb{P}[R^{(1)} = 1, R^{(2)} = 1] \cdot 1^2 + \mathbb{P}[R^{(1)} = -1, R^{(2)} = -1] \cdot (-1)^2 \\
&+ \mathbb{P}[R^{(1)} = 1, R^{(2)} = -1] \cdot 0 + \mathbb{P}[R^{(1)} = -1, R^{(2)} = 1] \cdot 0 \\
=& (1/4)1^2 + (1/4)(-1)^2 + (2/4) \cdot 0 = 1/2 \\
\mathbb{E}[\bar{V}_1^2] =& \mathbb{E}[\bar{V}_2^2] = (1/4)1/2 + (1/2)1 + 0 = 5/8,
\end{aligned}$$

where $R^{(i)}$ denotes the received reward in run $i$. The first term in the second line results from starting two times in state 1 or 2 and the second term in the second line from a single start in state 1. Setting the learning rate $\alpha_i$ to $\alpha_i = \frac{1}{i}$ for TD, the estimator for state 3 is equivalent to the corresponding MC estimator and therefore the variance is 1/2. In the first run the standard TD(0) update rule uses the initialization value of state 3 to calculate the estimate in state 1 or 2. This is advantageous and results in a variance of 1/2. Without exploiting this advantage the variance is 17/32. This is still lower than the variance of the MC estimator. Since both estimators are unbiased we obtain $\text{MSE}[TD(0)] < \text{MSE}[MC]$.

## 2.3 Comparison of Estimators: Theory

We have seen in the last section that the TD estimator does not outperform the MC estimator. That is astonishing as the MC/sample-mean estimator is a general estimator that can be applied to nearly any estimation problem at hand, while the TD estimator is specifically constructed for MDPs. In particular, the MC estimator of a state $i$ does not use paths that does not hit state $i$, while TD does. One would expect that the information contained in these paths would increase the performance. And indeed, this is the case. Just TD does not use this information efficiently.

In the following, we "condition" the MC estimator with "all the information" that is available through the observed paths and we show that the resulting estimator is the optimal unbiased estimator with respect to any convex loss function. The conditioning has two effects. First, the new estimator uses the Markov structure to make use of (nearly) all paths. Second, it uses "consistent" alternative cycles beside the observed ones. For example, if a cyclic connection from state $1 \to 1$ is observed once in the first run and three times in the second run, then the optimal estimator will use paths with the cyclic connection being taken 0 to 4 times. Consistent with this finding, we show that if MC uses all

information and the modification of cycles has no effect, then the MC estimator is already optimal.

Furthermore, we show in this section that the LSTD and ML estimators are very similar to the MVU. These estimators also use all information, but they are typically biased. We show that the bias of these estimators depends mainly on the discount $\gamma$ and a normalization. We conclude this section with a comparison of these estimators to TD(0) and MC.

We start our theoretical analysis with a discussion of two classes of estimators: (1) Estimators that are unbiased and (2) estimators that fulfill the Bellman equation. Following that we formalize what we mean with all information.

### 2.3.1 Unbiased Estimators and Bellman Equation

We say that an estimator $\bar{\mathbf{V}}$ fulfills the Bellman equation if a $\bar{\mathbf{r}}$ and $\bar{\mathbf{P}}$ exist, such that

$$\bar{\mathbf{V}} = \bar{\mathbf{r}} + \gamma \bar{\mathbf{P}} \bar{\mathbf{V}}.$$

First, we observe that the Bellman equation restricts the class of estimators considerably. Essentially, the only degree of freedom is the parameter estimate $\bar{\mathbf{P}}$. If $\bar{\mathbf{P}}$ is full rank then

$$\bar{\mathbf{V}} = (\mathbf{I} - \gamma \bar{\mathbf{P}})^{-1} \bar{\mathbf{r}} =: V(\bar{\mathbf{P}}, \bar{\mathbf{r}}),$$

i.e. $\bar{\mathbf{V}}$ is completely specified by $\bar{\mathbf{P}}$ and $\bar{\mathbf{r}}$. Here, $V(\bar{\mathbf{P}}, \bar{\mathbf{r}})$ denotes the value function for an MRP with parameters $\bar{\mathbf{P}}$ and rewards $\bar{\mathbf{r}}$.

The restrictions posed by the Bellman equation leads to an introduction of a bias. Essentially, for cyclic MRPs, the class of estimators fulfilling the Bellman equation and the class of unbiased estimators does not overlap. Here, unbiased refers to all value estimators, i.e. $\mathbb{E}[\bar{V}_i | \{N_i \geq 1\}] = V_i$ for every state $i$. We will later see that value estimators for specific states can be unbiased, even if the Bellman equation is fulfilled. On the contrary, if all estimators are unbiased, then the Bellman equation can in general not be fulfilled. This does not change with a growing number of examples. The bias is reduced through more examples, but only in the asymptotic case do estimators that fulfill the Bellman equation become unbiased. In the following we argue that the factors that influence the bias are: (1) the discount, (2) the normalizations $\mathbb{P}[\{N_i \geq 1\}]$ and (3) value estimates for cases where not all states have been visited, i.e. where an $i \in \mathbb{S}$ exists with $N_i = 0$. Points (2) and (3) are quiet technical and their effect vanishes quickly.

#### Discount

Consider the MRP from Figure 2.3 (A) with the following reward setting $R_{11} := 0$ and $R_{12} := 1$, for one run ($n = 1$) and for $\gamma < 1$. In this example we use a sample mean parameter estimate, i.e. $\bar{p} = i/(i+1)$ if the cyclic transition has been taken $i$ times. This is also the maximum likelihood parameter estimate. The value of state 1 is

$$V_1 = (1-p) \sum_{i=0}^{\infty} \gamma^i p^i = \frac{1-p}{1-\gamma p}$$

and therefore the value estimate is

$$\bar{V}_1 = \frac{1 - i/(i+1)}{1 - \gamma i/(i+1)}.$$

The estimator is unbiased if and only if

$$\begin{aligned}(1-p)\sum_{i=0}^{\infty}\gamma^i p^i \overset{?}{=} \mathbb{E}[\bar{V}_1] = (1-p)\sum_{i=0}^{\infty}\frac{1-i/(i+1)}{1-\gamma i/(i+1)}p^i \\ \Leftrightarrow \sum_{i=0}^{\infty}\left(\gamma^i - \frac{1-i/(i+1)}{1-\gamma i/(i+1)}\right)p^i = 0.\end{aligned}$$

With induction one sees that $\gamma^i \leq \frac{1-i/(i+1)}{1-\gamma i/(i+1)}$. *Induction step:*

$$\begin{aligned}\gamma^{i+1} &\overset{\text{I.H.}}{\leq} \gamma\frac{1-\frac{i}{i+1}}{1-\gamma\frac{i}{i+1}} \overset{?}{\leq} \frac{1-\frac{i+1}{i+2}}{1-\gamma\frac{i+1}{i+2}} \\ \Leftrightarrow \gamma\left(1-\frac{i}{i+1}-\frac{\gamma}{i+2}\right) &\leq 1-\frac{i+1}{i+2}-\frac{\gamma i}{(i+1)(i+2)} \\ \Leftrightarrow (1-\gamma i)(\gamma-1) \leq (1-\gamma)^2 &\Leftrightarrow -(i-\gamma i) \leq 1-\gamma,\end{aligned}$$

where the last inequality holds, because $-(i-\gamma i) \leq 0$ and $(1-\gamma) \geq 0$. I.H. denotes *Induction Hypothesis*. Further, for $i = 1$

$$0 < (1-\gamma)^2 = 1 - 2\gamma + \gamma^2 \Leftrightarrow \gamma - \frac{\gamma^2}{2} < \frac{1}{2} \Leftrightarrow \gamma < \frac{1-\frac{1}{2}}{1-\frac{\gamma}{2}}$$

holds. Hence, the estimator is biased for all $\gamma < 1$.

The estimator is biased because it makes use of the value function, respectively fulfills the Bellman equation. In general, no value estimator exists that is unbiased for each MRP and each $\gamma \in (0,1)$ which uses the value function:

**Theorem 1** (p. 42)**.** *For any estimator $\bar{\mathbf{P}}$ there exists an MRP and a $\gamma \in (0,1)$ such that $V(\bar{\mathbf{P}})$ is biased.*

### Normalization $\mathbb{P}[\{N_i \geq 1\}]$ and Value Estimates on $\{N_i = 0\}$

Consider the MRP shown in Figure 2.3 (B) for one run. The MRP starts in state 2 and has a chance of $p$ to move on to state 1. The value of state 1 and 2 is

$$V_1 = V_2 = (1-p)\sum_{i=0}^{\infty} i p^i = \frac{p}{1-p}.$$

Using the sample mean parameter estimate $\bar{p} = i/(i+1)$, we get the following value estimate for state 2:

$$\bar{V}_2(i) = \frac{\bar{p}}{1-\bar{p}} = i \quad \Rightarrow \quad \mathbb{E}[\bar{V}_2(i)] = (1-p)\sum_{i=0}^{\infty}\bar{V}_2(i)p^i = V_2,$$

where $\bar{V}_2(i)$ denotes the value estimate, given the cyclic transition has been taken $i$ times. The estimator fulfills the Bellman equation. Therefore, $\bar{V}_1(i) =$

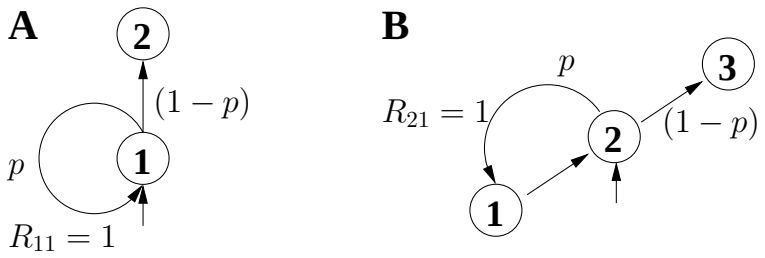


Figure 2.3: **A**: Cyclic MRP with starting state 1 and with probability p for the cyclic transition. Reward is 1 for the cyclic transition and 0 otherwise. **B**: Cyclic MRP with starting state 2 and with probability p for the cyclic transition. Reward is 1 for the cyclic transition from state 2 to state 1 and 0 otherwise.

$\bar{V}_2(i) = i$, given at least one visit of state 1, i.e. conditional on the event $\{N_1 \geq 1\}$. The expected value estimate for state 1 is therefore

$$\mathbb{E}[\bar{V}_1(i)|\{N_1 \geq 1\}] = \frac{(1-p)\sum_{i=1}^{\infty} ip^i}{(1-p)\sum_{i=1}^{\infty} p^i} = \frac{p/(1-p)}{1-(1-p)} = \frac{V_1}{p},$$

where $(1-p)\sum_{i=1}^{\infty} p^i = p$ is the normalization. Hence, the estimator is biased. More generally, assume $\bar{V}_1, \bar{V}_2$ are unbiased, i.e. $\mathbb{E}[\bar{V}_1|\{N_1 \geq 1\}] = \mathbb{E}[\bar{V}_2] = V_1 = V_2$ and the estimator fulfills the Bellman equation, i.e. $\bar{V}_1 = \bar{V}_2$ on $N_1 := \{N_1 \geq 1\}$. Then

$$\begin{aligned}\mathbb{E}[\bar{V}_2|N_1] &\overset{\text{Bellm.}}{=} \mathbb{E}[\bar{V}_1|N_1] \overset{\text{unb.}}{=} V_1 \overset{\text{Bellm.}}{=} V_2 \overset{\text{unb.}}{=} \mathbb{E}[\bar{V}_2] \\ &= \mathbb{E}[\bar{V}_2|N_1]\mathbb{P}[N_1] + \mathbb{E}[\bar{V}_2|N_1^c]\mathbb{P}[N_1^c] \\ &\Rightarrow \mathbb{E}[\bar{V}_2|N_1] = \mathbb{E}[\bar{V}_2|N_1^c] \Rightarrow \mathbb{E}[\bar{V}_2] = 0,\end{aligned}$$

where $N_1^c$ denotes the event $N_1 = 0$. $\mathbb{E}[\bar{V}_2] = 0$ contradicts unbiasedness. Besides using the Bellman equation and unbiasedness we used that the estimator of the deterministic transition $1 \to 2$ equals 1, i.e. $\bar{p}_{12} = 1$ and that $\bar{V}_2 = 0$ if only the transition $2 \to 3$ is taken.

More intuitively the reasons for the bias are: Firstly, $\bar{V}_1$ equals $\bar{V}_2$ on $\{N_1 \geq 1\}$ but the estimators differ (in general) on $\{N_1 = 0\}$. In the example, we made no use of this point. We could make use of it by introducing a reward for the transition $2 \to 3$. Secondly, the normalization differs, i.e. $\mathbb{E}[\cdot]$ versus $\mathbb{E}[\cdot|\{N_1 \geq 1\}]$. In our example we used this point. Both estimators are 0 on $N_1^c$ and are therefore always equivalent. However the expectation is calculated differently and introduces the bias.

How does these effects behave in dependence of the number $n$ of paths? Let $p_i$ denote the probability to visit state $i$ in one sampled path. Then the probability of the event $\{N_i = 0\}$ drops exponentially fast, i.e. $\mathbb{P}[\{N_i = 0\}] \leq (1-p_i)^n$ and the normalization $1/\mathbb{P}[\{N_i \geq 1\}]$ approaches exponentially fast 1. Therefore, if the estimates are upper bounded on $\{N_i = 0\}$ then the bias drops exponentially fast in $n$.

### 2.3.2 Maximum Likelihood Parameter Estimates and Sufficient Statistics

We start this section with a derivation of the maximum likelihood parameter estimates. After that we introduce a minimal sufficient statistics for MRPs and we show that this statistic equals the maximum likelihood estimates.

#### Maximum Likelihood Parameter Estimates

Let $p_{ij}$ be the transition probability of state $i$ to $j$, $p_i$ the probability to start in $i$ and $x$ a sample consisting of $n$ iid state sequences $x_1, \ldots, x_n$. The log-likelihood of the sample is

$$\log \mathbb{P}[x|p] = \sum_{k=1}^{n} \log \mathbb{P}[x_k|p].$$

The corresponding maximization problem is given by

$$\max_{p_{ij},p_i} \sum_{i=1}^{n} \log \mathbb{P}[x_i|p_{ij},p_i], \qquad \text{s.t.:} \sum_{j\in\mathbb{S}} p_{ij} = \sum_{j\in\mathbb{S}} p_j = 1.$$

The unique solution for $p_{ij}$ and $p_i$ (Lagrange multipliers) is given by

$$p_{ij} = \frac{\mu_{ij}}{K_i} =: \bar{p}_{ij} \quad \text{and} \quad p_i = \frac{1}{n}\big(K_i - \sum_{j\in\mathbb{S}} \mu_{ji}\big) =: \bar{p}_i, \tag{2.6}$$

where $K_i$ denotes the number of visits of state $i$, $\mu_{ij}$ the number of direct transitions from $i$ to $j$, $\bar{p}_{ij}$ the estimate of the true transition probability $p_{ij}$ and $\bar{p}_i$ the estimate of the true starting probability $p_i$.

#### Sufficient Statistics for the MRP Parameters

Information about a sample is typically available through a *statistic* $\mathcal{S}$ of the data (for example $\mathcal{S} = \sum_i x_i$, where $x$ is a sample). A statistic which contains all information about a sample is called *sufficient*. Important properties of sufficient statistics are *minimality* and *completeness*. The minimal sufficient statistics is the sufficient statistic with the smallest dimension (typically the same dimension as the parameter space). Formally, suppose that a statistic $\mathcal{S}$ is sufficient for a parameter $\theta$. Then $\mathcal{S}$ is minimally sufficient if $\mathcal{S}$ is a function of any other statistic $\mathcal{T}$ that is sufficient for $\theta$. Formally, a statistic $\mathcal{S}$ is complete if $\mathbb{E}_\theta[h(\mathcal{S})] = 0$ for all $\theta$ implies $h = 0$ almost surely. The theorem from *Rao and Blackwell* [75] states that for a complete and minimal sufficient statistics $\mathcal{S}$ and any unbiased estimator $A$ of a parameter $\theta$ the estimator $\mathbb{E}[A|\mathcal{S}]$ is the optimal unbiased estimator with respect to any convex loss function and hence the unbiased estimator with minimal MSE.

The maximum likelihood solution is a *sufficient statistics* for the MRP parameters. We demonstrate this with the help of the *Fisher-Neyman factorization theorem* [75]. It states that a statistic is sufficient if and only if the density $f(\mathbf{x}|\theta)$ can be factored into a product $g(\mathcal{S},\theta)h(\mathbf{x})$. For an MRP we can factor

the density as needed by the Fisher-Neyman theorem ($h(\mathbf{x}) = 1$ in our case),

$$\mathbb{P}(\mathbf{x}|p) = \prod_{i=1}^{n}\Big(p_{\mathbf{x}_i(1)}\prod_{j=2}^{L_i} p_{\mathbf{x}_i(j-1)\mathbf{x}_i(j)}\Big) = \prod_{s\in\mathbb{S}} p_s^{(K_s-\sum_{s'}\mu_{s's})}\prod_{s,s'\in\mathbb{S}} p_{ss'}^{K_s\mu_{ss'}},$$

where $\mathbf{x}_i(j)$ is the $j$th state in the $i$th path, $n$ the number of observed paths and $L_i$ the length of the $i$th path. $K_s\mu_{ss'}$ is sufficient for $p_{ss'}$ and because sufficiency is sustained by one-to-one mappings [75] this holds true also for $\mu_{ss'}$. The sufficient statistics is *minimal* because the maximum likelihood solution is unique [75][1]. The sufficient statistic is also *complete* because the sample distribution induced by an acyclic MRP forms an *exponential family* of distributions (Lemma 15, page 43). Due to [51] any exponential family of distributions is complete.

### 2.3.3 Optimal Unbiased Value Estimator

The *Rao-Blackwell theorem* [75] states that for any unbiased estimator $A$ the estimator $\mathbb{E}[A|\mathcal{S}]$ is the optimal unbiased estimator, given $\mathcal{S}$ is a minimal and complete sufficient statistic. For the case of value estimation this means that we can use any unbiased value estimator (e.g. the Monte Carlo estimator) and condition it with the statistic induced by the maximum likelihood parameter estimate to get the optimal unbiased value estimator.

**Theorem 2.** *Let $\bar{V}$ be the first-visit Monte-Carlo estimator and $\mathcal{S}$ the sufficient and complete statistics for a given MRP. The estimator $\mathbb{E}[\bar{V}|\mathcal{S}]$ is unbiased and the optimal unbiased estimator with respect to any convex loss function. Especially, it has minimal MSE.*

From now on, we refer to the estimator $\mathbb{E}[\bar{V}|\mathcal{S}]$ as the *Minimum Variance Unbiased estimator (MVU)*. For a deterministic reward the estimator $\mathbb{E}[\bar{V}|\mathcal{S}]$ is given by

$$\mathbb{E}[\bar{V}|\mathcal{S}] = \frac{1}{|\mathbf{\Pi}(\mathcal{S})|}\sum_{\boldsymbol{\pi}\in\mathbf{\Pi}(\mathcal{S})}\bar{V}(\boldsymbol{\pi}), \tag{2.7}$$

where $\boldsymbol{\pi} := (\pi_1, \ldots, \pi_i)$ denotes a vector of paths, $\mathbf{\Pi}(\mathcal{S})$ denotes the set of vectors of paths which are consistent with the observation $\mathcal{S}$, $|\cdot|$ is the size of a set and $\bar{V}(\boldsymbol{\pi})$ is the MC estimate for the vector of paths $\boldsymbol{\pi}$. Essentially, $\boldsymbol{\pi}$ is an ordered set of paths and it is an element of $\mathbf{\Pi}(\mathcal{S})$ if it produces the observed transitions, starts and rewards. The MC estimate is simply the average value for the paths in $\boldsymbol{\pi}$. The estimator $\mathbb{E}[\bar{V}|\mathcal{S}]$ is thus the average over all paths which could explain the (compressed) observed data $\mathcal{S}$. As an example, take the two state MRP from Figure 2.3 (A). Assume that an agent starts twice in state 1, takes three times the cycle in the first run and once in the second. The paths which are consistent with this observation are:

$$\mathbf{\Pi}(\mathcal{S}) = \{((1,1,1,2),(1,2)),((1,1,2),(1,1,2)),((1,2),(1,1,1,2))\}.$$

[1] It is needed to use the minimal parameter set of the MRP to be formally correct. The minimal sufficient statistics excludes also one value $\mu_{ss'}$, however the missing value is defined by the other $\mu$'s.

The MC estimator for the value of a state $s$ does not consider paths which do not hit $s$. On the contrary to that the conditioned estimator uses these paths. To see this, let us again take a look at the MRP from Figure 2.2 (A). Assume, that two paths were sampled: $(1, 2, 4)$ and $(2, 3)$. The MC value estimate for state one uses only the first path. Taking a look at

$$\mathbf{\Pi}(\mathcal{S}) = \{((1,2,4),(2,3)), ((\mathbf{1},\mathbf{2},\mathbf{3}),(2,4)), ((2,3),(1,2,4)), ((2,4),(\mathbf{1},\mathbf{2},\mathbf{3}))\},$$

we see that the conditioned estimator uses the information.

### Costs of Unbiasedness

The intuition that the MVU uses all paths is, however, not totally correct. Let us take a look at the optimal unbiased value estimator of state 1 of the MRP in Figure 2.3 (B) for $\gamma = 1$. Furthermore, assume that one run is made and that the path $(2, 1, 2, 3)$ is observed. No permutations of this path are possible and the estimate of state 1 is therefore the MC estimate of path $(1, 2, 3)$, which is 0. In general, if we make one run and we observe $i$ transitions from state 2 to state 1, then the estimate is $(i - 1)$. I.e. we ignore the first transition. As a consequence, we have on average the following estimate:

$$(1-p)\sum_{i=1}^{\infty}(i-1)p^i = p\frac{p}{1-p} = pV_1.$$

The term $p$ is exactly the probability of the event $\{N_1 \geq 1\}$ and the estimator is conditionally unbiased on this event. The intuition is, that the estimator needs to ignore the first transition to achieve (conditional) unbiasedness.

Hence, unbiasedness has its price. Another cost beside this loss in information is that the Bellman equation cannot be fulfilled. In Section 2.3.1 we started with estimators that fulfill the Bellman equation and we showed that the estimators are biased. Here, we have a concrete example of an unbiased estimator that does not fulfill the Bellman equation, as $\bar{V}_1 = (i - 1) \neq i = \bar{V}_2$. For this example this is counterintuitive as $p_{12} = 1$ and essentially no difference between the states exists in the undiscounted case.

### Undiscounted MRPs

In the undiscounted case permutations of paths do not change the cumulated reward. For example, $\sum_{i=1}^{n} R_{\pi(i)\pi(i+1)} = \sum_{i=1}^{n} R_{\pi(\sigma(i))\pi(\sigma(i)+1)}$, if $\sigma$ is a permutation of $(1, \ldots, n)$, because the time at which a reward is observed is irrelevant. This invariance to permutations implies already a simple fact. We need the following criterion to state this fact:

**Criterion 3** (Full Information)**.** *A state $s$ has full information if, for every successor state $s'$ of $s$ and all paths $\pi$, it holds that*

$$\pi(i) = s' \Rightarrow \exists j \text{ with } j < i \text{ and } \pi(j) = s.$$

$\pi(i)$ denotes the $i$th state in the path.

Let $\boldsymbol{\pi}$ be a vector of paths following the first visit of state $s$ that are consistent with the observations. $\bar{V}(\boldsymbol{\pi})$ is then given by $(1/|\boldsymbol{\pi}|)\sum_i \sum_j R^{(i)}_{jj+1}$, where $|\boldsymbol{\pi}|$

is the number of paths contained in $\boldsymbol{\pi}$ and $R^{(i)}_{jj+1}$ is the observed reward in path $i$ at position $j$. Rearranging the path does not change the sum and the normalizing term. Therefore each consistent path results in the same first-visit MC estimate and the MVU equals the first-visit MC estimator.

**Corollary 4.** *Let $\bar{V}$ be the first-visit MC estimator and let the value function be undiscounted. If the Full Information Criterion applies to a state $s$, then*

$$\mathbb{E}[\bar{V}_s|\mathcal{S}] = \bar{V}_s.$$

The undiscounted setting allows alternative representations of the optimal estimator. As an example, suppose we observed one path $\pi := (1,1,1,2)$ with reward $R(\pi) = 2R_{11}+1R_{12}$. The optimal estimator is given by $R(\pi)$. alternative representation of the estimator can be constructed by splitting the path in two: $\{\pi_1 := (1,1,2), \pi_2 := (1,1,1,1,2)\}$ with reward $R(\pi_1) = (1R_{11} + 1R_{12})$ and $R(\pi_2) = (3R_{11} + 1R_{12})$ and with $P(\pi_1) + P(\pi_2) = 1$, e.g. "$\frac{1}{2}((1,1,2) + (1,1,1,1,2)) = (1,1,1,2)$". More generally, the optimal unbiased estimator for this MRP can be represented as

$$\sum_j w_j(jR_{11} + R_{12}), \tag{2.8}$$

given $\sum_j w_j = 1$ and $\sum_j jw_j = i$, where $i$ denotes the number of times the transition $1 \to 1$ has been observed (in the example $i = 2$). We come back to this representation in the discussion of the ML estimator.

**Convergence**

Intuitively, the estimator should converge because MC converges in $L^1$ and almost surely. Furthermore, conditioning reduces norm-induced distances to the true value. This is already enough to follow $L^1$ convergence but the almost sure convergence is not induced by a norm. We therefore refer to an integral convergence theorem which allows us to follow a.s. from $L^1$ convergence under weak assumptions. Details are given in Appendix 2.B.3.

**Theorem 5** (p. 44)**.** $\mathbb{E}[\bar{V}|\mathcal{S}]$ *converges on average and almost surely to the true value.*

A MVU algorithm can be constructed using Equation 2.7. However, the algorithm needs to iterate through all possible paths and therefore has an exponential computation time.

### 2.3.4 Least-Squares Temporal Difference Learning

In this section we discuss the relation of the MVU to the LSTD estimator. The LSTD estimator was introduced by [19] and extensively analyzed in [17] and [18]. Empirical studies showed that LSTD often outperforms massively TD and MC with respect to convergence speed per sample size. In this section we support these empirical findings by showing that the LSTD estimator is equivalent to the MVU for acyclic MRPs and closely related to the MVU for undiscounted MRPs. We derive our statements not directly for LSTD, but for the maximum likelihood value estimator (ML) which is equivalent to LSTD

(Section 2.3.4). The estimator is briefly sketched in [77], where it is also shown that batch TD(0) is in the limit equivalent to the ML estimator. The estimator is also implicitly used in the *certainty-equivalence* approach, where a maximum likelihood estimate of the MDP is used for optimization.

### Maximum Likelihood Estimator

The ML value estimator is given by $V(\bar{\mathbf{P}}, \bar{\mathbf{r}})$, where $\bar{\mathbf{P}} := \left(\bar{p}_{ij}\right)$ is the maximum likelihood estimate of the transition matrix and $\bar{\mathbf{r}}$ is the vector of the maximum likelihood estimates of the expected one step reward. Hence, the ML value estimator is given by:

$$\bar{\mathbf{V}} = \sum_{i=0}^{\infty} \gamma^i \bar{\mathbf{P}}^i \bar{\mathbf{r}} = (\mathbf{I} - \gamma \bar{\mathbf{P}})^{-1} \bar{\mathbf{r}}, \tag{2.9}$$

whereas the Moore-Penrose pseudoinverse is used if $\bar{\mathbf{P}}$ is singular (e.g. too few samples).

### Acyclic MRPs

If an estimator is a function of the sufficient statistic (e.g. $A = f(\mathcal{S})$) then the conditional estimator is equal to the original estimator, $A = \mathbb{E}[A|\mathcal{S}]$. If the estimator $A$ is also unbiased then it is due to the Rao-Blackwell theorem the optimal unbiased estimator. The defined maximum likelihood estimator is a function of a minimal and complete sufficient statistic. It is unbiased in the acyclic case and therefore equivalent to the MVU.

**Theorem 6** (p. 45)**.** *The ML estimator is unbiased if the MRP is acyclic.*

**Corollary 7.** *The ML estimator is equivalent to the MVU if the MRP is acyclic.*

### Undiscounted MRPs

In the cyclic case the class of unbiased estimators is separated from the set of estimators that fulfill the Bellman equation if $\gamma < 1$ or the normalization effect of Section 2.3.1 applies. In this section we argue that these are essentially the only factors that separate the classes. In particular, if the Full Information Criterion holds for a state $i$, then the normalization $\{N_i \geq 1\}$ applies to all successor states of state $i$ and no successor state got estimates if the event $\{N_i = 0\}$ occurs. Thus, if this Criterion holds for a state then the normalization problem does not occur. And actually in this case estimators that fulfill the Bellman equation can be unbiased for state $i$. In particular, the ML estimator is, in this case, unbiased and equivalent to the MVU. This can be shown by using Theorem 5 from [69], which states that the ML estimator equals the first-visit MC estimator if the Full Information Criterion holds. Furthermore, in this case the first-visit MC estimator is equivalent to the MVU (Corollary 4).

**Corollary 8.** *The ML estimator of a state $i$ is unbiased and equivalent to the MVU if the Full Information Criterion applies to state $i$ and an undiscounted MRP is given.*

We analyze this effect using a simple MRP.

**Cyclic MRP - Unbiased** The value of state 1 for the MRP in Figure 2.3 (A) and $\gamma = 1$ is $V_1 = (1-p)\sum_{i=0}^{\infty} ip^i$. The ML estimate for state 1 given a sample of $n$ paths is

$$\bar{V}_1 = \Big(1 - \frac{k}{k+n}\Big)\sum_{i=0}^{\infty} i\Big(\frac{k}{k+n}\Big)^i = \Big(1 - \frac{k}{k+n}\Big)\frac{k/(k+n)}{(1-k/(k+n))^2} = \frac{k}{n},$$

where $k$ is the number of taken cycles (summed over all observed paths). Therefore

$$\mathbb{E}[\bar{V}_1] = (1-p)\sum_{i=0}^{\infty} \bar{V}(i)p^i = V_1,$$

where $\bar{V}(i)$ denotes the ML estimate if the cyclic connection has been taken $i$ times. Hence, the ML estimator is unbiased. This point is interesting as in general nonlinear transformations of unbiased parameter estimates produce biased estimators, as

$$\mathbb{E}[f(\hat{\theta})] = V = f(\theta) = f(\mathbb{E}[\hat{\theta}])$$

essentially means that $f$ is a linear transformation. However, in our example the estimator $\hat{\theta}$ is actually not unbiased. For $n = 1$:

$$\mathbb{E}\left[\frac{k}{k+1}\right] = (1-p)\sum_{k=0}^{\infty}\frac{k}{k+1}p^k < (1-p)\sum_{k=1}^{\infty}p^k = (1-p)\sum_{k=0}^{\infty}p^{k+1} = p.$$

The parameter is underestimated on average. The reason for this lies in the dependency between the visits of state 1. For a fixed number of visits, respectively for *iid* observations the parameter estimate would be unbiased. The relation between these two estimation settings is very similar to the first-visit and every-visit MC setting. The first-visit MC estimator is unbiased because it uses only one observation per path while the every-visit MC estimator is biased. In our case, the effect is particularly paradox as for the *iid* case the value estimator is biased.

The MVU estimate for this MRP is also $\frac{k}{n}$. This can be seen in the following way. Let $u$ be the number of ways how $k$ can be split onto $n$-paths. For each split the summed reward is $k$ and the MC estimate is therefore $\frac{k}{n}$. Hence, the MVU is $\frac{uk/n}{u} = \frac{k}{n}$. Thus the MVU is equivalent to the ML and to the MC estimator. The reason for the equivalence of the MVU and the ML estimator is the unbiasedness of ML. The MC estimator equals the MVU as the Full Information Criterion and Corollary 4 apply. Intuitively, the ML estimator equals the MVU for this MRP because ML fulfills Equation 2.8 with $w_j = (1-\bar{p})\bar{p}^j$ (for simplicity $n = 1$). In other words, the ML estimator is an alternative representation of the MVU.

### Relation to the MVU

The ML estimator differs from the MVU because it uses paths that are inconsistent with the observation $\mathcal{S}$. For example, given the MRP from Figure 2.3 (A) and the observation $(1,1,1,2)$. The set of paths consistent with this observation is again $\{(1,1,1,2)\}$. The ML estimator on the other hand uses the following set of paths:

$$\{(1,2),(1,1,2),(1,1,1,2),(1,1,1,1,2)\ldots\}.$$

This, however, does not mean that the ML estimator is worse than the MVU. The ML estimator is also a function of the sufficient statistics, it is just not unbiased. To demonstrate this, we present two examples based on the MRP from Figure 2.3 (A) in Appendix 2.B.5 (p. 45). One for which the MVU is superior and one where the ML estimator is superior.

#### The LSTD Estimator

The LSTD algorithm computes analytically the parameters which minimize the empirical quadratic error for the case of a linear system. [19] showed that the resulting algorithm converges almost surely to the true value. In [17] a further characterization of the least-squares solution is given. This turns out to be useful to establish the relation to the ML value estimator. According to this characterization, the LSTD estimate $\bar{V}$ is the unique solution of the Bellman equation, i.e.

$$\bar{\mathbf{V}} = \bar{\mathbf{r}} + \gamma \bar{\mathbf{P}} \bar{\mathbf{V}}, \tag{2.10}$$

where $\bar{\mathbf{r}}$ is the sample mean estimate of the reward and $\bar{\mathbf{P}}$ is the maximum likelihood estimate of the transition matrix.

Comparing Equation 2.10 with Equation 2.9 of the ML estimator it becomes obvious that both are equivalent if the sample mean estimate of the reward equals the maximum likelihood estimate.

**Corollary 9.** *The ML value estimator is equivalent to LSTD.*

### 2.3.5 Monte Carlo Estimation

We first summarize Theorem 5 from [69] and Corollary 4 from p. 25:

**Corollary 10.** *The (first-visit) MC estimator of a state i is equivalent to the MVU and to the ML estimator if the Full Information Criterion applies to state i and an undiscounted MRP is given.*

Essentially, the corollary tells us that in the undiscounted case it is only the "amount" of information that makes the difference between the MC estimator and the MVU, respectively the ML estimator. Amount of information refers here to the observed paths. If MC observes every path then the estimators are equivalent.

From a different point of view this tells us that in the undiscounted case the MRP structure is only useful for passing information between states, but yields no advantage beyond that.

#### Discounted MRPs

In the discounted cyclic case the MC estimator differs from the ML and the MVU estimator. It differs from ML because ML is biased. The MC estimator is equivalent to the MVU in the undiscounted case because the order in which the reward is presented is irrelevant. That means the time at which a cycle occurs is irrelevant. In the discounted case this is not true anymore. Consider again the MRP from Figure 2.3 (A) with the following two paths $\boldsymbol{\pi} = ((1,1,1,2),(1,2))$. The MC estimate is $1/2((1+\gamma)+0)$. The set of paths consistent with this observation is $\boldsymbol{\Pi}(\mathcal{S}) = \{((1,1,1,2),(1,2)),((1,1,2),(1,1,2)),((1,2),(1,1,1,2))\}$.

Hence, the MVU uses the cycle $(1, 1, 2)$ besides the observed ones. The MVU estimate is $1/3((1+\gamma)/2 + 2/2 + (1+\gamma)/2) = 1/3(2+\gamma)$. Both terms are equivalent if and only if $\gamma = 1$. For this example the Full Information Criterion applies.

Similarly, for acyclic MRPs the MC estimator is different from the ML/MVU estimator if $\gamma < 1$. Consider a 5 state MRP with the following observed paths: $((1,3,4),(1,2,3,5))$, a reward of $+1$ for $3 \to 4$ and $-1$ for $3 \to 5$. The ML estimate is $(1/4\gamma^2 + 1/4\gamma)(1-1) = 0$, while the MC estimate is $1/2(-\gamma^2+\gamma)$ which is 0 if and only if $\gamma = 1$. Again the Full Information Criterion applies.

### 2.3.6 Temporal Difference Learning

One would like to establish inequalities between the estimation error of TD and the error of other estimators like the MVU or the ML estimator. For the acyclic case TD($\lambda$) is essentially unbiased and the MVU and the ML estimator are superior to TD. However, for the cyclic case the analyzes is not straightforward, as TD($\lambda$) is biased for $\lambda < 1$ and does not fulfill the Bellman equation. So TD is in a sense neither in the estimator class of the MVU nor of the ML estimator and conditioning with a sufficient statistics does not project TD to either of these estimators.

The bias of TD can be verified with the MRP from Figure 2.3 (A). I.e. for the case of TD(0), a learning rate of $\alpha_j = 1/j$, $\gamma = 1$ and of one path, the value estimate for state 0 is $i/(i+1)\sum_{j=1}^{i} 1/j$ if $i$ cyclic transitions have been observed. The estimate should on average equal $i$ to be unbiased. However, for $i > 0$ it is strictly smaller than $i$.

While our tools are not usable to establish inferiority of TD, we can still interpret the weaknesses of TD with it. In the following we focus on the TD(0) update rule.

#### Weighting of Examples and Conditioning

We have seen in Section 2.2.4 that a weakness of TD(0) is that not all of the examples are weighted equally. In particular Equation 2.2 on page 15 suggests that no observation should be preferred over another. Intuitively, conditioning suggests so too: For an acyclic MRP TD(0) can be written as $\bar{V}_i = \tilde{p}_{ij}(R_{ij} + \gamma \bar{V}_j)$, whereas $\tilde{p}_{ij}$ differs from the maximum likelihood parameter estimates $\bar{p}_{ij}$ due to the weighting. Generally, conditioning with a sufficient statistics permutes the order of the observations and resolves the weighting problem. Therefore, one would assume that conditioning with the element $\bar{p}_{ij}$ of the sufficient statistics changes $\bar{V}_i$ to $\bar{p}_{ij}(R_{ij} + \gamma \bar{V}_j)$. As conditioning improves the estimate, the new estimator would be superior to TD(0). However, conditioning with just a single element $\bar{p}_{ij}$ must not modify the estimator at all, as the original path might be reconstructed from the other observations. E.g. if one observes a transition $1 \to 2$ and $2 \to 3$, with $2 \to 3$ being the only path from state 2 to state 3, then it is enough to know that transition $1 \to 2$ occurred and state 3 was visited.

Despite these technical problems, the superiority of $\bar{p}_{ij}$ over $\tilde{p}_{ij}$ and the weighting problem are reflected in the contraction properties of TD(0). Due to [77] TD(0) contracts towards the ML solution. Yet, the contraction is slow compared to the case where each example is weighted equally.

**Weighting of Examples and Contraction Factor**

We continue with another look at the familiar ML equation: $\bar{\mathbf{V}} = \bar{\mathbf{r}} + \gamma\bar{\mathbf{P}}\bar{\mathbf{V}} =: \bar{\mathbf{T}}\bar{\mathbf{V}}$. If the matrix $\bar{\mathbf{P}}$ is of full rank then the ML estimate is the sole fixed point of the Bellman operator $\bar{\mathbf{T}}$. The ML estimate can be gained by solving the equation, i.e $\bar{\mathbf{V}} = (\mathbf{I} - \gamma\bar{\mathbf{P}})^{-1}\bar{\mathbf{r}}$. Alternatively, it is possible to make a fixed point iteration. I.e. starting with an initial guess $\bar{\mathbf{V}}^{(0)}$ and iterating the equation, i.e $\bar{\mathbf{V}}^{(n)} = \bar{\mathbf{T}}\bar{\mathbf{V}}^{(n-1)}$. Convergence to the ML solution is guaranteed by the *Banach Fixed Point Theorem*, because $\bar{\mathbf{T}}$ is a contraction. The contraction factor is upper bounded by $\gamma||\bar{\mathbf{P}}|| \leq \gamma$, where $||\cdot||$ denotes in the following the *operator norm*. The bound can be improved by using better suited norms (e.g. [15]). Hence, for $n$ updates the distance to the ML solution is reduced by a factor of at least $\gamma^n$.

Applying the TD(0) update (Eq. 2.3) to the complete value estimate $\bar{\mathbf{V}}$ using $\bar{\mathbf{P}}$ and a learning rate of $1/n$ results in

$$\begin{aligned}\bar{\mathbf{V}}^{(n)} &= \bar{\mathbf{V}}^{(n-1)} + \frac{1}{n}\left(\mathbf{r} + \gamma\bar{\mathbf{P}}\bar{\mathbf{V}}^{(n-1)} - \bar{\mathbf{V}}^{(n-1)}\right) \\ &= \frac{n-1}{n}\bar{\mathbf{V}}^{(n-1)} + \frac{1}{n}\bar{\mathbf{T}}\bar{\mathbf{V}}^{(n-1)} =: \bar{\mathbf{S}}^{(n)}\bar{\mathbf{V}}^{(n-1)}.\end{aligned}$$

In this equation the weighting problem becomes apparent: The contraction $\bar{\mathbf{T}}$ affects only a part of the estimate. Yet, the operators $\bar{\mathbf{S}}^{(n)}$ are still contractions. For $\bar{\mathbf{V}}$ and $\bar{\mathbf{W}}$:

$$||\bar{\mathbf{S}}^{(n)}\bar{\mathbf{V}} - \bar{\mathbf{S}}^{(n)}\bar{\mathbf{W}}|| \leq \frac{n-1}{n}||\bar{\mathbf{V}} - \bar{\mathbf{W}}|| + \frac{1}{n}||\bar{\mathbf{T}}||||\bar{\mathbf{V}} - \bar{\mathbf{W}}|| \leq \frac{n-1+\gamma}{n}||\bar{\mathbf{V}} - \bar{\mathbf{W}}||.$$

The contraction coefficient is therefore at least $\frac{n-1+\gamma}{n}$. The ML solution (in the following $\bar{\mathbf{V}}$) is a fixed point for the $\bar{\mathbf{S}}^{(i)}$ and for $n$ iterations the distance is bounded by

$$||\bar{\mathbf{S}}^{(n)} \dots \bar{\mathbf{S}}^{(1)}\bar{\mathbf{V}}^{(0)} - \bar{\mathbf{V}}|| \leq \frac{\prod_{i=0}^{n-1}(i+\gamma)}{n!}||\bar{\mathbf{V}}^{(0)} - \bar{\mathbf{V}}||.$$

The smaller $\gamma$ the faster the contraction. Yet, even in the limit the contraction is much slower than the contraction with the ML fixed point iteration, i.e. for $\gamma = 0$ the distance decreases at least with $1/n$ while for the ML fixed point iteration it decreases with $\gamma^n$. For $\gamma = 0.1$ and two applications of the Bellman operator the contraction is at least $\gamma^2 = 1/100$ and it needs 100 iterations with the TD(0) equation to reach the same distance.

TD(0) is applied only to the current state and not to the full value vector. The same can be done with the ML fixed point iteration, i.e. $\bar{V}_i = \bar{p}_{ij}(\bar{R}_{ij} + \gamma\bar{V}_j)$. We analyze the contraction properties of this estimator in the empirical part and we refer to the estimator as the iterative Maximum Likelihood (iML) estimator. The costs of the algorithm are slightly higher than the TD(0) costs: $O(|\mathbb{S}|)$ (time) and $O(|\mathbb{S}|^2)$ (space).

The restriction to the current path does not affect the convergence, i.e. the restricted iteration converges to the ML solution. Intuitively, the convergence is still guarantied, as a contraction of $\gamma$ is achieved by visiting each state once and because each state is visited infinitely often. Using that idea the following Theorem can be proved:

**Theorem 11.** *iML is unbiased for acyclic MRPs, converges on average and almost surely to the true value.*

We use this algorithm only for the analysis and we therefore omit the proof.

## 2.4 Comparison of Estimators: Experiments

In this section we make an empirical comparison of the estimators. We start with a comparison using acyclic MRPs. For this case the ML estimator equals the MVU and the MVU solution can be efficiently computed. This allows us to make a reasonable comparison of the MVU/ML estimator with other estimators. In a second set of experiments we compare the MVU with the ML estimator using a very simple cyclic MRP. In a final set of experiments we compare the contraction properties of iML and TD(0).

We start with a table that summarizes the properties of the different estimators. The row **Optimal** refers to the class of unbiased estimators and to convex loss functions. The statement that ML is unbiased if the Full Information Criterion is fulfilled and $\gamma = 1$ applies state wise. I.e. for a cyclic MRP there will exist a state for which the ML estimator is biased. However, if the Full Information Criterion applies to a state, then the ML estimator for this state is unbiased.

| Estimator | **MVU** | **ML/LSTD** | **TD($\lambda$)** | **(First-visit) MC** |
|---|---|---|---|---|
| Convergence | a.s., $L^1$ | a.s., $L^1$ | a.s., $L^1$ | a.s., $L^1$ |
| Cost (Time)<br>Cost (Space) | exp? | $O(\|\mathbb{S}\|^3)$<br>$O(\|\mathbb{S}\|^3)$ | $O(\|\mathbb{S}\|)$<br>$O(\|\mathbb{S}\|)$ | $O(\|\mathbb{S}\|)$<br>$O(\|\mathbb{S}\|)$ |
| Unbiased | $\surd$ | Acyclic or Cr. 3 and $\gamma = 1$ | Acyclic (minor ch.) | $\surd$ |
| Bellman | Acyclic | $\surd$ | | |
| Optimal | $\surd$ | Acyclic or Cr. 3 and $\gamma = 1$ | | Cr. 3 and $\gamma = 1$ |

### 2.4.1 Acyclic MRPs

We performed three experiments for analyzing the estimators. In the first experiment we measured the MSE in dependence to the number of observed paths. In the second experiment we analyzed how the MRP structure affects the estimation performance. As we can see from Corollary 4 the performance difference between "MDP" based estimators such as TD or ML and model free estimators like MC depends on the ratio between the number of sequences hitting a state $s$ itself and the number of sequences entering the subgraph of successor states without hitting $s$. We varied this ratio in the second experiment and measured the MSE. The third experiment was constructed to analyze the practical usefulness of the different estimators. We measured the MSE in relation to the calculation time.

**Basic Experimental Setup** We generated randomly acyclic MRPs for the experiments. The generation process was the following: We started by defining a state $s$ for which we want to estimate the value. Then we generated randomly a graph of successor states. We used different layers with a random number

of states in each layer. Connections were only allowed between adjacent layers. Given these constraints, the transition matrix was generated randomly (uniform distribution). For the different experiments, a specific number of starts in state $s$ was defined. Beside that, a number of starts in other states were defined. Starting states were all states in the first layers (typically the first 4). Other layers which were further apart from $s$ were omitted as paths starting in these contribute few to the estimate, but consume computation time. The distribution over the starting states was chosen to be uniform. Finally, we randomly defined rewards for the different transitions (between 0 and 1), while a small percentage (1 to 5 percent) got a high reward (reward 1000). Beside the reward definition, this class of MRPs contains a wide range of acyclic MRPs. We tested the performance (empirical MSE) of the ML, iML, MC and TD estimators. For the first two experiments the simulations were repeated 300 000 times for each parameter setting. We splitted these runs into 30 blocks with 10 000 examples each and calculated the mean and standard deviation for these. In the third experiment we only calculated the mean using 10 000 examples. We used the modified TD(0) version which is unbiased with a learning rate of $1/i$ for each state. The ML solution was computed at the end and not at each run. This means no intermediate estimates were available, which can be a drawback. We also calculated the standard TD(0) estimates. The difference to the modified TD(0) version is marginal and therefore we did not include the results in the plots.

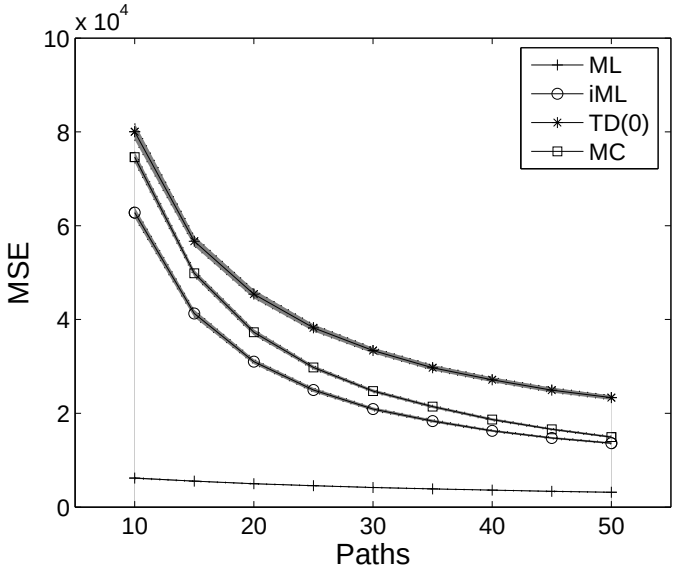


Figure 2.4: MSE of ML, iML, TD(0) and MC in relation to the number of observed paths. The state space consisted of 10 layers with 20 states per layer.

**Experiment 1: MSE in Relation to the Number of Observed Paths**

In the first experiment, we analyzed the effect of the number of observed paths given a fixed rate of $p_s = 0.2$ for starts in state $s$. The starting probability for state $s$ is high and beneficial to MC (The effect of $p_s$ is analyzed in the second experiment). Apart from ML, all three estimators perform quite similarly with a small advantage for iML and MC (Figure 2.4). ML is even for few paths strongly superior and the estimate is already good for 10 paths. Note that, due to the scale the improvement of ML is hard to observe.

**Experiment 2: MSE in Relation to the Starting Probability**

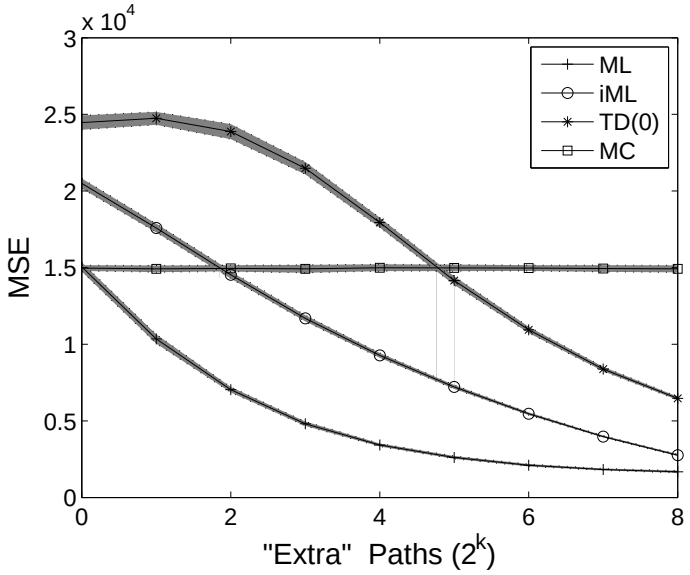


Figure 2.5: MSE of ML, iML, TD(0) and MC in relation to the starting probability of the estimated state. The state space consisted of 10 layers with 20 states per layer.

In the second experiment we tested how strongly the different estimators use the Markov structure. To do so, we varied the ratio of starts in state $s$ (the estimator state) to starts in the subgraph. The paths which start in the subgraph can only improve the estimation quality of state $s$ if the Markov structure is used. Figure 2.5 shows the results of the simulations. The $x$-axis gives the number of starts in the subgraph while the number of starts in state $s$ was set to 10. We increased the number exponentially. The exponential factor is printed on the x-axis. $x = 0$ is equivalent to always start in $s$. One can see that the MC and ML estimator are equivalent if in each run the path starts in $s$. Further, for this case MC outperforms TD due to the weighting problem of TD (Section 2.3.6). Finally, TD, iML and ML make a strong use of paths which does not visit state $s$ itself. Therefore, TD becomes superior to MC for a higher number of paths. The initial plateau for the TD estimator appeared

in the modified and the standard version. We assume that it is an effect of the one step error propagation of TD(0). For the one step error propagation a path starting in a state $s'$ in the $ith$ layer can only improve the estimate if $i$ paths are observed that span the gap between $s$ and $s'$. The probability of such an event is initially very small but increases with more paths.

**Experiment 3: MSE in Relation to Calculation Time**

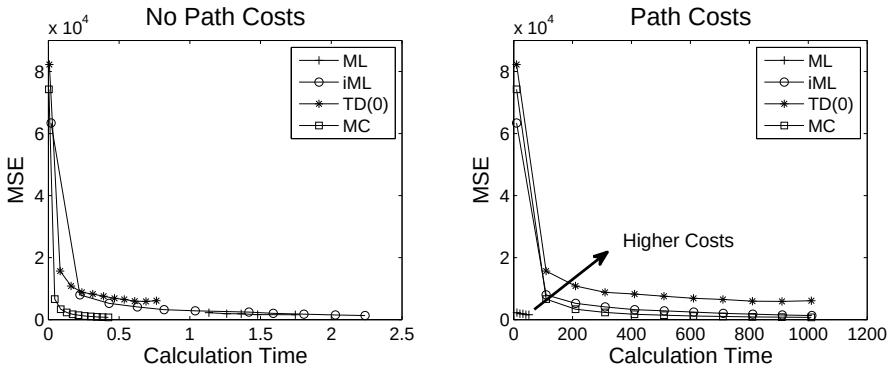


Figure 2.6: MSE in relation to the computation time of the ML, iML, TD(0) and MC estimator. The left plot shows pure computation time (we excluded computation time for MRP calculations like state changes). In the right plot, an extra factor for each observed path is included (one second per path). The state space consisted of 10 layers with 20 states per layer. We tracked for a given number of paths (ML: 10-50, iML, TD(0), MC: 10-1000) the MSE and the computation time. The plot was constructed with the mean values for every number of paths.

In many practical cases the convergence speed per sample is not the important measure. It is the convergence speed per time that is important. The time needed for reaching a specific MSE level consists of the MSE for a given number of paths, the costs to calculate the estimate from the sample, and the costs for generating the paths. We constructed an experiment to evaluate this relation (Figure 2.6). We first tested which estimator is superior if only the pure estimator computation time is regarded (left part). For this specific MRP the MC estimator converges fastest in dependence of time. The rate for starts in state $s$ was 0.2, which is an advantage for MC. The ratio will typically be much lower. The other three estimators seem to be more or less equivalent. In the second plot a constant cost of 1 was introduced for each path. Through this the pure computation time becomes less important while the needed number of paths for reaching a specific MSE level becomes relevant. As ML needs only very few paths, it becomes superior to the other estimators. Further, iML catches up on MC. For higher costs the estimators will be drawn further apart from ML (indicated by the arrow). The simulations suggest that MC or TD (dependent on the MRP) are a good choice if the path costs are low. For higher costs ML and iML are alternatives.

### 2.4.2 Cyclic MRPs: MVU - ML Comparison

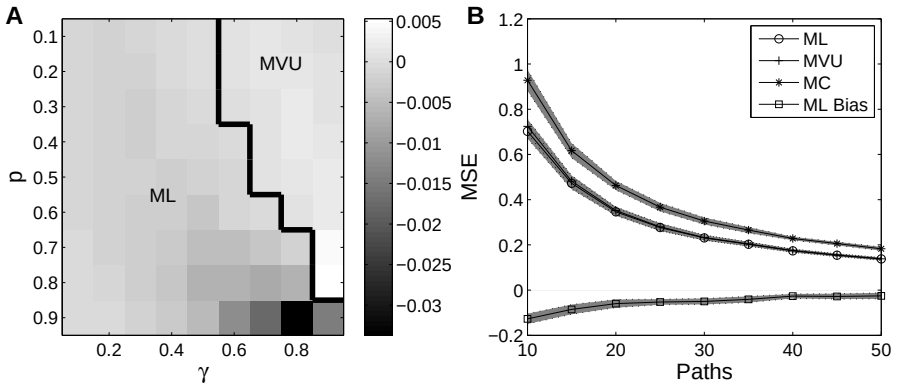


Figure 2.7: **A**: The plot shows the difference in MSE between the ML estimator and the MVU (MSE(ML)- MSE(MVU)) for 10 paths and different values of $\gamma$ and $p$. In the top right part the MVU is superior and in the remaining part the ML estimator. **B**: The plot shows the MSE of the ML, the MVU and the MC estimator and the bias of ML in dependence of the number of paths for $p = \gamma = 0.9$. 30 000 samples were used for the mean and the standard deviation (30 blocks with 1000 examples).

Calculating the MVU is infeasible without some algebraic rearrangements. Yet, the algebraic rearrangements get tricky, even for simple MDPs. We therefore restrict the comparison of the MVU and the ML estimator to the simplest possible cyclic MDP, i.e. the MDP from Figure 2.3 (A). The MC and ML value estimates are

$$\frac{1}{n}\sum_{u=1}^{n}\sum_{j=0}^{i_u}\gamma^{i_u} = \frac{1}{1-\gamma} - \frac{1}{n}\sum_{u=1}^{n}\frac{\gamma^{i_u+1}}{1-\gamma} \quad \text{and}$$

$$(1-\bar{p})\sum_{i=0}^{\infty}\frac{1-\gamma^{i+1}}{1-\gamma}\bar{p}^i = \frac{1}{1-\gamma} - \frac{\gamma}{1-\gamma}\frac{1-\bar{p}}{1-\gamma\bar{p}},$$

where $i_u$ denotes the number of times the cycle has been taken in run $u$. The MVU sums the MC estimates over all consistent sets, i.e. over all vectors $(k_1, \ldots, k_n)$ which fulfill $\sum_{u=1}^{n} k_u = s := \sum_{u=1}^{n} i_u$. With the normalization $\mathcal{N}$ being the size of this set the MVU is

$$\frac{1}{n\mathcal{N}}\sum_{(k_1,\ldots,k_n)=s} MC(k_1) + \ldots + MC(k_n)$$

$$= \frac{1}{n\mathcal{N}}\sum_{k_1=0}^{s}\ldots\sum_{k_{n-1}=0}^{s-k_1\ldots-k_{n-2}} MC(k_1) + \ldots + MC(k_n).$$

The number of times $k_u$ takes a value $j$ is independent of $u$, i.e. MC($k_1$) appears equally often as MC($k_n$) if $k_1 = k_n$. Therefore, it is enough to consider $MC(k_1)$

and the MVU is

$$\frac{1}{\mathcal{N}} \sum_{k_1=0}^{s} MC(k_1) \sum_{k_2=0}^{s-k_1} \dots \sum_{k_{n-1}=0}^{s-k_1 \dots - k_{n-2}} 1 =: \frac{1}{\mathcal{N}} \sum_{k_1=0}^{s} MC(k_1) \mathcal{C}(k_1).$$

Finally, the coefficient is $\mathcal{C}(k_1) = \binom{s+n-2-k_1}{n-2}$ and the normalization is $\mathcal{N} = \binom{s+n-1}{n-1}$. The derivation can be done in the following way. First, observe that $1 = \binom{k_n}{0}$. Then, that $\sum_{k_{n-1}=0}^{s-k_1 \dots - k_{n-2}} \binom{k_n}{0} = \binom{1+(s-k_1 \dots - k_{n-2})}{1}$ (e.g. rule 9 in [3] on page 13). And finally that

$$\sum_{k_{n-2}=0}^{s-k_1 \dots - k_{n-3}} \binom{1 + (s - k_1 \dots - k_{n-2})}{1} = \sum_{k_{n-2}=0}^{s-k_1 \dots - k_{n-3}} \binom{1 + k_{n-2}}{1}.$$

Iterating these steps derives the normalization and the coefficients. In summary the MVU is given by

$$\frac{1}{1-\gamma} - \frac{1}{(1-\gamma)\binom{s+n-1}{n-1}} \sum_{i=0}^{s} \binom{s+n-2-i}{n-2} \gamma^i. \tag{2.11}$$

We compared the MVU with the ML estimator in two experiments. The results are shown in Figure 2.7. One can observe in Figure 2.7 (A) that high probabilities for cycles are beneficial for ML and that the discount which is most beneficial to ML depends on the probability for the cycle. We have seen in Section 2.3.4 that the Bellman equation enforces the estimator to use all cycle times from 0 to "$\infty$" and thus in a sense "overestimates" the effect of the cycle. Furthermore, the probability for the cycle is underestimated by ML, i.e. $\mathbb{E}[\bar{p}] < p$ (Section 2.3.4), which can be seen as a correction for the "overestimate". The parameter estimate is independent of the true probability and the discount. Therefore, a parameter must exist which is most beneficial for ML, i.e. ML is biased towards this parameter. The experiment suggests that the most beneficial parameter $p$ is close to 1, meaning that ML is biased to systems with high probabilities for cycles.

In Figure 2.7 (B) the results of the second experiment are shown. In this experiment $\gamma = p = 0.9$ and the number of paths is varied. One can observe that the difference between the ML and the MVU estimator is marginal in comparison to the difference to the MC estimator. Furthermore, the bias of ML approaches quickly to 0 and the MVU and the ML estimator become even more similar.

### 2.4.3 Contraction: ML, iML and TD(0)

In a final set of experiments we compared the contraction factor of different operators. We generated randomly transition matrices for a state space size of 100 and applied the different operators. The results are shown in Figure 2.8. The left plot shows the results for the usual Bellman operator and the bound for different discount values. In the middle the TD(0) update equation is used and in the right plot the Bellman operator is applied state wise, whereas the state is chosen randomly from different priors. The prior probabilities for states $1, \dots, n := |\mathbb{S}|$ are given by: $p_1 = (1-c)m, p_2 = (1-c+1/(n-1))m, \dots, p_n =$

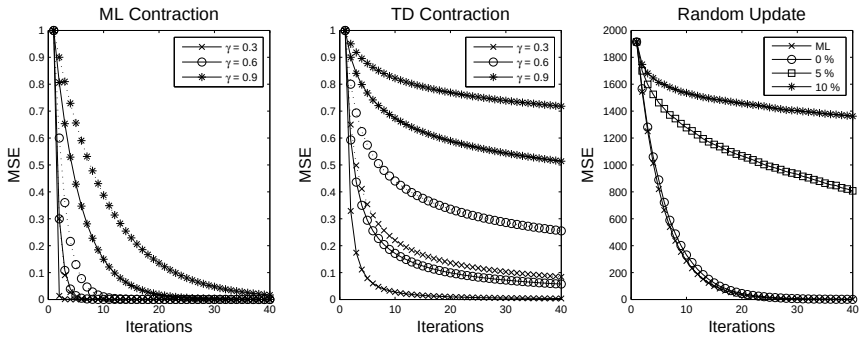


Figure 2.8: The three plots show the contraction rate of different operators to the ML solution. The $x$-axis denotes the number of applications of the operators and the $y$-axis shows the distance to the ML solution. The left and the center plot are normed with the initial distance (before the first application). **Left:** The Bellman operator is used. The discount $\gamma$ varies from 0.3 to 0.9. For each discount value the empirical distance and the bound (dotted line) is plotted. **Center:** Same setting as in the left plot but with the "TD(0)" operator. **Right:** In this plot $\gamma = 0.9$. The ML curve corresponds again to the Bellman operator. For the other three curves only single states are updated with the Bellman operator, whereas the states which are updated are chosen randomly. The percent values denote the deviation from the uniform prior for the states (0% means uniform). For the single state curves not one update was performed per iteration but $|\mathbb{S}|$ many.

$(1+c)m$, where $m = 1/n$ (mean) and $c$ denotes the deviation from the uniform prior. If $c = 0$ then we got a uniform distribution. If $c = 0.1$ then $p_1 = 0.9m, p_2 = (0.9 + 1/(n-1))m, \ldots, p_n = 1.1m$.

While we were not able to proof that TD is in general inferior to ML, respectively to iML the plots suggest this to be the case for typical MRPs. Especially, the contraction of TD (middle plot) to the ML solution is magnitudes slower than the contraction using the Bellman operator. The state-wise update reduces the contraction speed further. The right plot shows the difference between the fixed point iteration and the state-wise update with the Bellman operator (corresponding to iML). The contraction factor of the state-wise update depends crucially on the distribution of visits of the different states. At best (i.e. uniform distribution over the states) the contraction is about $|\mathbb{S}|$-times slower than the contraction with the Bellman operator applied to the full state vector.

## 2.5 Summary

In this work we derived the MVU and compared it to different value estimators. In particular, we analyzed the relation between the MVU and the ML estimator. It turned out that the relation between these estimators is directly linked to the relation between the class of unbiased estimators and estimators that fulfill the Bellman equation. If the ML estimator is unbiased then it is equivalent to the

MVU and more generally the difference between the estimators depends on the bias of ML. This relation is interesting, in particular as the estimators are based onto two very different algorithms and proving equivalence using combinatorial arguments is a challenging task. Furthermore, we demonstrated in this work that the MC estimator is equivalent to the MVU in the undiscounted case if both estimators got the same amount of information. The relation to TD is harder to characterize. TD is essentially unbiased in the acyclic case and therefore inferior to the MVU and the ML estimator in this case. In the cyclic case TD is biased and our tools are not applicable.

We want to conclude the section with open problems. Possibly, the most interesting problem is the derivation of an efficient MVU algorithm. The combinatorial problems that must be solved appear to be formidable. Therefore, it is astonishing that in the undiscounted case the calculation essentially boils down to calculating the ML estimate. In particular, the exponential runtime of a brute force MVU algorithm which is intractable even for simple MRPs decreases in this case to an $\mathrm{O}(n^3)$ factor. This efficiency is mainly due to the irrelevance of the time at which a reward is observed. In the discounted case the time of an observation matters and the algorithmical difficulties increase considerably. Instead of the full geometric series of ML with arbitrary long paths it seems to be needed to make a cutoff at a maximum number of cycles, i.e. replacing $(\mathbf{I}-\gamma\bar{\mathbf{P}})^{-1}$ with something like $(\mathbf{I}-\gamma\bar{\mathbf{P}}^s)(\mathbf{I}-\gamma\bar{\mathbf{P}})^{-1}$. Yet, Equation 2.11 shows that a weighting factor is associated with each time step and the MVU equation is not that simple.

Another interesting question concerns the bias of the ML estimator. We showed that the normalizations $\{N_i \geq 1\}$ are the reason for the bias. Furthermore, if the Full Information Criterion applies then the normalization problem is not present and we used a theorem from [69] to deduce unbiasedness of ML for this case. Yet, there seems to be a deeper reason for the unbiasedness of the ML estimator and the theorem from [69] appears to be an implication from this and from Corollary 4 (MVU=MC).

### 2.5.1 Discussion

In the discussion section we address two questions: (1) What is the convergence speed of the MVU? (2) Which estimator is to be preferred in which setting? In this section the emphasis is put onto gaining intuition and not on mathematical rigor.

**Convergence Speed** We are interested in the MSE and in the small deviation probability of the MVU. First, let us state the variance and the Bernstein inequality (e.g. [54]) for the first-visit MC estimator with $n$ paths available for estimation:

$$\mathbb{V}[\bar{V}^{(n)}] = \mathrm{MSE}[\bar{V}^{(n)}] = \frac{1}{n}\mathbb{V}[R] \quad \text{and}$$
$$\mathbb{P}(|\bar{V}^{(n)} - V| \geq \epsilon) \leq 2\exp\left(-\frac{\epsilon^2 n}{2\mathbb{V}[R] + 2d\epsilon/3}\right),$$

where $\mathbb{V}[R]$ is the variance in the cumulated reward (see [72] for the variance of an MRP) and $d$ is an upper bound for the cumulated reward of any path, i.e. $|\sum R_t - V| < d$.

How about the MVU? In the undiscounted case the MVU has the same variance and small deviation probability if the Full Information Criterion applies. The quality increases with further paths into the graph of successor states. Intuitively, the improvement in quality depends on the "distance" of the entry point $s'$ in the successor state graph to the state $s$ of which we want to estimate the value. A natural distance measure for this setting is the probability to move from state $s$ to $s'$. Furthermore, the improvement will depend on the variation in the cumulative reward of paths starting in $s'$. Paths, that run through regions in which the reward has high variance will yield a better performance increase than paths which run through near deterministic regions. The performance will, however, be lower bounded by the case that all of these $N$ paths start directly in $s$. Therefore, for undiscounted MRPs the rough lower bound $(1/N)\mathbb{V}[R]$ will hold:

$$\frac{1}{N}\mathbb{V}[R] \leq \mathrm{MSE}[\mathbb{E}[\bar{V}^{(n)}|\mathcal{S}]] \leq \frac{1}{n}\mathbb{V}[R].$$

If starts in the successor graph are $c$ times more often than starts in s, i.e. $N = cn$ then

$$\frac{1}{c}\mathrm{MSE}[\bar{V}^{(n)}] \approx \mathrm{MSE}[\mathbb{E}[\bar{V}^{(n)}|\mathcal{S}]].$$

Similarly, a "reasonable" Bernstein bound of the small deviation probability will lie between

$$2\exp\left(-\frac{\epsilon^2 N}{2\mathbb{V}[R] + 2d\epsilon/3}\right) = 2\exp\left(-\frac{c\epsilon^2 n}{2\mathbb{V}[R] + 2d\epsilon/3}\right) \quad \text{and}$$
$$2\exp\left(-\frac{\epsilon^2 n}{2\mathbb{V}[R] + 2d\epsilon/3}\right).$$

**Choosing an Estimator** Our study shows that we got essentially a tradeoff between computation time and convergence speed per sample. As one would expect, the methods which converge faster have a higher computation time. It seems that the fast methods with bad convergence speed are superior if we consider pure computation time (Experiment 3, Section 2.4.1). However, if there are costs involved for producing examples, then the expansive methods become competitive. In a high cost scenario it currently seems best to choose the ML/LSTD estimator. The MVU might become an alternative, but an efficient algorithm is currently missing. Furthermore, the algorithmic problems restricted the numerical comparison to ML and it is unclear in which setting which estimator is superior.

## 2.A Unbiased TD(λ)

In this section we introduce a (minorly) modified TD(λ) estimator. The estimates are, in contrast to the standard TD(λ) estimator, independent of the initialization. In the acyclic case this is already enough to guarantee unbiasedness of TD(λ). We first discuss the TD(0) case. This case contains the major arguments in an accessible form.

### 2.A.1 TD(0)

We first restate the TD(0) equation through unfolding the recursive definition (eq. 2.3, p. 15).

**Lemma 12.** *If the TD(0) estimator is initialized with 0 then for an acyclic MRP it equals*

$$\bar{V}_s^{(n)} = \sum_{i=1}^{n} \beta_i R^{(i)} + \sum_{s' \in \mathbb{S}} \Big( \sum_{i=1}^{n} T_{i,s'} \beta_i \gamma \bar{V}_{s'}^{(i-1)} \Big),$$

*where* $\beta_i := \Big( \alpha_i \prod_{j=i+1}^{n} (1-\alpha_j) \Big)$, $R^{(i)}$ *is the received reward in path* $i$ *and* $T_{i,s'}$ *a random variable which is one if in run* $i$ *the state* $s'$ *followed upon state* $s$ *and is zero otherwise.*

*Proof.* The recursive TD(0) definition (eq. 2.3) can be written as: $\bar{V}_s^{(n)} = \bar{V}_s^{(n-1)}(1-\alpha_n) + \alpha_n(R^{(n)} + \gamma \bar{V}_{s'}^{(n-1)})$. Substituting $\bar{V}_s^{(n-1)}$:

$$\begin{aligned}
\bar{V}_s^{(n)} =& \big(\gamma \bar{V}_s^{(n-2)}(1-\alpha_{n-1}) + \alpha_{n-1}(R^{(n-1)} + \gamma \bar{V}_{s''}^{(n-2)})\big)(1-\alpha_n) \\
&+ \alpha_n(R^{(n)} + \gamma \bar{V}_{s'}^{(n-1)}) \\
=& \gamma \bar{V}_s^{(n-2)}(1-\alpha_{n-1})(1-\alpha_n) + \alpha_{n-1}(1-\alpha_n)(R^{(n-1)} + \gamma \bar{V}_{s''}^{(n-2)}) \\
&+ \alpha_n(R^{(n)} + \gamma \bar{V}_{s'}^{(n-1)}) = \dots \\
=& \sum_{i=1}^{n} \Big( \alpha_i \cdot \prod_{j=i+1}^{n} (1-\alpha_j) \Big) \Big( R^{(i)} + \gamma \bar{V}_{s^{(i-1)}}^{(i-1)} \Big) =: \sum_{i=1}^{n} \beta_i \Big( R^{(i)} + \gamma \bar{V}_{s^{(i-1)}}^{(i-1)} \Big).
\end{aligned}$$

□

The estimate contains the values $\bar{V}_{s'}^{(0)}$ which bias the estimator towards the initialization. The estimator can be made unbiased for acyclic MRPs by excluding these values and by guarantying that the $\beta_i$ sum to one. With the following simple modification[2] we exclude the initialization values and we guarantee that the sum is one:

**Modification 1** Modified TD(0)

**if** $\bar{V}_s^{(i)}$ has seen no example **then**
  set the learning rate for this step to 1.
**end if**
**if** $\bar{V}_{s'}^{(i)}$ has seen no example **then**
  first update the estimate $\bar{V}_{s'}^{(i)}$
**end if**
Use the TD(0) update rule (2.3).

Setting the learning rate for the first example to 1 eliminates the initialization of $V_s$ itself. The second rule assures that the initialization of the estimators of

[2]The modification is easy to implement with a $TD(\lambda)$ algorithm. $\lambda$ is set to 0 if the successor state is initialized and it is set to 1 if the successor state is not. Further, the initial learning rate must be 1.

the successor states is eliminated. Setting the learning rate $\alpha_1$ to 1 has also the effect that the weighting factors $\beta_i$ sum to one, independent of the learning rate. For example for n=3, we have $\sum_{i=1}^{3}\beta_i = 1(1-\alpha_2)(1-\alpha_3)+\alpha_2(1-\alpha_3)+\alpha_3 = 1$.

**Theorem 13.** *The modified TD(0) estimator is unbiased if the MRP is acyclic.*

*Proof.* We prove this by induction. We start with the terminal states for which $\bar{V}_s = 0 = V_s$ holds. The induction step considers now the states which have only successors that have already been handled. This way the complete state space will be addressed. The expectation has the form (Lemma 12):

$$\mathbb{E}\Big[\sum_{i=1}^{n}\beta_i R^{(i)} + \sum_{s'\in\mathbb{S}}\Big(\sum_{i=1}^{n} T_{i,s'}\beta_i\gamma\bar{V}_{s'}^{(i-1)}\Big)\Big|K_s = n\Big] =$$
$$\mathbb{E}\Big[\sum_{i=1}^{n}\beta_i R^{(i)}\Big|K_s = n\Big] + \sum_{s'\in\mathbb{S}}\sum_{i=1}^{n}\beta_i\gamma\mathbb{E}\Big[T_{i,s'}\bar{V}_{s'}^{(i-1)}\Big|K_s = n\Big] =$$
$$\mathbb{E}\Big[\sum_{i=1}^{n}\beta_i R^{(i)}\Big|K_s = n\Big] + \sum_{s'\in\mathbb{S}} p_{ss'}\sum_{i=1}^{n}\beta_i\gamma\mathbb{E}\Big[\bar{V}_{s'}^{(i-1)}\Big|K_s = n\Big].$$

It remains to show that $\mathbb{E}\Big[\bar{V}_{s'}^{(i-1)}\Big|K_s = n\Big]$ is unbiased. For $i \geq 2$ this follows from the induction hypothesis. For the case $i = 1$ Modification 2 guarantees that the estimator has at least one example for estimation and is unbiased due to the induction hypothesis. Further, the $\beta_i$'s sum to one due to the modification and $\sum_{s'\in\mathbb{S}} p_{ss'}\gamma V_{s'}\sum_{i=1}^{n}\beta_i = \sum_{s'\in\mathbb{S}} p_{ss'}\gamma V_{s'}$. □

### 2.A.2 TD(λ)

The TD(λ) case is essentially the same. The main difference is that the estimates of all states of a path are used. Therefore, it is not enough that the estimators of the direct successor states are set to "reasonable" values, but all states of the path must be:

**Modification 2** Modified TD(λ)

**if** $\bar{V}_s^{(i)}$ has seen no example **then**
  set the learning rate for this step to 1.
**end if**
**if** for a successor $s'$ in the path $\bar{V}_{s'}^{(i)}$ has seen no example **then**
  first update the estimate $\bar{V}_{s'}^{(i)}$
**end if**
Use the TD(λ) update rule.

**Theorem 14.** *The modified TD(λ) estimator is unbiased if the MRP is acyclic.*

*Proof.* Proof by induction. **Induction Hypothesis:** $\mathbb{E}[\bar{V}_s|K_s = n] = V_s$.
**Induction Basis:** For terminal states the Hypothesis trivialy holds.
**Induction Step:** Let $\pi(i,j)$ be state $i$ in path $j$ and let $R_{\pi(i,n)}$ be the reward received at state $j$ in run $i$. In the acyclic case TD(λ) can be written as

$$\bar{V}_s^{(n)} = (1-\alpha_n)\bar{V}_s^{(n-1)} + \alpha_n \left( \sum_{i=1} (\gamma\lambda)^{i-1} R_{\pi(i,n)} + \gamma(1-\lambda) \sum_{i=2} (\gamma\lambda)^{i-2} \bar{V}_{\pi(i,n)} \right)$$
$$= (1-\alpha_1)\bar{V}_s^{(0)} + \sum_{j=1}^{n} \beta_j \left( \sum_{i=1} (\gamma\lambda)^{i-1} R_{\pi(i,j)} + \gamma(1-\lambda) \sum_{i=2} (\gamma\lambda)^{i-2} \bar{V}_{\pi(i,j)} \right).$$

We suppressed the "iteration" index of $\bar{V}_{\pi(i,j)}$ for readability. Like in the TD(0) case $\beta_j := \left(\alpha_j \prod_{k=j+1}^{n} (1-\alpha_k)\right)$. Applying the expectation operator and using $\alpha_1 = 1$, we get

$$\mathbb{E}[\bar{V}_s^{(n)} | K_s = n] \tag{2.12}$$
$$= \mathbb{E}\left[ \sum_{j=1}^{n} \beta_j \left( \sum_{i=1} (\gamma\lambda)^{i-1} R_{\pi(i,j)} + \gamma(1-\lambda) \sum_{i=2} (\gamma\lambda)^{i-2} \bar{V}_{\pi(i,n)} \right) \middle| K_s = n \right]$$
$$= \sum_{j=1}^{n} \beta_j \left( \sum_{i=1} (\gamma\lambda)^{i-1} \mathbb{E}[R_{\pi(i,j)} | K_s = n] + \gamma(1-\lambda) \sum_{i=2} (\gamma\lambda)^{i-2} \mathbb{E}[\bar{V}_{\pi(i,j)} | K_s = n] \right). \tag{2.13}$$

Instead of $\mathbb{E}[R_{\pi(i,j)}]$ and $\mathbb{E}[\bar{V}_{\pi(i,j)}]$ we use $\mathbb{E}[R_i]$ and $\mathbb{E}[\bar{V}_i]$ in the following to denote the expected reward in step $i$, respectively the expected value estimate in step $i$ (expected state times expected value estimate for that state). Due to the induction hypothesis

$$\mathbb{E}[\bar{V}_i | K_s = n] = V_i = \sum_{j=i} \gamma^{j-i} \mathbb{E}[R_j].$$

Substituting this term into equation 2.13:

$$\sum_{j=1}^{n} \beta_j \left( \sum_{i=1} (\gamma\lambda)^{i-1} \mathbb{E}[R_i] + \gamma(1-\lambda) \sum_{i=2} (\gamma\lambda)^{i-2} \sum_{j=i} \gamma^{j-i} \mathbb{E}[R_j] \right).$$

Taking a specific $\mathbb{E}[R_i]$, we see that for the coefficient

$$(\gamma\lambda)^{i-1} + \gamma(1-\lambda)(\gamma^{i-2}\lambda^{i-2} + \ldots + \gamma^{i-2}\lambda + 1)$$
$$= \gamma^{i-1} \left( \lambda^{i-1} + (1-\lambda) \sum_{j=0}^{i-2} \lambda^j \right) = \gamma^{i-1} \left( \lambda^{i-1} + (1-\lambda) \frac{1-\lambda^{i-1}}{1-\lambda} \right) = \gamma^{i-1}$$

holds. We know already that the $\beta_j$ sum to one. Hence, the modified TD($\lambda$) is unbiased. □

## 2.B Proofs

### 2.B.1 Unbiased Estimators - Bellman Equation

**Theorem 1** *For any estimator $\bar{\mathbf{P}}$ there exists an MRP and a $\gamma \in (0,1)$ such that $V(\bar{\mathbf{P}})$ is biased.*

*Proof.* We use the MRP from Figure 2.3 (A) with the reward setting $R_{11} := 0$ and $R_{12} := 1$, $n = 1$ and $\bar{p}$ being a parameter estimator. For $V(\bar{p})$ to be unbiased, it must hold that

$$\mathbb{E}\left[\sum_{i=0}^{\infty}\gamma^i\bar{p}^i\right] = \sum_{i=0}^{\infty}\gamma^i p^i \Rightarrow \sum_{i=0}^{\infty}\gamma^i(\mathbb{E}[\bar{p}^i] - p^i) = 0.$$

If the equality holds for all $\gamma \in (0,1)$, then $\mathbb{E}[\bar{p}^i] = p^i$. Otherwise, with $x_i := \mathbb{E}[\bar{p}^i] - p^i$ and $x_n$ being the first term different from 0: $|\gamma^n x_n| = |\sum_{i=n+1}^{\infty}\gamma^i x_i|$ and therefore $|x_n| = |\gamma\sum_{i=n+1}^{\infty}\gamma^{i-(n+1)}x_i|$. The sequence $|x_i|$ is bounded, i.e. $|x_i| < 2$ for all $i$ because $|\mathbb{E}[\bar{p}^i] - p^i| \leq |\mathbb{E}[\bar{p}^i]| + |p^i| \leq 2$. Hence,

$$|\gamma\sum_{i=n+1}^{\infty}\gamma^{i-(n+1)}x_i| \leq \gamma\sum_{i=n+1}^{\infty}\gamma^{i-(n+1)}2 = \frac{2\gamma}{1-\gamma}.$$

With $\gamma := |x_n|/(2-|x_n|)$ the term and the remaining part of the sum becomes smaller than $|x_n|$. This contradicts the assumption. Hence, $\mathbb{E}[\bar{p}^i]$ must equal $p^i$ for all $i$.

Hence, $p = \mathbb{E}[\bar{p}], \mathbb{E}[\bar{p}^2] = p^2 = \mathbb{E}[\bar{p}]^2, \ldots, \mathbb{E}[\bar{p}^i] = p^i = \mathbb{E}[\bar{p}]^i$ and therefore all central moments of $\bar{p}$ are zero. Consequently, $\bar{p}$ must be a constant. Otherwise, we would get a contradiction with the following argument: The possible values of $\bar{p}$ are countable (countable many outcomes). We denote the values with $a_i$ and with $q_i$ the probabilities for the values $a_i$. From $\mathbb{E}[\bar{p}^2] = \mathbb{E}[\bar{p}]^2$ it follows that $\sum_{i=0}^{\infty}q_i a_i^2 = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty}q_i q_j a_i a_j \Rightarrow \sum_{i=0}^{\infty}\sum_{j\neq i}q_i q_j a_i a_j = 0$. Furthermore, $q_i, a_i >= 0$ for all $i$ and therefore $q_i q_j a_i a_j = 0$ for all $i \neq j$. As a consequence, there can be only one $a_i > 0$. Because of unbiasedness it holds that $a_i = p/q_i$ and $q_i a_i^2 = p^2 = a_i^2 q_i^2$. This is impossible as $q_i \neq 0$.

The only possibility is now that $\bar{p}$ is a constant with $\bar{p} = p$. This, however, is not a well defined parameter estimator. □

### 2.B.2 Markov Reward Process

**Lemma 15.** *A MRP with finite state space and iid sequences forms an s-dimensional exponential family, where s is the number of free MRP parameters.*

*Proof.* A family $\{P_\theta\}$ of distributions is said to form an s-dimensional exponential family if the distributions $P_\theta$ have densities of the form

$$p_\theta(x) = \exp\left(\sum_{i=1}^{s}\eta_i(\theta)T_i(x) - A(\theta)\right)h(x) \tag{2.14}$$

with respect to some common measure $\mu$ [51]. Here, the $\eta_i$ and $A$ are real-valued functions of the parameters, the $T_i$ are real-valued statistics and $x$ is a point in the sample space. The $\eta$'s are called *natural parameters*. It is important that the natural parameters are not functionally related. In other words no $f$ should exist with $\eta_2 = f(\eta_1)$. Otherwise, the family forms only a *curved exponential family* [51]. A curved exponential family is not complete. Firstly, we demonstrate that the transition distribution forms an exponential family. The density can be written as

$$\mathbb{P}(X_1 = \pi^{(1)}, \ldots, X_l = \pi^{(l)}) = \prod_{i=1}^{\infty}P_{\pi_i}^{c_i} = \exp\Big(\sum_{i=1}^{\infty}c_i\log P_{\pi_i}\Big),$$

with $\pi^{(i)}$ being the observed paths, $(\pi_i)_{i\in\mathbb{N}}$ the set of paths, $c_i$ the number of times path $i$ has occurred and $P_\pi$ the probability of path $\pi$. The parameters $P_\pi$ are redundant. We explore now the MRP structure to find natural parameters that are not functionally dependent. The size of this set of parameters is the number of necessary MRP parameters, that is

$$\sharp\text{Starting States} - 1 + \sum_{i\in\mathbb{S}} \big(\sharp\text{Direct Successors of } i - 1\big).$$

We reformulate the exponential expression to reduce the number of parameters. First, one can observe that $\prod_{i=1}^{\infty} P_{\pi_i}^{c_i}$ is equivalent to $\prod_{i\in\mathbb{S}} \left(p_i^{n_i} \prod_{j\in\mathbb{S}} p_{ij}^{\mu_{ij}}\right)$, where $n_i$ is the number of starts in state $i$. The parameters are still redundant: Let state 1 be a starting state and $S$ the remaining set of starting states, then $p_1 = 1 - \sum_{j\in S} p_j$. Furthermore, we got one redundant parameter $p_{ij}$ for every state $i$. The first problem can be overcome by using $A(\theta)$ in the following way: $n \log\left(1 - \sum_{i\in S} p_i\right) + \sum_{i\in S} n_i \log \frac{p_i}{\left(1-\sum_{j\in S} p_j\right)}$. Here, $A(\theta)$ equals the $n$ term and $n_i$ is the number of starts in state $i$. Using the same approach for the transition parameters results in $K_i \log\left(1 - \sum_{j\in S(i)} p_{ij}\right) + \sum_{j\in S} \mu_{ij} \log \frac{p_{ij}}{\left(1-\sum_{u\in S(i)} p_{iu}\right)}$, with $S(i)$ being the set of successor states of $i$ without the first successor. This time the $K_i$ term cannot be moved into $A(\theta)$, as $K_i$ is data dependent. This problem can be overcome by observing that $K_i = n_i + \sum_{j\in\mathbb{S}} \mu_{ji}$ and by splitting the $K_i$ terms. As a result we get

$$\exp\Bigg( n \log\left(1 - \sum_{i\in S} p_i\right)\left(1 - \sum_{u\in S(1)} p_{1u}\right) + \sum_{i\in S} n_i \log \frac{p_i\left(1 - \sum_{u\in S(i)} p_{iu}\right)}{\left(1 - \sum_{j\in S} p_j\right)}$$
$$+ \sum_i \sum_{j\in S(i)} \mu_{ij} \log \frac{p_{ij}\left(1 - \sum_{u\in S(j)} p_{ju}\right)}{\left(1 - \sum_{u\in S(i)} p_{iu}\right)}\Bigg).$$

If the reward is deterministic and the examples consist of state sequences, then the MRP forms an exponential family. If the reward is a random variable then it depends on the distribution of this random variable. In many cases, like for the binomial or multinomial distribution, the resulting MRP still forms an exponential family. □

### 2.B.3 MVU

**Theorem 5** $\mathbb{E}[\bar{V}|\mathcal{S}]$ *converges on average and almost surely to the true value.*

*Proof.* The estimator converges on average, because $\mathbb{E}[|\mathbb{E}[\bar{V}|\mathcal{S}] - V|] \leq \mathbb{E}[|\bar{V} - V|] \overset{n\to\infty}{\longrightarrow} 0$, where $n$ denotes the number of observed paths. The inequality follows from the Rao-Blackwell Theorem, respectively from the Jensen inequality because $|\cdot|$ is convex.

It converges almost surely if the density function is bounded (e.g. discrete or gaussian reward distribution). Convergence follows from $\lim \bar{V} = V$ a.s. $\Rightarrow \mathbb{E}[\lim \bar{V}|\mathcal{S}] = V$ a.s. ([7][§15 Conditional Expectation]) and from $\mathbb{E}[\lim \bar{V}|\mathcal{S}] = \lim \mathbb{E}[\bar{V}|\mathcal{S}]$. The later equality holds due to an integral convergence theorem from [8][Theorem 15.1]. The theorem can be applied because $\bar{V}$ converges in $L^1$

and because the change in measure induced by $\mathcal{S}$ does not effect $L^1$ convergence (Bounded density and [8][Theorem 14.8]). □

### 2.B.4 ML Estimator

**Theorem 6** *The ML estimator is unbiased if the MRP is acyclic.*

*Proof.* The value function can be written as

$$V_s = \mathbb{E}[R_s] + \sum_{s' \in \mathbb{S}} \sum_{\pi \in \Pi_{ss'}} P_\pi \gamma^{|\pi|} \mathbb{E}[R_{s'}],$$

where $\Pi_{ss'}$ is the set of paths from $s$ to $s'$, $P_\pi$ the probability of path $\pi$, $|\pi|$ the length of the path and $\mathbb{E}[R_s] = \sum_{s' \in \mathbb{S}} p_{ss'} \mathbb{E}[R_{ss'}]$. The ML estimator can be written in the same form, whereas $P_\pi$ is replaced with $\bar{P}_\pi := \prod_i \bar{p}_{\pi_i \pi_{i+1}}$ and the expected reward with the reward estimator. The sample mean estimator $\bar{p}$ is unbiased and the reward estimator is unbiased if, for example, the reward distribution is multinomial or gaussian. The main problem is to show that $\bar{P}_\pi$ is unbiased, i.e. that

$$\mathbb{E}\Big[\prod_i \bar{p}_{\pi_i \pi_{i+1}} \Big| K_s = n\Big] \stackrel{?}{=} \prod_i p_{\pi_i \pi_{i+1}}.$$

The last of these estimators (denote it with $p_{\hat{s}\hat{\hat{s}}}$) is conditionally independent of the others given the number of visits of state $\hat{s}$ ($K_{\hat{s}}$). This is also the main point where acyclicity is needed. Using this together with the law of total probability and the fact that $\bar{p}$ is unbiased, leads to the following statement (with $L$ being the length of the path $\pi$):

$$\begin{aligned}
&\mathbb{E}\Big[\prod_{i=1}^{L-1} \bar{p}_{\pi_i \pi_{i+1}} \Big| K_s = n\Big] = \sum_{l=1}^{n} \mathbb{E}\Big[\prod_{i=1}^{L-1} \bar{p}_{\pi_i \pi_{i+1}} \Big| K_s = n, K_{\hat{s}} = l\Big] \mathbb{P}[K_{\hat{s}} = l | K_s = n] \\
&\stackrel{ind}{=} \sum_{l=1}^{n} \mathbb{E}\Big[\prod_{i=1}^{L-2} \bar{p}_{\pi_i \pi_{i+1}} \Big| K_s = n, K_{\hat{s}} = l\Big] p_{\hat{s}\hat{\hat{s}}} \mathbb{P}[K_{\hat{s}} = l | K_s = n] \\
&= p_{\hat{s}\hat{\hat{s}}} \sum_{l=1}^{n} \mathbb{E}\Big[\prod_{i=1}^{L-2} \bar{p}_{\pi_i \pi_{i+1}} \Big| K_s = n, K_{\hat{s}} = l\Big] \mathbb{P}[K_{\hat{s}} = l | K_s = n] \\
&= p_{\hat{s}\hat{\hat{s}}} \mathbb{E}\Big[\prod_{i=1}^{L-2} \bar{p}_{\pi_i \pi_{i+1}} \Big| K_s = n\Big].
\end{aligned}$$

We used that for $l = 0$ the last estimator $\bar{p}$ in the product is zero. The procedure has to be repeated for every $\bar{p}$. As a result the expectation of this estimator is equal to the path probability. One can handle the reward estimator with the same procedure. In summary we find that the value estimator is unbiased. □

### 2.B.5 Counterexample: MVU/MC - ML

We show by means of counterexamples that neither the MVU is superior to the ML estimator nor is the ML estimator superior to the MVU or to the MC estimator. We use again the MRP from Figure 2.3 (A) with the following

reward setting $R_{11} := 0$ and $R_{12} := 1$ and $n = 1$. As we showed before, the value for state 1 is $(1-p)/(1-\gamma p)$ and the ML estimate is $(1-\bar{p})/(1-\gamma\bar{p})$, where $\bar{p} = i/(i+1)$ and $i$ denotes the number of times the cyclic connection has been taken. The MC estimate and therefore the MVU estimate is given by $\gamma^i$. Because of the unbiasedness of the MVU/MC estimator the MSE is given by:

$$\begin{aligned}\mathrm{MSE}[\bar{V}_1] &= \mathbb{E}[\bar{V}_1^2] - V_1^2 = (1-p)\sum_{i=0}^{\infty}\gamma^{2i}p^i - \frac{(1-p)^2}{(1-\gamma p)^2} \\ &= \frac{p(1-p)}{(1-\gamma p)^2(1-\gamma^2 p)}(1-\gamma)^2,\end{aligned}$$

where $\bar{V}_1$ denotes the MVU/MC estimator. For the MSE of the ML estimator $\bar{\bar{V}}_1$ we need to calculate the first and the second moment. The first moment:

$$\mathbb{E}[\bar{\bar{V}}_1] = (1-p)\sum_{i=0}^{\infty}\frac{1-\bar{p}}{1-\gamma\bar{p}}p^i = (1-p)\sum_{i=0}^{\infty}\frac{1}{1+(1-\gamma)i}p^i.$$

In the following, we chose $\gamma$ such that $(1-\gamma)^{-1} = m \in \mathbb{N}$. The sum can then be written as

$$\begin{aligned}&\frac{m(1-p)}{p^m}\sum_{i=0}^{\infty}\frac{1}{m+i}p^{m+i} = \frac{m(1-p)}{p^m}\left(\sum_{i=1}^{\infty}\frac{p^i}{i} - \sum_{i=1}^{m-1}\frac{p^i}{i}\right) \\ &= \frac{m(1-p)}{p^m}\left(\ln\frac{1}{1-p} - \sum_{i=1}^{m-1}\frac{p^i}{i}\right).\end{aligned}$$

The second moment:

$$\begin{aligned}\mathbb{E}[\bar{\bar{V}}_1^2] &= (1-p)\sum_{i=0}^{\infty}\frac{(1-\bar{p})^2}{(1-\gamma\bar{p})^2}p^i = (1-p)\sum_{i=0}^{\infty}\frac{1}{(1+(1-\gamma)i)^2}p^i \\ &= \frac{(1-p)m^2}{p^m}\sum_{i=0}^{\infty}\frac{1}{(m+i)^2}p^{m+i} = \frac{(1-p)m^2}{p^m}\left(\sum_{i=1}^{\infty}\frac{p^i}{i^2} - \sum_{i=1}^{m-1}\frac{p^i}{i^2}\right).\end{aligned}$$

The infinite sum is called *Spence function* or *dilogarithm* and is denoted with $Li_2(p)$. Using these terms one can derive the MSE:

$$\begin{aligned}\frac{(1-p)^2m^2}{p^m(m(1-p)+p)}\Bigg(&\frac{m(1-p)+p}{(1-p)}\left(Li_2(p) - \sum_{i=1}^{m-1}\frac{p^i}{i^2}\right) - \frac{2}{m(1-p)}\Bigg(\ln\frac{1}{1-p} \\ &- \sum_{i=1}^{m-1}\frac{p^i}{i}\Bigg) + \frac{p^m}{(m(1-p)+p)}\Bigg).\end{aligned}$$

For $\gamma = p = 1/2$ the MSE of the MVU/MC estimator is 0.127 and 0.072 for the ML estimator. Contrary, for $p = 0.99$ the MSE of the MVU/MC estimator is 0.0129 and 0.0219 for the ML estimator.

# 2.C Supplementary Material

## 2.C.1 Convergence Proof and Unbiasedness of iML

**Theorem 16.** *iML is unbiased for acyclic MRPs. Futhermore, it converges in the average and almost surely to the true value.*

*Proof.* We start with unbiasedness. The proof is similar to the TD proof.

$$\begin{aligned}\mathbb{E}[\bar{V}_s|K_s=n] &= \mathbb{E}\Big[\bar{R}_s+\gamma\sum_{s'\in\mathbb{S}}\frac{\mu_{ss'}}{K_s}\bar{V}_{s'}^{(k(s'))}\Big|K_s=n\Big]\\ &= R_s+\frac{\gamma}{n}\sum_{s'\in\mathbb{S}}\mathbb{E}[\mu_{ss'}\bar{V}_{s'}^{(k(s'))}|K_s=n].\end{aligned}$$

We use the conditional independence between $\mu$ and $\bar{V}_{s'}$ given the number of visits to $s'$:

$$\begin{aligned}&\sum_{m=1}\mathbb{E}[\mu_{ss'}\bar{V}_{s'}^{(k_{s'})}|K_{s'}=m,K_s=n]\\ &=\sum_{m=1}\mathbb{E}[\mu_{ss'}|K_{s'}=m,K_s=n]\mathbb{E}[\bar{V}_{s'}^{(k_{s'})}|K_{s'}=m,K_s=n]\\ &=\sum_{m=1}\mathbb{E}[\mu_{ss'}|K_{s'}=m,K_s=n]V_{s'}\\ &=V_{s'}\sum_{m=0}\mathbb{E}[\mu_{ss'}|K_{s'}=m,K_s=n]=np_{ss'}V_{s'}.\end{aligned}$$

We continue with the almost sure convergence. We prove the discounted case. The undiscounted case where from each state a terminal state can be reached is nearly identical.

Almost sure convergence holds if and only if for every $\epsilon_1,\epsilon_2$ there exist an $N$ such that for $n\geq N$ $\mathbb{P}[\{\sup_{m\geq n}||\bar{\mathbf{V}}^{(m)}-\mathbf{V}||_\infty>\epsilon_1\}]<\epsilon_2$, where $||\cdot||_\infty$ denotes the sup-norm.

We start with the case that the update equation is $\bar{\mathbf{V}}^{(n+1)}=\mathbf{T}^{(n)}\bar{\mathbf{V}}^{(n)}$, where $\mathbf{T}^{(n)}:=\bar{\mathbf{r}}^{(n)}+\gamma\bar{\mathbf{P}}^{(n)}$ is a contraction with contraction factor $\gamma$. Due to the Banach fixed point theorem $(\mathbf{T}^{(n)})^l x$ converges to the unique fixed point $\bar{\mathbf{V}}^{(n)}$ for $l\to\infty$ and any $x$.

The fixed point $\bar{\mathbf{V}}^{(n)}=(\mathbf{I}-\gamma\bar{\mathbf{P}}^{(n)})^{-1}\bar{\mathbf{r}}$ depends continuously on $\bar{\mathbf{r}}$ and $\bar{\mathbf{P}}$. Continuous dependency can be observed by noting that for invertible matrices $\mathbf{A},\mathbf{B}$: $||\mathbf{A}^{-1}-\mathbf{B}^{-1}||=||\mathbf{A}^{-1}(\mathbf{B}-\mathbf{A})\mathbf{B}^{-1}||\leq||\mathbf{A}^{-1}||||\mathbf{B}^{-1}||||\mathbf{A}-\mathbf{B}||$ and that $||(\mathbf{I}-\gamma\bar{\mathbf{P}}^{(n)})^{-1}||=||\sum\gamma^i{\bar{\mathbf{P}}^{(n)}}^i||\leq\sum\gamma^i||\bar{\mathbf{P}}^{(n)}||^i=\frac{1}{1-\gamma}$, where $||\cdot||$ denotes the operator norm (using $||\cdot||_\infty$ for the elements). Hence, for every $\epsilon$ a $\delta$ exists with $||\bar{\mathbf{V}}^{(n)}-\mathbf{V}||_\infty\leq\epsilon$ if $||\bar{\mathbf{P}}^{(n)}-\mathbf{P}||\leq\delta$ and $||\bar{\mathbf{r}}^{(n)}-\mathbf{r}||_\infty\leq\delta$.

$\bar{\mathbf{P}}^{(n)}$ and $\bar{\mathbf{r}}$ converge almost surely. We chose $N$ such that for $n\geq N$ the probability of the event $C:=\{\sup_{m\geq n}||\bar{\mathbf{P}}^{(m)}-\mathbf{P}||<\tilde{\epsilon}\}\cap\{||\sup_{m\geq n}\bar{\mathbf{r}}^{(m)}-\mathbf{r}||_\infty<\tilde{\epsilon}\}$ is greater than $1-\epsilon_2$, where $\tilde{\epsilon}$ is chosen such that for $m\geq N$: $||\bar{\mathbf{V}}^{(m)}-\mathbf{V}||_\infty\leq\epsilon:=\frac{\epsilon_1}{8}(1-\gamma)$ holds. Thus for event $C$ all the $\mathbf{V}^{(m)}$ are contained in an $\frac{\epsilon_1}{8}(1-\gamma)$-ball around $\mathbf{V}$.

We now got the problem that in each step another contraction operator $\mathbf{T}^{(m)}$ is applied each with its own fixed point. However, each of the operators drags the value estimate to the ball $U_\epsilon(\mathbf{V})$. Notice that the ball $U_\epsilon(\mathbf{V})$ must not be

reached, but the estimate will be dragged into the larger ball $U_{\epsilon_1}(\mathbf{V})$. First, we observe that for $||x - U_\epsilon(\mathbf{V})||_\infty > \frac{\epsilon_1}{2}$ any $\mathbf{T}^{(m)}$ contracts towards $U_\epsilon(\mathbf{V})$ at least with $\frac{1+\gamma}{2} < 1$:

$$\begin{aligned}||\mathbf{T}^{(m)}x - \mathbf{T}^{(m)}U_\epsilon(\mathbf{V})||_\infty &\leq ||\mathbf{T}^{(m)}x - \bar{\mathbf{V}}^{(m)}||_\infty + ||\bar{\mathbf{V}}^{(m)} - \mathbf{T}^{(m)}U_\epsilon(\mathbf{V})||_\infty \\ &\leq ||\mathbf{T}^{(m)}x - \bar{\mathbf{V}}^{(m)}||_\infty + \epsilon \leq \gamma||x - \bar{\mathbf{V}}^{(m)}||_\infty + \epsilon \leq \gamma||x - U_\epsilon(\mathbf{V})||_\infty + 2\epsilon \\ &\leq \frac{\gamma+1}{2}||x - U_\epsilon(\mathbf{V})||_\infty\end{aligned}$$

as $(\frac{1+\gamma}{2} - \gamma)||x - U_\epsilon(\mathbf{V})||_\infty - 2\epsilon = \frac{1-\gamma}{2}||x - U_\epsilon(\mathbf{V})||_\infty - \frac{(1-\gamma)\epsilon_1}{4} \geq 0$. Hence, a factor $l$ exists, such that $(\mathbf{T}^{(m)})^l x \in U_{\epsilon_1}(\mathbf{V})$ for each $x$ on the event $C$ and almost sure convergence follows.

In the case of iML we update only one state in each turn, e.g. for state $i$: $\tilde{x} = \mathbf{T}_i^{(n)}x$, where $\mathbf{T}_i^{(n)}$ is for $j \neq i$ the idendity and is for $i$ given by $\tilde{x}_i = \bar{\mathbf{r}}_i^{(n)} + \gamma\bar{\mathbf{p}}_{i\bullet}^{(n)}x$. For simplicity assume that $\bar{\mathbf{r}}$ and $\bar{\mathbf{p}}$ does not change with $n$. For a sequence of operators $\tilde{\mathbf{T}} = \mathbf{T}_{i_1} \dots \mathbf{T}_{i_u}$ where each state is updated at least once, we have $||\tilde{\mathbf{T}}x - \tilde{\mathbf{T}}y||_\infty \leq \gamma||x - y||_\infty$, because for $\tilde{x} = \mathbf{T}_i x$ and $\tilde{y} = \mathbf{T}_i y$ we have $|\tilde{x}_i - \tilde{y}_i| = \gamma|\bar{\mathbf{p}}_{i\bullet}(x - y)| \leq \gamma||x - y||_\infty$. As sequences of the form $i_1 \dots i_u$ occur infinitely often with probability one the convergence in $||\cdot||_\infty$ is established. Convergence does not depend on the norm as the different norms on $\mathbb{R}^{|\mathbb{S}|}$ are equivalent. The case where $\mathbf{T}_i^{(n)}$ depends on $n$ is proved the same way as the $\mathbf{T}^{(n)}$ case.

Convergence in the average can for bounded reward functions be shown with the Lebesgue convergence theorem. Because $\bar{\mathbf{V}}_i^{(n)} \to \mathbf{V}_i$ almost surely, $\bar{\mathbf{V}}_i^{(n)} \in L^1$ and for all $n$: $|\bar{\mathbf{V}}_i^{(n)}| \leq r_{max} \in L^1$ the theorem can be applied and proves $L^1$ convergence. □

### 2.C.2 iML Algorithm

**Algorithm 3 iML**

**Procedure** main
**init:** $K$=0, $\mu = 0$, $A = 0$
**for** $i = 1$ to number of paths **do**
  **step 1:** Observe starting state s
  **step 2:** Run: estimate(s)
**end for**

**Procedure** estimate(s)
**if** $s$ is absorbing state **then**
  **return**
**end if**
**step 1:** Take one step and observe reward $r$ and successor state $s'$
**step 2:** Run: estimate(s')
**step 3:** Update estimate:

$$\bar{V}(s) \leftarrow \frac{K(s)}{K(s)+1}\Big(\bar{V}(s) - \gamma\frac{\mu(s,s')}{K(s)}A(s,s')\Big) + \frac{\gamma}{K(s)+1}\Big(\mu(s,s')\bar{V}(s') + r\Big)$$

**step 4:** Update $\mu$, $K$ and $A$:

$$\mu(s,s') \leftarrow \mu(s,s') + 1$$
$$K(s) \leftarrow K(s) + 1$$
$$A(s,s') \leftarrow \bar{V}(s')$$

CHAPTER 3

# Risk Sensitive Control: Mean-Variance Tradeoffs


## Abstract

It is common in Reinforcement Learning to choose strategies that maximize the mean without regarding the corresponding variance. This strategy, however, is undesirable in many cases as the derived policies are unreliable and the performance may have high positive and negative peaks causing fatal effects (e.g. the bankruptsy of a trading company). In operations research several variations of mean-variance tradeoffs are used to control the variance and thus, the risk. The application of this mean-variance tradeoffs in a Reinforcement setting is straight forward. However, the objectives have the drawback that a policy iteration cannot be applied and the optimization is slow, leading to problems for large state spaces. We overcome this problem by deriving a variation of the mean-variance objective that is solvable by policy iteration and thus, allows us to use common methods such as the Temporal Difference Learning. Furthermore, we derive estimators for the variance of a policy which can be used for evaluation of the associated risk. We compare the objectives and the methods in different experimental settings. We use two tasks from the series of paper by D. J. White on Markov Decision Processes with real-world applications for the evaluation.


## 3.1 Introduction

In *Reinforcement Learning (RL)*, an agent interacts with an unknown environment and attempts to maximize its cumulative reward. However, a policy that maximizes the reward is not necessarily reliable and the gained reward can vary arbitrarily over different runs. In many settings such a fluctuation is harmful and one is willing to trade in some of the expected reward for reliability. Different traders at a financial market, for example, will have different preferences over the risk that they are willing to accept. A risky trader may go for the max-

imum payoff while a conservative trader will be more satisfied with a strategy that yields smaller payoff but has high reliability. In this work we introduce objectives to RL that incorporates preferences about the risk.

Little attention has thus far been paid to such risk-sensitive objectives in RL. In contrast, in the field of operations research several studies have addressed risk-sensitive objectives for *Markov Decision Processes (MDPs)* [72, 33]. One common approach for risk-sensitive objectives is the mean-variance tradeoff, where the mean is penalized by the variance. Unlike the maximum expected reward criterion, however, the mean-variance objective leads to an optimization problem that is hard to solve. In fact, the intuitive mean-variance tradeoff that uses the variance of an MDP appears unusable [72] and one needs to fall back on alternative objectives [33].

Here, we provide an overview of the popular approaches that incorporate a measure for the variance and discuss how they can be applied to the RL setting. In general, the risk-sensitive objectives lead to non-linear programs (mostly concave programs) that cannot be solved with a policy iteration. Corresponding optimization methods for the objectives rely on all MDP parameters and thus, are practically impossible to apply to MDPs with large state spaces. We present an alternative objective by approximating the *discount-normalized variance* [33] using a Taylor series. While there is some loss in quality, the optimziation problem can be solved efficiently with standard approaches such as a value-policy iteration using the Temporal Difference estimator.

Since there are number of alternative objectives, the question is which of them is the "best", or rather, what is "best". In this work we take the intuitive mean-variance criterion as the benchmark. While the optimziation problem with this criterion is not solvable, it still is the "correct" measure of the variance. Alternative objectives, thus, should not result in strategies which results in poor performances with respect to this criterion.

If the MDP is not fully known, one must rely on estimates of the variance to analyze the risk of different policies. Variance estimators can be constructed in complete analogy to value estimators. We introduce two estimators that are straight forward to derive: (1) the Monte-Carlo variance estimator (MC) and (2) the Maximum-Likelihood estimator (ML). The ML value estimator is also known as the *Least-Squares Temporal Difference Estimator* (LSTD, [19]). We derive basic properties such as the almost sure convergence for these estimators.

In an empirical evaluation we compare the risk-sensitive objectives and methods in different MDPs. We use two MDPs from the series of papers of D. J. White [82, 83, 84] for the evaluation. In these papers White collects models that had an industrial "impact", i.e., models which had certain influence onto industrial decisions. These papers are quite dated and the models used are relatively simple, but not as arbitrary as artificially created examples.

The first scenario is the control of storage reservoirs for the Grand Coulee plant at Columbia River. The reservoirs of the hydroelectric system collect water during high river flows for use during low flows season. A model for this task is described in [52]. Data of the river flows is available online. Monthly data during 1955-2005 were used.

The second task deals with claim decisions by policyholders in car insurance systems [79]. The goal is a decision rule on filing or not filing a claim when an accident occurs according to the current risk category of the driver and the number of claims that have already been filed.

## 3.2 Preliminaries and Definitions

We begin with basic definitions. A *Markov Decision Process (MDP)* $\boldsymbol{M}$ on *states* $\mathbb{S} = \{1, \ldots, N\}$ and with *actions* $a_1, \ldots, a_k$ consists of: (1) *transition probabilities* $p_{ij}^a \geq 0$, which for any action $a$, and any states $i$ and $j$, specify the probability of reaching state $j$ from state $i$ after executing action $a$. Thus, $\sum_j p_{ij}^a = 1$ for any state $i$ and action $a$. (2) *Starting probabilities* $q_i \geq 0$, which specify the probability to start in state $i$, thus $\sum_i q_i = 1$. (3) A *payoff* or *reward* $R_{ij}$, for each transition from state $i$ to $j$. For simplicity we assume a deterministic payoff, however, the extension to a random payoff is straight forward. A *policy* in $\boldsymbol{M}$ is a mapping $\pi : \{1, \ldots, N\} \to \{a_1, \ldots, a_k\}$. An MDP $\boldsymbol{M}$, combined with a policy $\pi$, yields a *Markov Reward Process (MRP)* on the states.

It is useful to use matrix notation in discussing MDPs. State $i$ is then represented by a vector of length $N$ with all elements being zero except the $i$th which is one. The transition probabilities for policy $\pi$ are given by the stochastic matrix $\mathbf{P}^\pi$, where $\mathbf{P}_{ij}^\pi = p_{ij}^a$ and $a = \pi(i)$. Similarly, the payoff for the states is given by a vector $\boldsymbol{r}^\pi$ of length $N$, where the $i$th element is given by: $\sum_j p_{ij}^a R_{ij}$. We write $\mathbf{P}$ instead of $\mathbf{P}^\pi$ if we do not want to refer to the policy $\pi$, e.g. the policy stays the same.

The usual goal in RL is to find a policy which maximizes the expected future reward (*value*). The value of a state $i$ given a policy $\pi$ is defined as $V_\pi = \mathbb{E}_\pi[\sum_{t=0}^\infty \gamma^t R_t | s_0 = i]$, where $\mathbb{E}_\pi$ denotes the expectation with respect to the policy $\pi$, $s_0$ is the state at time $t = 0$, $R_t$ is the received reward at time $t$ and $\gamma \in (0, 1]$ is the discount factor. The average value for a given starting distribution $\boldsymbol{q}$ is denoted with $V_{\pi,q} = \mathbb{E}_{\pi,q}[\sum_{t=0}^\infty \gamma^t R_t]$. The value can also be written as a function of the transition matrix and the reward, $\boldsymbol{v} := V(\mathbf{P}^\pi, \boldsymbol{r}) := (\mathbf{I} - \gamma \mathbf{P}^\pi)^{-1} \boldsymbol{r}$, where $\mathbf{I}$ is the identity matrix. An important task in RL is the estimation of the value from observations and a number of different estimators have been proposed for this goal, including the *Monte-Carlo estimation* (MC) and the *Maximum Likelihood* (ML)/*Least-Squares TD* (LSTD, [19]).

The MC estimate is the sample mean of the cumulative discounted reward, $\frac{1}{n} \sum_{i=1}^n \left( \sum_{t=0}^\infty \gamma^t R_t^{(i)} \right)$, where $n$ is the number of paths and $R_t^{(i)}$ is the received reward in path $i$ at step $t$. The estimate converges a.s. and the estimator is unbiased if each path is used only once (*First visit MC*, [69]).

An alternative approach is to use the value equation in combination with a maximum likelihood parameter estimate $\bar{p}_{ij}^a$. The parameter estimate is the sample mean of taking transition $i \to j$ given state $i$ and action $a$. The value estimator is then given by $V(\bar{\mathbf{P}}^\pi, \boldsymbol{r})$, where $\bar{\mathbf{P}}^\pi$ is the sample mean, respectively the maximum likelihood estimate of the transition matrix. The resulting ML estimator is equivalent to the LSTD estimator [19] and the value estimate is also known as the value of the *certainty-equivalence* model. The estimator converges a.s..

## 3.3 Risk Sensitive Control

Mean-Variance tradeoffs are studied for a long time in operations research. A milestone is the work from Sobel in 1982 [72]. Sobel presented a formula for the variance of a MDP and a counter-example for the usability of the intuitive

mean-variance tradeoff. The variance is given by

$$\mathbb{V}(\mathbf{P}, R) := (\mathbf{I} - \gamma^2 \mathbf{P})^{-1} \boldsymbol{\theta}$$
$$\text{with} \quad \boldsymbol{\theta}_i := \sum_j \mathbf{p}_{ij} (R_{ij} + \gamma \boldsymbol{v}_j)^2 - \boldsymbol{v}_i^2. \tag{3.1}$$

However, the intuitive mean-variance approach, i.e., $\max_\pi V(\mathbf{P}^\pi, R) - \lambda \mathbb{V}(\mathbf{P}^\pi, R)$ cannot be used. The reason lies in the missing *monotonicity* of the variance. Having monotonicity implies that a policy which is "good" for successor states of a state $i$ is also "good" for state $i$. This does not hold for the variance. Monotonicity is related to the existence of optimal stationary policies and to the convergence of a policy improvement algorithm to an optimum. As the variance lacks the monotonicity, there is no guarantee that an optimum exists and that a policy improvement algorithm converges [72].

### 3.3.1 Discount Normalized Variance

Whilst the variance is truly meaningful as a measure of uncertainty, the corresponding optimization problem seems to make the approach inapplicable. Due to this problem "alternative" definitions of the variance have been proposed (e.g. [33]). Two important alternatives are the *stage-wise* and the *discount-normalized variance* [33]. Assuming that the rewards at different stages $t$ are independent the variance can be rewritten with the Biénayme equality as follows:

$$\mathbb{V}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t R_t \right] = \sum_{t=0}^{\infty} \gamma^t \mathbb{V}_\pi \left[ R_t \right].$$

The right hand side of the equation is the definition of the stage-wise variance. Only in trivial MDP's the $R_t$'s will be independent and therefore both definitions will in general differ.

While the objective using the stage-wise variance is appealing, the corresponding optimization problem is still problematic for two reasons. First, the algorithm in [33] requires a "product" of MDPs to solve the optimization problem and thus the state space size increases to $|\mathbb{S}|^2$. Second, as shown in the example in [33], the corresponding mean-variance cost function $C$ does not fulfill the recursive equation $C_i = c_i + \gamma \sum p_{ij} C_j$, where $c_i$ is the direct costs occuring in $i$. The problem lies in the squared bias term $\mathbb{E}_\pi[R_t]^2$. For example, for $t = 3$:

$$\begin{aligned} \mathbb{E}_1[R_t]^2 &= \left( \sum_{i_1} p_{1 i_1} \gamma \sum_{i_2} p_{i_1 i_2} \gamma \sum_{i_3} p_{i_2 i_3} R_{i_2 i_3} \right)^2 \\ &= \gamma^2 \sum_{i_1} \sum_{\tilde{i}_1} p_{1 i_1} p_{1 \tilde{i}_1} \mathbb{E}_{i_1}[R_t]^2 \mathbb{E}_{\tilde{i}_1}[R_t]^2. \end{aligned}$$

It is thus questionable whether a policy iteration can be applied.

The second alternative definition of the variance, i.e., the discount-normalized variance, is in a sense a further simplification of the stage-wise variance. As seen above, the main problem with the stage-wise variance is the mean term $\mathbb{E}_\pi[R_t]$ around which the fluctuation is measured. For the discount-normalized variance this term is replaced with the average reward $(1 - \gamma) V_{\pi,q}$. $(1 - \gamma)$ is used

to normalize the "weights" $\gamma^t$ for the rewards $R_t$, i.e. $(1-\gamma)\sum\gamma^t = 1$. The discount-normalized variance, denoted hereafter by $\mathscr{V}$, is defined as:

$$\mathscr{V}_{\pi,q} := \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{\pi,q}\left[(R_t - (1-\gamma)V_{\pi,q})^2\right] = \mathbb{E}_{\pi,q}\left[\sum_{t=0}^{\infty}\gamma^t R_t^2\right] - (1-\gamma)V_{\pi,q}^2.$$

In summary, $\mathbb{V}$ and $\mathscr{V}$ are similar if the rewards at different time steps are independent and if $(1-\gamma)V_{\pi,q}$ is a good approximation of $\mathbb{E}[R_t]$. Independence of the $R_t$'s is related to the independence between the future behavior and the current state, e.g. the probability to enter a state $u$ is independent of the current state: $p_{iu} \approx p_{ju}$ for $i, j \in \mathbb{S}$. The similarity between $(1-\gamma)V_{\pi,q}$ and $\mathbb{E}[R_t]$ depends largely on the time $t$ and on $\gamma$ as the weighting for $R_t$ is $(1-\gamma)\gamma^t$, e.g. for $\gamma = 0.1$ the coefficient is approximately $10^{-t}$. However, as the resulting term is scaled by another $\gamma^t$ to calculate $\mathscr{V}$ the difference in the mean is not that cruicial.

### 3.3.2 Taylor Approximation of the Discount Normalized Variance

The quadratic terms of the mean-variance objective introduces few undesired properties, some of which can be avoided by replacing $\mathbb{E}[(\sum_t \gamma^t R_t)^2]$ with the term $\sum_t \gamma^t \mathbb{E}[R_t^2]$ and $\mathbb{E}[R_t]^2$ with $(1-\gamma)V_{\pi,q}^2$. Yet, the associated optimization problems are still computationally expensive and moreover, a policy iteration cannot be applied. The problem arises from the $V_{\pi,q}^2$ term. Similar to $\mathbb{E}[R_t]^2$, this term introduces products $p_{i_1 i_2} p_{\tilde{i}_1 \tilde{i}_2}$ which prevent us from formulating the cost function recursively and thus from applying a policy iteration. One way to overcome this problem is a linearization of the normalization term using a first order Taylor approximation around an expansion point $V_0$, i.e.

$$V_{\pi,q}^2 \approx 2V_{\pi,q}V_0 - V_0^2.$$

Using this approximation we are nearly able to apply a policy iteration. The final problem is the average with respect to $q$. As different states have different prior "weights" $q_i$, there will be a problem with the recursive definition. The easiest way to avoid this complication is to omit averaging and use the value of the current state. We denote the corresponding approximation of the discount-normalized variance with $\mathcal{V}$. The complete approximation of the variance for a state $i$ and policy $\pi$ is given by

$$\mathcal{V}_{i,\pi} := \mathbb{E}_{i,\pi}\left[\sum_{t=0}^{\infty}\gamma^t R_t^2\right] - (1-\gamma)(2V_{i,\pi}V_0 - V_0^2).$$

The linearization of the normalization induces multiple effects. The advantages are the computation time and the possibility to formulate the cost function recursively to derive a policy iteration. The disadvantage is that the approximated value may be far off if we take an inappropriate expansion point. In particular, care must be taken that $V_0$ does not overestimate the value. Otherwise the normalization term may end up being higher than the second moment term and thus, the "variance" can be negative.

### 3.3.3 Mean-Variance Tradeoffs

Using $\mathscr{V}$, respectively $\mathcal{V}$, mean-variance tradeoffs can be defined. We begin with the mean-variance tradeoff using to the Taylor approximation, as the optimization is straight forward.

#### Taylor Approximation

The Mean-Variance Tradeoff for the Taylor approximation is given by

$$\max_{\pi} V_\pi - \lambda \mathcal{V}_\pi. \tag{3.2}$$

Rewriting this to formulate a recursive expression and dropping the index $\pi$ gives:

$$\begin{aligned} V_i - \lambda \mathcal{V}_i &= \mathbb{E}_i \left[ \sum_{t=0}^{\infty} \gamma^t (R_t - \lambda R_t^2) \right] + 2\lambda V_0 (1-\gamma) V_i - \underbrace{(1-\gamma)\lambda V_0^2}_{=:c} \\ &= \mathbb{E}_i \left[ \sum_{t=0}^{\infty} \gamma^t (1 + 2\lambda V_0 (1-\gamma) R_t - \lambda R_t^2) \right] - c. \end{aligned}$$

The constant $c$ is uneffected by the policy and can be ignored for the optimization. The first term on the right hand side of the equation is the value term with a modified reward, i.e. $\tilde{R}_t := 1 + 2\lambda V_0 (1-\gamma) R_t - \lambda R_t^2$. One can therefore treat the optimization problem like an ordinary value optimization and apply the usual methods.

#### Discount Normalized

The tradeoff corresponding to the discount-normalized variance is given by

$$\max_{\pi} V_\pi - \lambda \mathscr{V}_\pi. \tag{3.3}$$

The optimization is not as straight forward as for the Taylor approximation. Yet, in [33] it is shown that this optimization problem is well-posed, i.e, there is an optimum and the corresponding policy is stationary and deterministic. Due to [33] the optimal policy is the solution of the following quadratic optimization problem with linear constraints:

**Optimization Problem: Discount Normalized**

$$\max_x \sum_{i,a} R_{ia} x_{ia} - \lambda \sum_{i,a} R_{ia}^2 x_{ia} + \lambda (1-\gamma) \Big( \sum_{i,a} R_{ia} x_{ia} \Big)^2$$

$$\text{s.t.} \quad \sum_{i,a} (\delta_{ij} - \gamma p_{ij}^a) x_{ia} = q_j \quad \text{and} \quad x_{ia} \geq 0.$$

Here, $r_{ia} := \sum_{j=1}^{N} p_{ij}^a R_{ij}$ and $x_{ia}$ denotes the probability to select action $a$ in state $i$. The maxima of the optimization problem correspond to deterministic policies. At the extreme points for every state $i$ only one $x_{ia}$ is greater than zero and the optimal policy is simply $\pi(i) = a$ for $x_{ia} > 0$. For the optimization

problem, there are specialized and efficient algorithms (concave minimization under linear constraints, [42]). We used the standard algorithm for constraint nonlinear programming (Matlab Optimization Toolbox) which worked well for our tasks. There are different alternatives to this optimization problem. See [43] and [73] for details.

### 3.3.4 Estimation

If the MDP parameters are unknown then two common approaches are: (1) use maximum likelihood estimates of the parameters, treat the estimate of the MDP as the true MDP and apply the usual optimization methods (certainty equivalence approach); or (2) estimate the costs for the different states and use these estimates in a value-policy iteration like optimization.

The first approach has the advantage that the estimates converge fast in dependence of the number of observed paths. For example, the ML/LSTD value estimator usually massively outperforms the value estimators such as the TD($\lambda$). The disadvantage is the high computational costs as the number of parameters that must be to estimateed are in the order of $|\mathbb{S}|^2$, e.g., for 1000 states we need to keep track of about 1000 000 entries.

Similar to the first approach, the second approach has the disadvantage of slow convergence per observed path and the advantage of low computational costs. It is for example needed to keep track of only $|\mathbb{S}|$ entries.

For the discount-normalized objective only the first approach is applicable and the size of the MDP for which the optimization is computationally tractable is rather small, i.e. some thousands of states.

The Taylor approximation has got the advantage that the second approach can be applied and thus, the objective can be used for large state spaces.

## 3.4 Evaluation of the Risk: Variance Estimation

Although the variance of an MDP cannot be used for the optimization, it is still the "correct" measure of the risk associated with a policy. It is thus useful to have estimators of the variance at hand to evaluate the found solutions, might they be derived from value maximization, the discount-normalized tradeoff or completely different objectives. We introduce two variance estimators: the MC and the ML variance estimator. The latter converges faster but is only applicable for small numbers of states.

### 3.4.1 Monte Carlo Variance Estimator

The MC estimate is the sample mean of the variance of the cumulative discounted reward, where a normalization of $1/(n-1)$ is used to make it unbiased:

$$\frac{1}{n-1}\sum_{i=1}^{n}\left(\sum_{t=0}^{\infty}\gamma^t R_t^{(i)} - \bar{V}\right)^2,$$

$R_t^{(i)}$ is the received reward in path $i$ at step $t$ and $\bar{V}$ is the MC estimate of the value.

Analogous to the MC value estimator, there are different possibilities if a state is visited for multiple times in a run [69]. If each path is used only once for each estimate (*First visit MC*), then the estimator is unbiased. The proof of this property and of almost sure convergence is straight forward:

**Lemma 17.** *The First-visit MC variance estimator is unbiased and converges almost surely.*

### 3.4.2 Maximum Likelihood Variance Estimator

We use Equation (3.1) with the maximum likelihood estimates to get an estimator for the variance:

$$\mathbb{V}(\bar{\mathbf{P}}, R) = (\mathbf{I} - \gamma^2 \bar{\mathbf{P}})^{-1} \bar{\boldsymbol{\theta}} \quad \text{with} \quad \bar{\boldsymbol{\theta}}_i := \sum_j \bar{p}_{ij} (R_{ij} + \gamma \bar{\boldsymbol{v}}_j)^2 - \bar{\boldsymbol{v}}_i^2$$

where, $\bar{\boldsymbol{v}}_i$ is the ML value estimate for state $i$. It is straight forward to show the convergence:

**Lemma 18.** *The ML variance estimator $\mathbb{V}(\bar{\mathbf{P}}, R)$ converges almost surely.*

*Proof.* The sample mean parameter estimates converge a.s. due to the strong law of large numbers [7], i.e. $\bar{p}_{ij} \to p_{ij}$ (a.s.). That means $\mathbb{P}\{\lim_n \bar{p}_{ij}^{(n)} \neq p_{ij}\} = 0$. As we have a finite set of parameters

$$\begin{aligned}\mathbb{P}\{\text{An } i,j \text{ exists with } \lim_n \bar{p}_{ij}^{(n)} \neq p_{ij}\} &= \mathbb{P}\big(\bigcup_{i,j} \{\lim_n \bar{p}_{ij}^{(n)} \neq p_{ij}\}\big) \\ &\leq \sum_{i,j} \mathbb{P}\{\lim_n \bar{p}_{ij}^{(n)} \neq p_{ij}\} = 0.\end{aligned}$$

Hence, $\mathbb{P}(\{\lim_n \mathbb{V}(\bar{\mathbf{P}}^{(n)}, R) \neq \mathbb{V}(\mathbf{P}, R)\}) = 0$ and the estimator converges a.s.. □

The ML variance estimator has different desirable properties concerning bias and convergence speed per sample. In particular, it is *nearly unbiased* for the *acyclic* and the *undiscounted* MDPs. Furthermore, the difference between the ML estimator and the optimal unbiased estimator depends essentially on the size of the bias. In the unbiased case both are equivalent and for small bias values they are very similar. With optimal we refer to the expected loss where the loss function needs to be convex. An example for such an expected loss is the *mean squared error (MSE)*.

We restrict the discussion to the acyclic. We first split the estimator into a second moment part and a value part:

**Proposition 3.4.1.** *$\mathbb{V}(\mathbf{P}, R)$ is equal to $M(\mathbf{P}, R) - V(\mathbf{P}, R) \odot V(\mathbf{P}, R)$, where*

$$M(\mathbf{P}, R) = (\mathbf{I} - \gamma^2 \mathbf{P})^{-1} \boldsymbol{\vartheta} \quad \textit{with} \quad \boldsymbol{\vartheta}_i := \sum_j \boldsymbol{p}_{ij} (R_{ij}^2 + 2\gamma R_{ij} \boldsymbol{v}_j).$$

*$M(\mathbf{P}, R)$ is the second moment of the reward sequence and $\odot$ denotes the entry wise multiplication.*

*Proof.* $\mathbb{V}(\mathbf{P}, R) = M(\mathbf{P}, R) - V(\mathbf{P}, R) \odot V(\mathbf{P}, R)$ is the standard variance decomposition, e.g. [7]. Equation (3.1) can be written as $\mathbb{V}(\mathbf{P}, R) = \boldsymbol{\theta} + \gamma^2 \mathbf{P} \mathbb{V}(\mathbf{P}, R)$. Combining the equations

$$
\begin{aligned}
& M(\mathbf{P}, R) - V(\mathbf{P}, R) \odot V(\mathbf{P}, R) = \boldsymbol{\theta} + \gamma^2 \mathbf{P}\big(M(\mathbf{P}, R) - V(\mathbf{P}, R) \odot V(\mathbf{P}, R)\big) \\
& \Rightarrow M(\mathbf{P}, R) = \underbrace{\big(\boldsymbol{\theta} + V(\mathbf{P}, R) \odot V(\mathbf{P}, R) - \gamma^2 \mathbf{P} V(\mathbf{P}, R) \odot V(\mathbf{P}, R)\big)}_{\boldsymbol{\vartheta}} + \gamma^2 \mathbf{P} M(\mathbf{P}, R) \\
& \Rightarrow M(\mathbf{P}, R) = (\mathbf{I} - \gamma^2 \mathbf{P})^{-1} \boldsymbol{\vartheta}.
\end{aligned}
$$

□

The ML variance estimator is biased even for the acyclic MDPs. The reason is analogous to that of the sample mean the normalization, i.e. $1/(n)$ instead of $1/(n-1)$. Yet, for the associated estimator of the second moment we got

**Lemma 19.** *For the acyclic MDP the estimator $M(\bar{\mathbf{P}}, R)$ of the second moment is unbiased and the unbiased estimator with minimal expected loss if the loss function is convex.*

Essentially, one needs to show that the estimator is unbiased. The optimallity statement follows then from the *Rao-Blackwell* theorem [51] as the maximum likelihood parameter estimates are a *complete sufficient statistics* for a MDP. The derivation of unbiasedness is analogous to the case of the ML value estimator.

## 3.5 Experiments

We performed three sets of experiments. In the first set we compared the values of the different "variances". In the second set, we determined the optimal policies with respect to the value, the discount-normalized, and the Taylor approximation trade-off. In the last set we compared the performance of the variance estimators.

### 3.5.1 Tasks

For the evaluation, we used two models that were applied to real-life situations. These models were chosen from the papers from D. J. White [82, 83, 84]. In these papers MDPs that had an impact onto industrial decisions were collected.

#### Grand Coulee Plant

Water reservoirs of a hydroelectric system collect water during the high river flow and use during low flows, aiming to generate adequate amounts of electricity that meet the demand throughout the year. In [52], the amount used and stored at the reservoir for the Grand Coulee steam plant on the Columbia River is modelled to find the optimum water usage. The model consists of: an electricity demand; the predicted amount of water influx (probability density function generated from the 39-year record of river flows); the amount stored before and after the usage; and the action. The action is defined as the amount of water used to generate the electricity (indexed by 1 to 24). The states are defined by

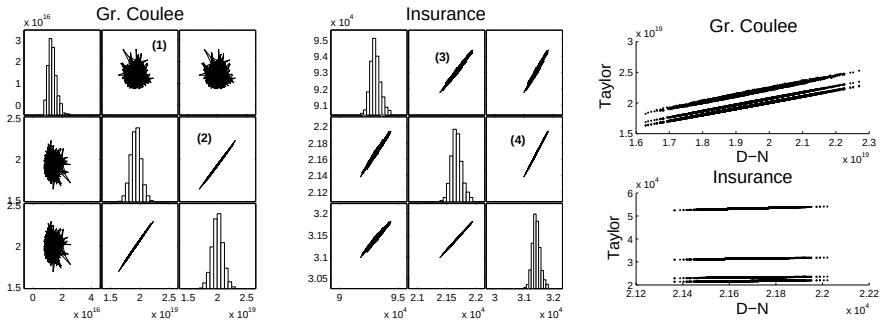


Figure 3.1: The left two plots are scatter plots of the true variance, the Discount Normalized variance and the Taylor approximation (line/row 1: true variance; 2: Discount Normalized; 3 Taylor Approximation). Each point corresponds to a different policy. On the diagonal the marginals are plotted for the different variances. The right two plots compare Discount Normalized with the Taylor Approximation for different expansion points (distance 0 to 90%, see text).

the current amount stored (1 to 40) and the month of the year (1 to 12). The transition probabilities depend on the predicted water influx of the month the amount used at the hydroelectric system. Rewards depend on the state and the action (Equation 5 in [52]).

**Automobile Insurance**

In most automobile insurance systems the insurance premium depends on the number of filed claims per year. In [79] the german automobile insurance system is modelled. We use this model with the slight simplification that we assume stationarity over the years. The state is given by the number of accidents that occured in a year (0 to 4) and the risk category of the insured (1 to 10). The actions are the number of claims filed (0 to 4). The transition probabilities depend deterministically on the old risk category and the number of claims filed (Table 1 in [79]) and probabilistically on the number of occured accidents. The number of occurences is modelled in [79] with a Poisson distribution. We used a mean of 2. The rewards depend solely on the risk category (Equation 1 and Table 1 in [79]).

### 3.5.2 Comparison of the "Variances"

To compare the values of the different "variances", we generated random policies for the two models and calculated the mean-variance costs (10000 runs). Figure 3.5.1 shows the results. One can observe that the Taylor approximation is well correlated with the discount normalized variance (2 and 4). Yet, the discount normalized is only well correlated with the true variance for the insurance data (3) while it is decorrelated for the Grand Coulee data (1). The right side of the Figure shows the effect of the expansion point (growing distance from bottom to top). While the values differs strongly the correlation stays nearly the same (for the insurance data this is hard to observe on the plot).

### 3.5.3 Mean-Variance Tradeoffs

We calculated optimal policies for the data sets based on: value, discount normalized and Taylor Tradeoff. We measured the performance with the value equation, with the true variance and with the discount normalized variance. We were unable to calculate the discount normalized solution for the Grand Coulee data set, as the memory requirement was too high. The optimal value policy is best for the Grand Coulee data with regard to the variance. The reasons are two: (1) the discount normalized variance is rather decorrelated from the true variance and (2) the optimization of the value reduces for this MDP strongly the variance.

| Task | **Grand Coulee** | | |
|---|---|---|---|
| Policy | Discount-Normalized | Taylor | Value |
| Value | NA | $4.55 \cdot 10^9$ | $\mathbf{6.4 \cdot 10^9}$ |
| Variance | NA | $7.64 \cdot 10^{15}$ | $\mathbf{3.99 \cdot 10^{15}}$ |
| D-N Variance | NA | $\mathbf{1.88 \cdot 10^{18}}$ | $2.78 \cdot 10^{19}$ |
| Task | **Insurance** | | |
| Policy | Discount-Normalized | Taylor | Value |
| Value | 615 | 615 | **632** |
| Variance | $\mathbf{9.06 \cdot 10^4}$ | $\mathbf{9.06 \cdot 10^4}$ | $9.52 \cdot 10^4$ |
| D-N Variance | $\mathbf{2.21 \cdot 10^4}$ | $\mathbf{2.21 \cdot 10^4}$ | $2.22 \cdot 10^4$ |

### 3.5.4 Variance Estimation

We compared the ML and MC variance estimators. The following table shows the MSE (10000 runs). First number is for Gr. Coulee, second for Insurance. The errors are scaled by the terms written beside ML and MC. ML is in the order of magnitudes better than MC.

| Runs | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|
| ML ($10^3/10^6$) | 5.45/5.93 | 4.43/3.08 | 2.56/1.98 | 2.46/1.06 | 0.57/0.48 |
| MC ($10^6/10^9$) | 5.8 /8.38 | 4.81 /7.99 | 3.35/7.82 | 1.9/7.75 | 0.53/7.7 |

## 3.6 Summary

We introduced different Risk-Sensitive objectives to RL. In particular, we derived a Taylor approximation for the discount normalized variance. The quality of the approximation is very good in the simulations. The discount normalized variance, however, is not always a good measure for the true variance. Care must therefore be taken in applying the objectives. The simulations suggest that the solutions should be cross-checked with the true variance or with variance estimators.

# Part II

# Sensoric Processing and Control in Feature Space

# CHAPTER 4

# Navigation Using Slow Feature Analysis and Reinforcement Learning


## Abstract

We present an adaptive control system for navigating a robot using camera images. The main feature of the system is the visual processing which generates a Fourier expansion on the area in which the robot acts. In our approach the system for visual processing is not "hard-coded" but learned from examples of the concrete environment using a Slow Feature Analysis. The control is learned by applying the Least-Squares Policy Iteration and the Least-Squares Temporal Difference Value estimator onto the learned visual features. The combination of the linear value estimator and the features corresponding to the elements of the Fourier expansion turns out to be a powerful tool to approximate value functions. The navigation capability of the control system is demonstrated and analyzed on a real robot as well as in simulated environments. In addition, we derive a Kernel Slow Feature Analysis to increase the capability of the preprocessing.


## 4.1 Introduction

A long term goal of robotics is to develop robots that can reliably interact with everyday environments to support humans in all kinds of tasks. One major difficulty for such robotic systems is the immense variability of environments and tasks. Assume, for example, that a company wants to sell a cleaning robot. Every robot will face a different household and owner with particular expectations about what the robot should do. Therefore, it appears to be impossible to predefine the control and it seems more reasonable to aim for a robot that adapts quickly to the household and the task. As the task is only known to the owner, there is a need of interaction with him. The usual owner will not have a technical background and the interaction should therefore stay as simple

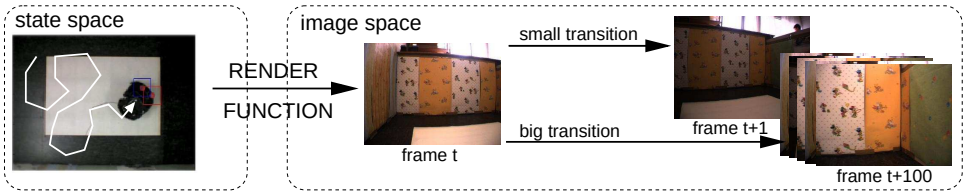


Figure 4.1: **Left Side:** Top view of a quadratic room and a path taken by the robot. **Right Side:** Transformation of the camera images in relation to the delay.

as possible. A natural and simple way for such an interaction is by means of reward and punishment. Adaptation to the concrete task can then be achieved by applying Reinforcement Learning methods.

A major problem with real-world environments is that no state space is available, but only sensory observations of the environment. In recent years a feature space setting has become increasingly popular in Reinforcement Learning, which can be applied to such environments. Different value estimators like *Least Squares Temporal Difference Learning* (LSTD, [19, 18]) and policy iteration methods like *Least Squares Policy Iteration* (LSPI, [50]) have been developed for this setting. The methods, however, rely on a well structured input space to achieve a good performance. In most cases, raw sensory observations are not sufficient and a sophisticated preprocessing is needed. Images from a camera, for example, induce a high-dimensional space and it is intractable to apply methods like LSTD directly.

In the feature space setting the performance of a learned control depends strongly on how well the value function can be approximated. The question now is, can we construct a feature space that facilitates the approximation capabilities? Furthermore, the feature space should be low dimensional such that methods like LSTD and LSPI can be applied and topological properties of the environment like the distance between states should be preserved in the feature space. I.e. if states $s$ and $s'$ have a small distance then $\phi(s)$ and $\phi(s')$, where $\phi$ denotes the mapping into the feature space, should also have a small distance in feature space. The motivation for this property is that the difference in value between states usually depends on the distance between the states, i.e. two states that are close by will have a similar value.

To make the feature space low dimensional is certainly not a problem. More interesting is the value approximation problem. This problem is, however, extensively studied in approximation theory (e.g. [27]) and we can use results from there. In this work we use a "classical" approximation space: the space of *trigonometric functions*. Projecting a function into this space corresponds to a Fourier transform of this function.

Till here it sounds straight forward to construct a "good" feature space, but there is a catch! We want to construct trigonometric functions on our unknown state space. How should that be possible if we have only sensory data like camera images detached from the state space? The key idea is now that images from the robot camera are *temporally* close by if they *correspond to similar coordinates*, i.e. two images that are taken with a delay of 1 second will correspond to similar coordinates while images that are taken with a delay of 100 seconds are likely

to be far apart (Figure 4.1). Therefore, a function that has small distances in feature space for temporally close feature vectors will automatically have small distances with respect to the coordinates in the environment. We can put it differently: the topology of the environment corresponds to the topology that is induced by the temporal structure in the observed sensory data.

This similarity between the topologies allows us to define optimization problems that generate a feature space with a distance measure that is related to the distances in the environment, i.e. $\boldsymbol{\phi}(s) \sim \boldsymbol{\phi}(s') \Leftrightarrow s \sim s'$, where $\boldsymbol{\phi}(s)$ corresponds to the representation in feature space of state $s$ and $\sim$ refers to suitable metrics in feature space, respectively in the environment. One way to define this as an optimization problem is to enforce small distances between $\boldsymbol{\phi}(s)$ and $\boldsymbol{\phi}(s')$ for states $s$ and $s'$ that are close to each other. Whereas we need to replace the minimization with respect to the distance between $s$ and $s'$ in the environment with the "temporal distance" between the sensory signals corresponding to $s$ and $s'$. The distance in feature space can be measured with different metrics, like the ones that are induced by the $L^1$- or $L^2$-norm. Beside this distance measure it is needed to define constraints to avoid trivial solutions like $\boldsymbol{\phi}(s) = 0$ for all $s$. Both the metric and the constraints can be used to define the form of the features. We are interested in generating trigonometric functions, therefore we need to find the right metric and the constraints which enforce this form under reasonable assumptions on the initial sampling process.

A method that fulfills all our demands is a method called *Slow Feature Analysis* (SFA, [88]), where slow refers to small changes between successive feature vectors that are enforced by the SFA objective. Under suitable conditions the SFA optimization problem generates trigonometric functions [87, 34]. Whereas exact trigonometric functions can only be achieved in theory. In practice these optimal solutions cannot be achieved because the representation of the data (images from the video) and the class of functions used (e.g. linear, polynomial) restricts the solution space. A crucial element is the model complexity and the number of samples that are used for optimization. The original linear SFA from [88] is to weak to deal with video data. Therefore, in [34] a hierarchical SFA was developed which can be applied to video data. Yet, the hierarchy introduces a large number of parameters and the optimization is cumbersome. Here, we make a kernel based approach and we extend the linear SFA to a kernel SFA to increase the capability of the method. The complexity of the kernel SFA can directly be adjusted by the number of support vectors that are used. With growing model complexity the features generated by the kernel SFA get increasingly similar to the "optimal features", i.e. the trigonometric functions.

We use the SFA to preprocess the camera images and to generate a "good" feature space for Reinforcement Learning methods. We use in this work the LSPI policy iteration in combination with the LSTD value estimator to learn a control. Here, we use LSTD to estimate Q-values, i.e. values for state-action combinations. This is needed to circumvent the problem of "building" a model for transitions between states. Such a model would be needed if we would estimate values for states.

We tested our system in simple navigation tasks. The goal in these navigation tasks was always to reach a specific spot inside a room. To evaluate the performance, we conducted robotic experiments in a small rectangular room and virtual experiments in two connected rooms. The control system was learned without prior knowledge about the environment, i.e. no specialized feature de-

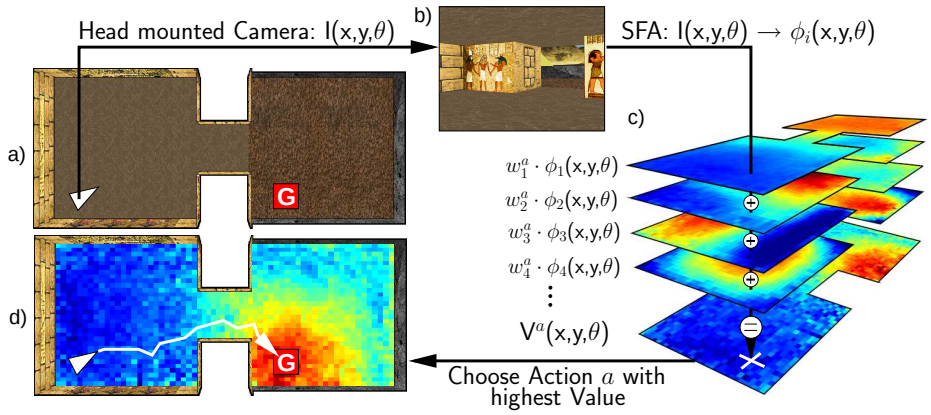


Figure 4.2: The figure sketches the control process for a virtual two room environment following the arrow from (a) to (d). (a) shows the environment (top view). The goal is to reach the position **G**. At the agents position (white arrow) a camera image is available which is shown in (b). The image is processed with the SFA and results in a feature vector $(\boldsymbol{\phi}_1(x,y,\theta), \boldsymbol{\phi}_2(x,y,\theta), \ldots)^\top$ representing the position. The current feature vector is marked by the vertical line connecting the layers (c). The features are weighted by a value approximation method and summed up to provide the value estimate at that location (bottom image in (c)). The value estimate is used to navigate the agent (d).

tectors for graphic processing were used. The robot gained a reward if it reached the goal and a punishment if it crashed into a wall.

The complete control process from the camera image to the control signal is shown in Figure 4.2. The camera image (b) is preprocessed with SFA. As a result, we get a feature vector $\boldsymbol{\phi}$, where each component $\boldsymbol{\phi}_i$ corresponds to a multivariate trigonometric function. The camera image at location $(x, y)$ and orientation $\theta$ is mapped to a vector $\boldsymbol{\phi}(x, y, \theta)$ (vertical line in (c)). Applying LSTD onto these features results in a weighted sum of trigonometric functions, i.e., an approximation of the value function (lowest layer in c). We use this approximation to control the robot (d).

In the "real-world" robotic experiments the system succeeded (reached the goal) in 17 of 20 trials if the goal was in the middle of the room and in 15 of 20 trials if the goal was in the lower right corner (Figure 4.7, p. 76).

### 4.1.1 Alternative Approaches

Camera based navigation using Reinforcement Learning has to our best knowledge not been done before. Visual processing is, however, an active field and approaches exist that are similar to our visual processing. The approaches that are most similar address the *simultaneous localization and mapping (SLAM)* problem. The goal of SLAM is the estimation of positions of landmarks and of the camera position (for *visual SLAM*). SLAM was originally used with sonar sensors [71]. Roughly a decade ago, the methods were adapted to visual sensors [24] and are currently under development by several groups [25, 68, 28].

Many of the SLAM methods rely on *extended Kalman filters (EKF)* for po-

sition estimation [24, 28]. The EKF tracks dependencies between landmarks over different camera positions and therefore has a complexity of $O(n^2)$, where $n$ is the number of landmarks. The complexity makes the approach not usable for larger environments. One approach that is used to tackle this problem are *particle filters* [60, 58]. In this approach each particle represents a camera position with the corresponding landmarks. Landmark dependencies over different camera positions are ignored. The particle filter methods have a complexity of $O(mn)$, where $m$ is the number of particles. Particle filters have been applied to visual SLAM by [68] and [28]. An alternative approach to particle filters has been made in [32]. There, the EKF approach has been extended by producing independent local maps of limited size. These local maps are stitched together in a global layer and the algorithm is called *hierarchical maps*. Recently, in [23] this approach has been applied to visual SLAM.

Another speedup can be achieved by using so called *active measurement* strategies. In the common approach a set of landmarks is available which must be compared to the camera image. I.e. each landmark is a small 2-D patch and these patches are convolved with the whole image. If a convolution at a point $(x, y)$ in the image yields a high value, then the landmark is detected. Testing each landmark at each position of the image is computationally expensive [68]. Active measurement strategies overcome this problem by incorporating information about the expected location of a landmark. This way, visual SLAM can be performed in real time [25, 23, 28].

**Differences** The SLAM approach differs from the SFA approach in different aspects. The SLAM approaches are less adaptive then the SFA approach. Usually, the SLAM approaches "adapt" to the environment by building a database of landmarks. In visual SLAM a fixed set of landmark detectors like edge or corner detectors are usually used for this task. A problem with these detectors are all sorts of transformations, like rotations or optical distortions (e.g. by a lens). Approaches to overcome this problem are (1) the application of 2-D transformations onto the set of landmarks [28], i.e. rotate a corner and perform a convolution with the rotated corner and (2) the use of scale and rotation invariant features (SIFT, [68]). The SFA approach is different. A set of images from the environment is recorded and the SFA constructs automatically a visual processing that is best suited to detect differences in this concrete environment. For example, if a lot of optical distortions are present in the images, then the SFA will try to generate a visual processing that is unaffected by these transformations. So the SFA based system is very specialized to the concrete environment.

Another difference is that SLAM tracks the current state of the system and updates this with incoming observations. The SFA based system is instantaneous and has no internal estimate of the current state.

## 4.2 Control: Reinforcement Learning

Our control setting differs from the classical Reinforcement Learning setting as we observe only camera images and because we have a *continuous state space* with *discrete actions* at *discrete time steps*. The continuous state space is not problematic since we use an approximation of the value function [15]. Yet, the

formal definitions are cumbersome and we fall back to the discrete state space case for explaining the concepts. The interested reader may, however, consult [14] for formal details.

We start with the definition of a *Markov Decision Process* (MDP). A MDP consists of a state space $\boldsymbol{S}$, an action space $\boldsymbol{A}$, transition probabilities $\boldsymbol{p}_{ij}^{a}$ to move from state $i$ to state $j$ with action $a$ and a reward $\boldsymbol{r}_i$ associated with state $i$. The control is introduced through policies $\pi$. In this work we use *deterministic* and *randomized* policies. The goal in Reinforcement Learning is the maximization of the *value* with respect to the policy. The value for a state $i$ given a policy $\pi$ is $V^{\pi}(i) := \mathbb{E}_{\pi}[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = i]$, where $\mathbb{E}_{\pi}$ denotes the expectation given policy $\pi$, $\gamma \in (0, 1]$ is the discount and $r_t$ the reward received at time $t$.

We use a policy-iteration (PI) based approach to find a policy that maximizes the value [76, 15]. The PI algorithm alternates between a policy evaluation and a policy improvement step. A policy $\pi_u$ at iteration $u$ is evaluated with a value estimator and the estimate is used to calculate a new policy $\pi_{u+1}$. In our setting we do not observe the states directly. Instead we have observations from a high dimensional space that represent the states. Instead of a value for each state $i$, i.e. $V(i)$, we got now an approximation of the value function, i.e. a function $\mathcal{V}$ that maps an observation $o \in \mathcal{O}$ representing state $i$ onto an approximate value ($\mathcal{V}(o) \approx V(i)$). Instead of the exact PI we must use in this setting an *Approximate Policy Iteration* (API, [15]). An API must not converge to the optimal policy, respectively it can happen that it does not converge at all.

In the following we shortly describe two problems that we encountered in the practical application of current API approaches and that are rarely documented: (1) If a value estimator like LSTD is used then a *model for the transition probabilities between states* is needed to apply the API. In [31] this problem is circumvented by providing the system with the true transition model and in [64] an explicit transition model is learned. Learning a transition model for a complex real-world problem is, however, a challenging task. An easier solution is provided by switching from value estimation to Q-value (values for state-action pairs) estimation. This approach is documented in [50] and the method is called LSPI. We followed this approach.

(2) The sampling, i.e. the movement of the robot is important. If the sampling of this training data is unbalanced, meaning a lot of transitions are observed at few locations of the environment or if only a subset of the available actions are observed then the estimates will be bad for the other locations, respectively actions. The sampling is particularly problematic for an API if actions are chosen deterministically (e.g. greedy). For such a sampling the Q-values are badly estimated for most actions with fatal effects for the API. In [29] such an approach is taken and is named SARSA. In [50] the problem is circumvented by using samples $(s, a) \rightarrow (s', a')$, where only $a'$ is chosen greedily while $a$ is chosen randomly. We essentially followed this approach, with the minor modification of including a sample for every possible action $a' \in \mathbf{A}$ (in our case we had 3 actions). This small modification improved the convergence properties considerably. Yet, it is just a heuristical modification without a theoretical argument supporting it.

Popular value/Q-value estimators are linear, i.e. $\bar{V}(\boldsymbol{\phi}) = \boldsymbol{\phi}^{\top}\boldsymbol{w}$ [19, 29], where $\boldsymbol{\phi}$ is a feature vector (e.g. an preprocessed image). In contrast to supervised learning the value is not observed directly. Therefore, the methods

minimize the *Bellman error*. We use in this work the LSTD value estimator. The corresponding optimization problem and the solution is given by

$$\min_{\boldsymbol{w}} ||\boldsymbol{\Phi}(-\dot{\boldsymbol{\Phi}}_\gamma^\top \boldsymbol{w} - \boldsymbol{r})|| \Rightarrow \quad \boldsymbol{w} = -(\dot{\boldsymbol{\Phi}}_\gamma \mathbf{K} \dot{\boldsymbol{\Phi}}_\gamma^\top)^{-1} \dot{\boldsymbol{\Phi}}_\gamma \mathbf{K} \boldsymbol{r},$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{k \times n}$ denotes the data matrix, $\mathbf{K} := \boldsymbol{\Phi}^\top \boldsymbol{\Phi}$, $k$ is the size of the feature space and $n$ is the number of examples. Furthermore, $\dot{\boldsymbol{\Phi}}_\gamma \in \mathbb{R}^{k \times n}$ denotes the discrete derivative of the observations scaled by the discount, e.g. the $t$'th column is given by $\gamma \boldsymbol{\phi}_{t+1} - \boldsymbol{\phi}_t$.

## 4.3 Slow Feature Analysis

The goal of the preprocessing are the generation of trigonometric functions on the environment in which the robot acts. SFA is the method which produces these functions. We start with defining the SFA optimization problem in a general form. In the following section we describe the linear SFA from [88] and we extend it to a kernel SFA. After these sections we describe the "optimal SFA features" [87, 34]. These features are generated if our model space is not restricted and if different assumptions concerning the environment and the sampling are fulfilled. Following that we discuss the approximation properties of the SFA features and we link them to API convergence statements.

### 4.3.1 Optimization Problem

Given a sequence of observations $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)} \in \mathcal{O}$, we want to find a set of $p$ continuous and differentiable mappings $\boldsymbol{\phi}_i(\boldsymbol{x}) : \mathcal{O} \to \mathbb{R}$ where $1 \leq i \leq p$ that satisfy [88]:

$$\begin{aligned} \min \quad s(\boldsymbol{\phi}_i) &:= \mathbb{E}[\dot{\boldsymbol{\phi}}_i^2] \\ \text{s.t. } \mathbb{E}[\boldsymbol{\phi}_i] &= 0 \quad \text{(Zero Mean)} \\ \mathbb{E}[\boldsymbol{\phi}_i^2] &= 1 \quad \text{(Unit Variance)} \\ \forall j \neq i : \mathbb{E}[\boldsymbol{\phi}_i \boldsymbol{\phi}_j] &= 0 \quad \text{(Decorrelation)}, \end{aligned} \tag{4.1}$$

where $\mathbb{E}$ denotes the expectation, $\dot{\boldsymbol{\phi}}$ the temporal derivative and $s(\boldsymbol{\phi}_i)$ the *slowness*. The slowness of a *feature* $\boldsymbol{\phi}_i$ is measured with the 2-norm. The indices $i$ are sorted in ascending order according to their slowness. Trivial solutions like the constant $\boldsymbol{\phi}_i(\boldsymbol{x}) = 0$ are avoided by the unit variance constraint. The decorrelation constraint guarantees that more than one function is generated, i.e. without it all $\boldsymbol{\phi}_i$ would be equivalent.

### 4.3.2 Linear SFA

We aim to find a set of $k$ functions that solve minimization problem 4.1. The expectation is replaced with the empirical average $\hat{\mathbb{E}}$ with respect to $n$ observations $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)} \in \mathcal{O} = \mathbb{R}^d$. In this section we restrict the solution space in which we search for a minimizing element to the space of affine functions, i.e. functions $\boldsymbol{\phi}_i(\boldsymbol{x}^{(t)}) = \boldsymbol{w}_i^\top \boldsymbol{x}^{(t)} - c_i$, where $\boldsymbol{w}_i$ is a weight and $c_i$ a constant. To keep the notation simple, we group observations and features into the matrices

$\mathbf{X}_{it} = x_i^{(t)}$ and $\mathbf{\Phi}_{it} = \boldsymbol{\phi}_i(\boldsymbol{x}^{(t)})$. Instead of the temporal derivative the *discrete temporal derivative* $\dot{\mathbf{X}} = [\boldsymbol{x}^{(2)} - \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)} - \boldsymbol{x}^{(n-1)}]$ is used.

The $k$ solutions are calculated simultaneously in 3 steps [88]:

1. *Centering of the data*: $\boldsymbol{x}_c := \boldsymbol{x} - \boldsymbol{x}_m$, where $\boldsymbol{x}_m = \hat{\mathbb{E}}[\boldsymbol{x}]$.

2. *Sphering of the centered data*: $\boldsymbol{x}_s := \mathbf{S}\boldsymbol{x}_c$, where $\hat{\mathbb{E}}[\boldsymbol{x}_s \boldsymbol{x}_s^\top] = \mathbf{I}$. Sphering establishes *unit variance* and *decorrelation*. An eigenvalue decomposition of the covariance matrix $\hat{\mathbb{E}}[\boldsymbol{x}_c \boldsymbol{x}_c^\top] = \frac{1}{n}\mathbf{X}\mathbf{X}^\top - \boldsymbol{x}_m \boldsymbol{x}_m^\top = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, can be used to determine the *sphering matrix* $\mathbf{S} = \mathbf{\Lambda}^{-1/2}\mathbf{U}^\top$.

3. *Minimizing the slowness*: This can only be achieved by a rotation of the sphered data, since any other operation would violate the already fulfilled constraints. Again, an eigenvalue decomposition of the covariance matrix $\hat{\mathbb{E}}[\dot{\boldsymbol{x}}_s \dot{\boldsymbol{x}}_s^\top] = \frac{1}{n-1}\mathbf{S}\dot{\mathbf{X}}\dot{\mathbf{X}}^\top\mathbf{S}^\top = \tilde{\mathbf{U}}\tilde{\mathbf{\Lambda}}\tilde{\mathbf{U}}^\top$ can be used. The solutions are given by the $p$ smallest eigenvectors $\tilde{\mathbf{U}}_p$.

Combining the steps the affine function $\boldsymbol{\phi}$ that solves the optimization is given by $\boldsymbol{\phi}(\boldsymbol{x}) = \mathbf{W}^\top \boldsymbol{x}$, where $\mathbf{W} = \mathbf{U}\mathbf{\Lambda}^{-1/2}\tilde{\mathbf{U}}_p^\top$ and $\boldsymbol{c} = \mathbf{W}^\top \boldsymbol{x}_m$

The algorithm has to collect the mean $\frac{1}{n}\mathbf{X}\mathbf{1}$, the covariances $\frac{1}{n}\mathbf{X}\mathbf{X}^\top$, $\frac{1}{n-1}\dot{\mathbf{X}}\dot{\mathbf{X}}^\top$ and has to perform two eigenvalue decompositions in step 2 and 3. Since the eigenvalue decompositions have complexity $O(d^3)$, the algorithm has an overall complexity of $\max(O(d^2 n), O(d^3))$.

### 4.3.3 Dual Approach - Kernel SFA

Linear methods are usually not powerful enough for real-world applications. The usual approach to increase their power is an embedding of the data items into a high-dimensional *feature space* and an application of the learning algorithm in this space. This embedding is usually done implicitly with the help of a *kernel function*. Learning methods can make use of this approach if they do not need the coordinates of the data items but only their pair wise inner product.

SFA can be formulated in terms of inner products and the approach can therefore be applied. For SFA the approach is particularly simple due to the close relation between kernel and covariance matrix. Furthermore, due to this relation the linear SFA algorithm needs to be modified only minorly.

#### Background

In this section we give a rough summary of the needed kernel terminology. For a detailed introduction see for example [66]. First, let $\boldsymbol{\psi} : \mathbb{R}^d \to \mathbb{R}^k$ be an embedding of the data items into an usually high dimensional *feature space*. If an algorithm is formulated in terms of inner products $< \cdot, \cdot >$, then $< \boldsymbol{x}, \boldsymbol{x'} >$ can be replaced by $< \boldsymbol{\psi}(\boldsymbol{x}), \boldsymbol{\psi}(\boldsymbol{x'}) >$ and the algorithm can be run in feature space without modification.

Usually $k >> d$ and algorithms cannot be applied efficiently in feature space. An alternative is provided by kernels. Instead of an explicit construction of the feature space kernels imply a feature space, i.e. given a symmetric and positive definite kernel $k(\boldsymbol{x}, \boldsymbol{x'})$ a Hilbert space $(\mathcal{H}, < \cdot, \cdot >_\mathcal{H})$ and a mapping $\boldsymbol{\psi}$ exists such that $k(\boldsymbol{x}, \boldsymbol{x'}) =< \boldsymbol{\psi}(\boldsymbol{x}), \boldsymbol{\psi}(\boldsymbol{x'}) >_\mathcal{H}$. An inner product based algorithm can

now be applied by replacing $< \boldsymbol{x}, \boldsymbol{x'} >$ with $k(\boldsymbol{x}, \boldsymbol{x'})$. Popular kernels are the *polynomial* kernel of degree $g$ $k(\boldsymbol{x}, \boldsymbol{x'}) = (\boldsymbol{x}^\top \boldsymbol{x'}+1)^g$ and the *RBF* (or *Gaussian*) kernel $k(\boldsymbol{x}, \boldsymbol{x'}) = exp(-\frac{||\boldsymbol{x}-\boldsymbol{x'}||_2^2}{2\sigma^2})$, where $\sigma$ denotes the standard deviation.

### Covariance Matrix and Kernel Matrix

The SFA algorithm performs an eigenvalue decomposition of the covariance matrix. In feature space this corresponds to $n\mathbb{E}[\boldsymbol{\psi}\boldsymbol{\psi}^\top] = \boldsymbol{\Psi}\boldsymbol{\Psi}^\top = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$, where $\boldsymbol{\Psi}_{it} = \boldsymbol{\psi}_i(\boldsymbol{x}^{(t)})$ denotes the matrix of the embedded samples. The decomposition has a complexity of $O(k^3)$ and is therefore infeasible in a high dimensional space.

Luckily, the nonsingular eigenvalues of the *kernel matrix* $\mathbf{K} = \boldsymbol{\Psi}^\top\boldsymbol{\Psi} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top$ are the same as the eigenvalues of the covariance matrix and the eigenvectors are related by $\mathbf{U} = \boldsymbol{\Psi}\mathbf{V}\boldsymbol{\Lambda}^{-1/2}$ [66]. If we express the eigenvalue decomposition of the covariance matrix in terms of the kernel matrix, the complexity decreases to $O(n^3)$, where usually $n << k$.

### Kernel SFA Algorithm

In this section we derive the kernel SFA. For a related cost function a kernel approach has been made in [20]. Beside the difference in the cost function their approach differs from ours because they do not use a kernel matrix but work directly with support vectors. Using a specific kernel matrix approximation, their approach turns out to be a special case of ours.

In the following let $k(\boldsymbol{x}, \boldsymbol{x'})$ be a given kernel and $\boldsymbol{\psi}$ the corresponding feature mapping. Similar to linear SFA the kernel SFA algorithm consists of three steps:

1. *Centering* can be performed directly on the kernel matrix [66]: $\mathbf{K}_c = (\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)\mathbf{K}(\mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top)$. In the following we make use of the centered matrix $\mathbf{K}_c$ and with it $\boldsymbol{\Psi}_c$.

2. *Sphering*: As discussed in 4.3.3, we substitute the nonsingular eigenvectors $\mathbf{U}$ in the sphering matrix with those of the kernel matrix: $\mathbf{S} = \frac{1}{\sqrt{n}}\boldsymbol{\Lambda}^{-1}\mathbf{V}^\top\boldsymbol{\Psi}_c^\top$.

3. *Minimizing the slowness*: Let $\dot{\boldsymbol{\Psi}}_c := \boldsymbol{\Psi}_c\mathbf{D}$, with $\mathbf{D} \in \mathbb{R}^{n\times n-1}$ being a matrix which is everywhere zero except for $\mathbf{D}_{i,i} = -1$ and $\mathbf{D}_{i+1,i} = 1$. This way we can express the sphered covariance matrix $\frac{1}{n-1}\mathbf{S}\dot{\boldsymbol{\Psi}}_c\dot{\boldsymbol{\Psi}}_c^\top\mathbf{S}^\top$ in feature space as $\frac{1}{n(n-1)}\boldsymbol{\Lambda}^{-1}\mathbf{V}^\top\dot{\mathbf{K}}_c\dot{\mathbf{K}}_c^\top\mathbf{V}\boldsymbol{\Lambda}^{-1}$ with $\dot{\mathbf{K}}_c = \boldsymbol{\Psi}_c^\top\boldsymbol{\Psi}_c\mathbf{D}$. The eigenvectors corresponding to the smallest $p$ eigenvalues of this covariance matrix are the rotations $\tilde{\mathbf{U}}_p$ which optimize the problem.

The kernel solution to optimization problem 4.1 is given by $\boldsymbol{\phi}(\boldsymbol{x}) = \mathbf{W}^\top\boldsymbol{k}(\boldsymbol{x})-\boldsymbol{c}$, $\mathbf{W} = (\mathbf{I}-\frac{1}{n}\mathbf{1}\mathbf{1}^\top)\mathbf{V}\boldsymbol{\Lambda}^{-1}\tilde{\mathbf{U}}_p^\top$ is the weight matrix, $\boldsymbol{k}(\boldsymbol{x}) := [k(\boldsymbol{x}^{(1)}, \boldsymbol{x}), \ldots, k(\boldsymbol{x}^{(n)}, \boldsymbol{x})]^\top$ and $\boldsymbol{c} = \frac{1}{n}\mathbf{W}^\top\mathbf{K}\mathbf{1}$ is the weighted mean.

The algorithm collects the matrix $\mathbf{K}$ and derives $\dot{\mathbf{K}}_c\dot{\mathbf{K}}_c^\top$ from it. The two eigenvalue decompositions in steps 2 and 3 are responsible for an overall complexity of $O(n^3)$.

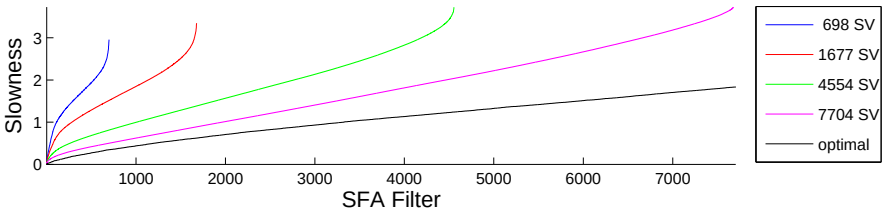


Figure 4.3: The figure shows the performance of the kernel SFA in dependence of the model complexity (number of support vectors). The measure is the slowness of the produced features. The lower the slowness the better the method. The lowest line is the optimal solution, i.e. the slowest possible features. One can observe that with growing model complexity more "reasonable" SFA features are produced and the earlier features become very similar to the optimal solutions.

### Kernel Matrix Approximation

Working with kernel matrices is restricted to a small amount of samples (5000-10000). Different approaches have been made to overcome this problem by means of an approximation $\tilde{\mathbf{K}} \in \mathbb{R}^{m \times m}$ of the kernel matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$, where $m << n$ (see for example [65]).

The simplest approach is called *subset of data*. In this approach only a subset of $m$ samples (*support vectors*) is used and the approximation has size $m \times m$. Because, only the support vectors influence $\tilde{\mathbf{K}}$ all information of the remaining $n - m$ samples is lost.

A better approach is called *projected process*. Here, the non support vector rows are removed but all columns are kept and the resulting matrix $\hat{\mathbf{K}}$ has size $\mathbb{R}^{m \times n}$. The matrix $\hat{\mathbf{K}}\hat{\mathbf{K}}^\top \in \mathbb{R}^{m \times m}$ is then used to approximate $\mathbf{K}^2$. While preserving much more information, the method can only be applied to algorithms which use $\mathbf{K}^2$.

If one aimes to preserve as much information as possible, the choice of suppot vectors is crucial. Finding the optimal set of support vectors is, however, a NP hard problem. Therefore a number of heuristics have been proposed. We use a heuristic which was proposed in [21] and applied to Gaussian Process Temporal Difference Learning [30]. This heuristic runs sequentially through the sample and includes data points into the set of support vectors if the orthogonal residual between the data point and the span of the current set of support vectors $\{s_1, \ldots, s_n\}$ is above a threshold $\beta$, i.e. if

$$\min_{\alpha_1,\ldots,\alpha_n} |\boldsymbol{\psi}(\boldsymbol{x}) - \sum_i \alpha_i < \boldsymbol{\psi}(\boldsymbol{x}), \boldsymbol{\psi}(\boldsymbol{s}_i) > \boldsymbol{\psi}(\boldsymbol{s}_i)| > \beta.$$

Intuition being that new data points contribute essentially the residual, i.e. if the residual is zero then nothing is gained.

The number of support vectors $m$ is largely dependent on the data and the kernel. This can lead to problems, since the complexity of the algorithm scales with $O(m^2 n)$.

One would like to apply the *projected process* method to the kernel SFA, but this method only provides approximations of $\mathbf{K}^2$. Luckily, $\mathbf{K}^2 = \mathbf{V}\boldsymbol{\Lambda}^2\mathbf{V}^\top$

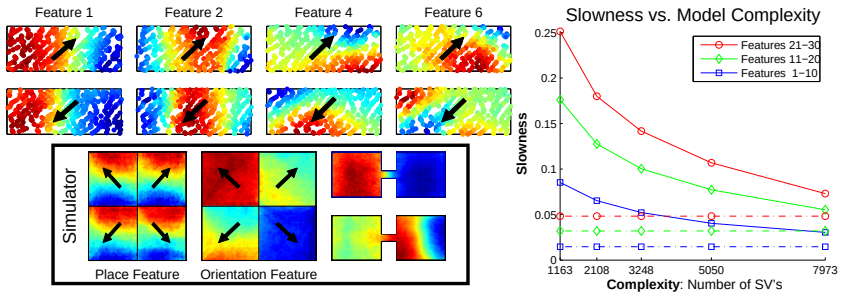


Figure 4.4: The upper rows show four learned SFA features, evaluated on a test set. The arrows indicate the orientation of the robot. On the right, a plot of the achieved slowness against different model classes (as number of used support vectors) is shown. The continuous lines are means over 10 sfa features each and are getting closer to the optimal features (dashed lines) the more support vectors are used. The bounded box shows results from the simulator. Here, we get place and orientation sensitive features by varying the ratio between movements and rotations in the training videos. Further, for settings with multiple rooms, we get features that encode the room or specialize in a specific room. All experiments were performed with kernel SFA and RBF kernels.

has the same eigenvectors and the squared eigenvalues of $\mathbf{K}$. If we perform the eigenvalue decomposition on $\hat{\mathbf{K}}\hat{\mathbf{K}}^\top$ and take the square root of the eigenvalues, the solution is the projected process approximation of $\mathbf{K}$.

With this approximation, the algorithm has complexity $O(m^2 n)$ because it requires the collection of the matrices $\hat{\mathbf{K}}\hat{\mathbf{K}}^\top$ and $\dot{\hat{\mathbf{K}}}\dot{\hat{\mathbf{K}}}^\top$. Additionally, one has to pick a set of support vectors, which usually is costly, too.

We tested the performance of the kernel SFA in an experiment (Figure 4.3 and right side of Figure 4.4). We compared kernel SFA with different numbers of support vectors to the optimal solutions (see next Section). With growing model complexity the kernel SFA approaches the optimal solutions.

### 4.3.4 Theoretical Solutions

One can address optimization problem 4.1 with *(classical) variational calculus* to find solutions in the class of continuous functions. The linear SFA and the kernel SFA produce continuous functions. With growing model complexity (more support vectors) the set of functions that can be represented grows and the solution of the kernel SFA approaches the variational solution. The variational solutions tell us therefore how the "limit solution with respect to model complexity" looks like.

The application of variational calculus, however, is not straight forward. As with optimization problems in finite dimensional spaces only in few cases the analytical solutions can be calculated explicitly. The difficulties for optimization problem 4.1 are introduced through boundary conditions, which depend on the structure of the room in which the robot navigates.

In [34] the variational approach has been taken for a rectangular room. For this case the solutions to the optimization problem are trigonometric functions.

More precisely, let $K := [0,1] \times [0,1] \times [0, 2\pi)$ denote the room[1] and the viewing angle and let $(x, y, \theta) \in K$ be a point in the room. If the movements in the different directions $(x, y, \theta)$ are decorrelated then the (variational) solutions of problem 4.1 are

$$\boldsymbol{\phi}_{ijl}(\iota(x,y,\theta)) = \begin{cases} \sqrt[3]{2}\cos(i\pi x)\cos(j\pi y)\sin(\frac{l+1}{2}\theta), & l \text{ odd} \\ \sqrt[3]{2}\cos(i\pi x)\cos(j\pi y)\cos(\frac{l}{2}\theta), & l \text{ even,} \end{cases}$$

where $i, j, l \in \mathbb{N}$ and $\iota(x, y, \theta) \to \mathcal{O}$ represents the function that maps coordinates onto camera images. We refer to these solutions as the *optimal features*.

If features have the same slowness, a linear mixture is also possible. For example, if $\boldsymbol{\phi}_{100}$ and $\boldsymbol{\phi}_{010}$ have the same slowness, then $s(\boldsymbol{\phi}_{100}) = s(\boldsymbol{\phi}_{010}) = s(a\boldsymbol{\phi}_{100} + b\boldsymbol{\phi}_{010})$ holds as long as $a^2 + b^2 = 1$. This corresponds to an arbitrary rotation between features of equal slowness.

The slowness of the optimal feature is influenced by the frequency and by the driving speed in the three directions of the room $(x, y, \theta)$. If, for example, the rotation speed is high compared to driving speed then features are generated that are invariant to rotation. These features are also called place cells. Similarly, if the rotation speed is low then features are generated that are invariant to the place (orientation features). Place and orientation features are shown in Figure 4.4. Under realistic conditions separated place and orientation features appear seldom. Usually, mixtures of them are generated. The upper line of Figure 4.4 shows different features, whereas we controlled the robot tightly to produce orientation invariant features. The first two features are orientation invariant, but, despite our effort, mixtures appear already at feature 4.

We did the same experiment using the simulator (for details about the simulator see Section 4.4.3). There we were able to produce very clean place and orientation features (Simulator box, Figure 4.4).

A derivation of the variational solutions for general room structures seems to be impossible. In experiments with other room structures we found features that are often very similar to these trigonometric functions. However, beside these features other solutions exist. For example, in the case of two connected rooms features that identify rooms and features that are exclusive to one of the rooms were generated (Simulator box, Figure 4.4).

### 4.3.5 Function Approximation and Policy Iteration

In general one is interested in having guaranties for the performance of a learned control. For example, when we apply a policy iteration in combination with a value estimator we would like to get a policy that is in some way close to the optimal control. According to [15] and [59] it is actually possible to bound the distance between the achieved performance and the performance of the optimal policy (*regret*). These bounds rely on the approximation quality (*bias*) of the approximate value function and on the estimation quality (*variance*). The bound of [15] is based on the $||\cdot||_\infty$-norm while the bound from [59] is based on $||\cdot||_p$-norm. In general, the measure of the estimation error is the mean-squared error (MSE). Therefore, we get a good estimate in the $||\cdot||_2$-norm and not in the $||\cdot||_\infty$-norm from estimators like LSTD. For these the $||\cdot||_p$-norm

[1] quadratic for simplicity

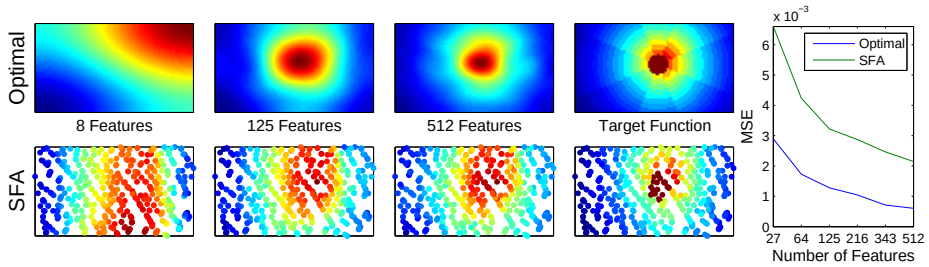


Figure 4.5: The plot shows the value approximation quality in dependence of the number of features. The **upper line** shows approximations of a value function for optimal SFA features and the **lower line** for learned SFA features. The learned SFA features are evaluated on video frames. The value functions (**target functions**) are shown on the right. The shape of the value approximations in the upper and lower line differ, because the learned SFA features got more frequencies in $x$-direction and fewer in $y$-direction then the optimal features. The approximation error (MSE) in dependence of the number of features is plotted on the **right side** of the figure (green line: learned features, blue line: optimal features). For the optimal features one can clearly see that the shape of the approximation becomes more similar to the target function with growing feature size. For the learned features this is more difficult to observe due to the scatter plot, however in the MSE plot a clear improvement can be observed.

bounds are better suited. We will not discuss the bounds any further, but we will concentrate on the approximation (bias) part on which these bounds rely.

The first interesting question is: Given a growing number of optimal features, can a large set of value functions be approximated arbitrarily well? In the appendix we show that we are able to approximate continuous functions arbitrarily well in $||\cdot||_\infty$. Furthermore, we can approximate bounded functions arbitrarily well in the $||\cdot||_p$-norms. Our value functions are not continuous since we choose actions at discrete time steps. However, they are bounded.

Beside statements about the asymptotic case it is possible to derive statements for a finite number of features. In particular, it is possible to bound the approximation error for function classes like the continuous functions or the bounded functions using so called *Jackson Theorems* [45, 27] from approximation theory.

In Figure 4.5 the approximation behavior in dependence to the expansion size is shown for optimal and kernel SFA features. The figure shows the least squares fit to a value function and the corresponding approximation error (MSE). The value function is generated "by hand". The reward is in the middle of the room. The needed number of steps is the sum of the number of needed rotations to turn the "agent" to the middle of the room and the number of steps to move the agent to the goal after rotating it. This heuristic is very similar to the optimal policy. One can observe that the approximation error drops continuously with a growing number of features, despite the fact that the value function is not continuous.

a) Robot b) Robot View c) Ceiling View

Figure 4.6: The Real-World setting. a)The robot. b) An image from the robot camera. c) The rectangular room from above (ceiling camera). Two balls on top of the robot are detected (blue and red rectangle). These are used to log the experiments (robot coordinates and orientation).

## 4.4 Empirical Analysis

We tested the method on a robot in a rectangular room (Section 4.4.1) and additionally on a simulator in two other room structures (Section 4.4.3).

The only sensory input for the control are the images from the video camera, mounted on top of the robot, respectively, the virtual images. Part b) of Figure 4.6 shows an example image. The fact, that no prior knowledge about the environment is given, makes the navigation task challenging.

### 4.4.1 Setup

Since experiments with the robot are time consuming, we aimed to reduce the necessary training data to one random walk video. The frames of this video were scaled down to $32 \times 16$ RGB images. The SFA was trained on a subset of approximately 35.000 of these. Out of this training set, approximately 8000 support vectors were chosen according to Section 4.3.3.

An advantage of the used LSPI method is that no additional trajectories are needed for the policy iteration, but a set of transitions from the training video and the gained rewards are sufficient. We chose such a set with approximately 12.000 transitions. To each transition there corresponds one of 3 possible actions: Move approximately 30 cm forward, turn 45 degrees left or right. We chose LSTD as the value approximator. The robot receives a positive reward if a transition ends at the goal. Contrary, if the robot comes too close to a wall it receives a negative reward. After training, the robot follows a greedy policy.

### 4.4.2 Robotic Experiments

We performed two types of experiments: the first set of experiments was done with the robot (40 runs, Figure 4.7). These experiments are time consuming and are not suited for a large scale analysis. Therefore, we performed a second set of experiments using optimal SFA features (2000 runs).

In both settings the goal areas were placed for half the runs in the middle of the room and for the other half in the lower right corner (red painted area). The training set was the same for both experiments the only difference was that we replaced the learned SFA features with optimal SFA features for the large scale experiment. The control behavior was in both experimental settings very

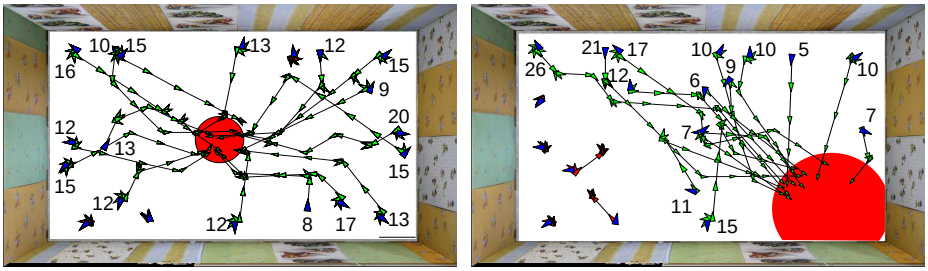


Figure 4.7: The recorded robot trajectories using SFA features. The plot shows 40 trajectories with random starting positions and viewing angles. Successful trajectories are green and unsuccessful trajectories are red. The goal area is plotted in red. The numbers indicate the number of steps the robot needed to reach the goal.

similar. For example, in both experiments starts in the lower left corner were problematic if the goal was in the middle.

The trajectories for the robot experiments are shown in the figure. Red trajectories mark the unsuccessful runs. In numbers, the success rate for the robot was 17/20 (middle) and 15/20 (corner). For the optimal features we achieved a success rate of 907/1000 (middle) and 986/1000 (corner).

### 4.4.3 Simulator Experiments

By simulations we tested the degree to which the performance depends on the room structure. We used 3 different room settings: (1) a quadratic room, (2) a quadratic room with a block in the middle (O-Room) and (3) a two-room environment (the two-room is shown in Figure 4.2). The SFA was trained with a weaker model class (around 3000 support vectors) and the images were obtained by a simulator. To simplify the policy iteration step, we used a discrete state space (a grid of equidistant points), where each state corresponds to exactly one image. In this setting it is easily possible to apply other PI methods. Besides LSPI, we used an *optimistic policy iteration* version of *Sarsa* [29]. We used 180 SFA features for all but the O-Room for which we used 324 features. Each experiment was repeated 10 times. The following table shows the mean and standard deviation of the regret (absolute value).

| Task | Qu. Room | O-Room | Two-Rooms |
|---|---|---|---|
| ♯States | 8192 | 7040 | 15744 |
| LSPI | 12.9/2.6 | 36/3.3 | 28/1.7 |
| O-Sarsa | **11.5/0.9** | **16.5/1.2** | **27/2.5** |
| LS | 2.7 | 6.8 | 18 |
| Random | 32.7/0.2 | 55.5/0.4 | 53.5/0.1 |

The **LS** row shows the performance if no PI is used but a least-squares fit of the true value function (about the best possible policy for the given feature space). The **Random** row shows the results for a random policy.

One can observe that the least-squares fit is considerably better than the policy iteration methods. That means that the SFA features allow in principle much better policies, but the policy iteration does not converge to them. Finally, the performance decreases with growing room complexity (e.g. LS row). Nevertheless, the method is in principle applicable to other enviroments. However, the computational demand increases with increasing complexity of the enviroment.

## 4.5 Conclusion

We presented a new approach to robotic navigation. The central idea is the use of a feature space consisting of trigonometric functions as the basis for reinforcement methods. The motivation for the trigonometric functions is their usefulness for function approximation (e.g. approximation of the value function). It is possible to learn with SFA a projection from sensory data into this feature space. As a result we are able to learn a control process for almost any kind of sensory systems. We demonstrated this on a simple real-world task: A robot learned to navigate using camera images only. Furthermore, we demonstrated with a simulator that the method is also applicable in more complex environments if the computational power is sufficient. In general, the limiting factor is the computational power, as the number of support vectors directly relates on it. We did not focus on computational aspects and we assume that a big improvement can be achieved with more sophisticated numerical methods.

We think that our approach has the potential to allow robot navigation in manifold environments. The advantages of the approach are: (1) The system adapts to the environment quiet fast (in our setting around 6 hours of recording) and can therefore be applied to very different settings. (2) The approach is not restricted to visual data. Essentially, without any adjustments it would be possible to use any other sensory system, e.g. ultrasonic sensors.

## 4.A Approximation Results

In this section we discuss approximation properties of linear functions that are applied on optimal SFA features. We are interested in the following question: Can we approximate a large class of functions well in an appropriate norm in the limit?

In the following let $K := [a_x, b_x] \times [a_y, b_y] \times [0, 2\pi]$ describe the rectangular room and the viewing angle. Let $(\boldsymbol{\phi}_i)_{i \geq 1}$ be the set of optimal features, let $\boldsymbol{\phi}_0$ be the constant function $\mathbf{1}$ and let $\mathscr{M}$ be the set of functions for which $f(x, y, 0) = f(x, y, 2\pi)$.

### 4.A.1 Denseness

We start with the limit case. The function classes we are interested in are the continuous functions and the $L^p$ functions. In approximation theory a lot of powerful theorems are known for denseness. To apply these theorems it is typically sufficient to check a number of simple properties of $\mathscr{G}$

**Theorem 20.** *The space spanned by the $\boldsymbol{\phi}_i$ is dense in the continuous functions $(C(K) \cap \mathscr{M}, ||\cdot||_\infty)$, $(C(K) \cap \mathscr{M}, ||\cdot||_p)$ and in $(L^p(K) \cap \mathscr{M}, ||\cdot||_p)$, with $1 \leq p < \infty$, if the assumptions from [34] are fulfilled (Section 4.3.4).*

Dense means that we can approximate any of the $C(K)$ and $L^p(K)$ functions arbitrarily well in the corresponding norm. In particular, we can approximate any bounded function in the p-norms arbitrarily well.

*Proof.* First the case $(C(K) \cap \mathscr{M}, ||\cdot||_\infty)$. We use a version of the Stone-Weierstraß Theorem ([27][Th. 4.3]). We denote the space spanned by the $\boldsymbol{\phi}_i$ with $\mathscr{G}$. We need to check that **(a)** $g(x) = 0$ for all $g \in \mathscr{G}$ implies $f(x) = 0$. Due to the constant 1 which we included there is no $x$ for which all $g$ are 0 and (a) is fulfilled.

**(b)** $\mathscr{G}$ needs to separate any two points separated by $f$. $f$ does not separate $(x, y, 0)$ and $(x, y, 2\pi)$, therefore we can ignore these points. Further, points at the boundary of the rectangle are not a problem, as the slowest features in $x$, respectively $y$ direction seperate them. For any other set of points one can argue that if a cosine assigns the same value to the positions then a cosine with a frequency increased by 1 does not.

**(c)** $\mathscr{G}$ must be an algebra. Due to [27] it is sufficient to check that $g_1, g_2 \in \mathscr{G}$ implies $g_1 g_2 \in \mathscr{G}$. Each $g_i$ is a linear combination of $\phi = \cos(ax)\cos(by)\cos(c\theta)$, respectively $\sin(c\theta)$ factors, where $a, b = n/2, n \in \mathbb{N}$ and $c = n \in \mathbb{N}$ [34]. Taking the product $g_1 g_2$ we get linear combinations of terms $\phi\tilde{\phi}$. Therefore, if every $\phi\tilde{\phi}$ is contained in $\mathscr{G}$ then every $g_1 g_2$ is contained in $\mathscr{G}$. Let, $\phi\tilde{\phi} := \big(\cos(ax)\cos(\tilde{a}x)\big)\big(\cos(by)\cos(\tilde{b}y)\big)\big(\cos(c\theta)\cos(\tilde{c}\theta)\big)$. Each such cosine product, e.g. $\cos(ax)\cos(\tilde{a}x)$ is a linear combination of the $\phi$'s. Let $a \geq \tilde{a}$, then $\cos(ax)\cos(\tilde{a}x) = 1/2\cos((a+\tilde{a})x) + 1/2\cos((a-\tilde{a})x)$ and $a \pm \tilde{a} \in \mathbb{N}$. The cases $\cos(c\theta)\sin(\tilde{c}\theta)$ and $\sin(c\theta)\sin(\tilde{c}\theta)$ are similar. In summary $\mathscr{G}$ is closed under multiplication and an algebra. (a),(b) and (c) are the needed properties for the Stone-Weierstraß Theorem to apply.

The case $(C(K) \cap \mathscr{M}, ||\cdot||_p)$ follows directly as $||f-g||_p \leq A||f-g||_\infty$, with $A$ being the size of the rectangle times $2\pi$. Further, $(C(K) \cap \mathscr{M}, ||\cdot||_p)$ is dense in $(L^p(K) \cap \mathscr{M}, ||\cdot||_p)$ (e.g. [81]). □

# Part III

# Control in Partially Observable Markov Decision Processes

# CHAPTER 5

## Attention Driven Memory


## Abstract

Categorization is a skill which is used extensively in everyday life and is therefore an important aspect of human cognition. Consequently a variety of studies exist which address the topic and revealed that diverse factors affect human categorization performance. A critical but not extensively studied factor is time. Imagine watching a basketball game for 30 minutes. In this period of time plenty of actions will take place resulting in diverse impressions which make you afterwards categorize the game as interesting or boring. Such a categorization task is very similar to a time series classification task in the context of machine learning. In the field of machine learning a phenomena called "vanishing gradient" is known which makes it generally hard to solve such a categorization task. A prominent method that overcomes this phenomena is the long short term memory which basically consists of a memory that is controlled by two gate units which can be interpreted as adaptive encoding and recall units. Critical points which make the processing of the structure differ from human processing concern the encoding and the storage: (1) The structure is built to massively store information instead of carefully selecting few impressions for storage in memory. (2) Reweighting of stored information due to changing constellations is not possible. Coming back to the example this would mean that a nice action at the beginning of the game has a strong impact on your categorization independent of what kind of actions - might they be impressive or not - followed afterwards. In this work we tackle these points through introducing an attention mechanism which drives the encoding and the storage of the structure. We analyze the model behavior in category learning tasks.

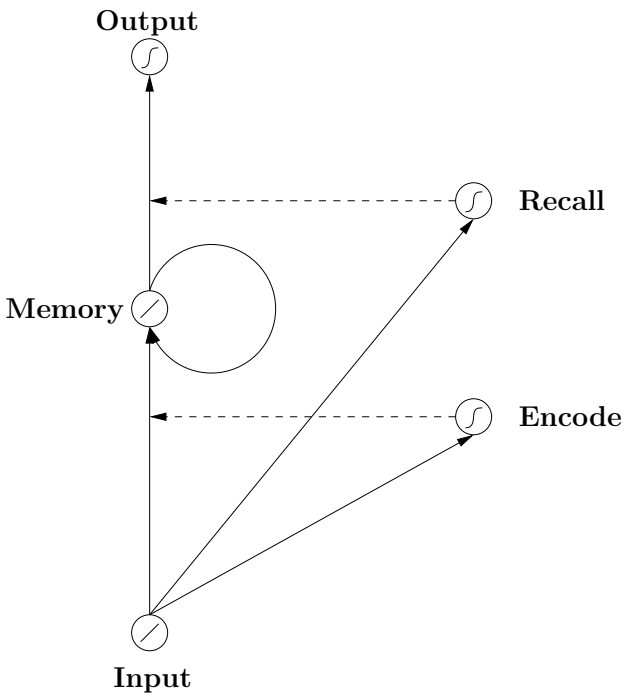


Figure 5.1: Interpretation of the LSTM. The central element is the memory which is controlled by adaptive encoding and recall units. Encoding/recall activation scales the memory input/output through a multiplicative relation to the ordinary pathway (dashed line). The hidden layer (memory, encode, recall) is fully connected (not drawn due to readability).

## 5.1 Introduction

Diverse factors like category dimensionality [67], covariance complexity [4] or, for rule-based task, if a rule is conjunctive or disjunctive [16] are responsible for human categorization performance. Another important but not extensively studied factor is time. Categorization tasks which involve the temporal domain come up all the time in everyday life, think for example of the different ways you can take to your work place. You will surely have categorized them according to the time you need to arrive at work.

In the machine learning field similar classification problems are studied, whereas the machine learning methods here face a similar problem like a human. The information they need for classification is spread over time so not all of the relevant information is available at one moment. Just like a human these methods have to memorize relevant information to yield good performance in the task.

Interestingly, studies in the machine learning field revealed that such classification tasks are generally hard to solve. This mainly relies on the so called *vanishing gradient problem* ([39], [12] and [40]). This "problem" comes up when stored information decays with time. This results in the effect that information that might be strongly related to a later signal being downscaled with time and

finally when the later signal comes the original information can no longer be distinguished from noise. This way the relation between the signals cannot be revealed, respectively learned. This has dramatic effects on the performance of methods. Due to this effect traditional recurrent networks[1] are hardly able to relate two signals which are 15 time steps or more apart from each other.

A prominent machine learning method that overcomes this problem is the Long Short Term Memory (LSTM, [38]). The LSTM is a recurrent neural network structure which basically consists of a memory unit that is controlled by adaptive encoding and recall units (see Figure 5.1 and next section). Similarities of the LSTM structure to recent neuropsychological models of working memory exist as stated in [61].

Despite these similarities some obvious differences still exist between the processing through the LSTM structure and human processing of information. Two points are especially critical: (1) The LSTM tends to massively store information in its memory in contrast to the very selective storage in humans. (2) Information that is once stored cannot be reweighted due to changed conditions.

We address these two points through introducing an attention mechanism which drives the encoding unit and rescales the memory content when needed.

Furthermore, we analyze the behavior of this process model in category learning tasks.

## 5.2 Long Short Term Memory (LSTM)

Figure 5.1 shows the LSTM structure [38]. The key element is the memory unit in the center of the LSTM. The unit uses the identity as an activation function and is self-connected with a recurrent connection. This recurrent connection has a weight of one which guaranties that no information is lost, respectively information is conserved and so the vanishing gradient problem does not apply.

A problem that occurs directly from the conservation of all information which is encoded in the unit, is that the capacity of the memory unit will be quickly exceeded. This problem is overcome through introducing an encoding unit which controls the inflow of information into the memory. Beside that a recall unit is used to control the flow of information out of the memory cell back into the network.

The control of the inflow and outflow is achieved through scaling the input to the memory respectively the output from the memory, with the activation of the encoding/recall unit. Whereas both units got a sigmoidal activation function in $[0, 1]$. Thus when the activation of these control units tends to zero the information flow is stopped.

An important aspect about the encoding and recall units is that they are adaptive and through training learn when to store information and when to use stored information.

## 5.3 Attentional Driven Encoding and Storage

Encoding in human memory is driven by attention. This has the advantage that the memory can be used according to current needs or objectives. Fur-

[1] For an introduction to neural network models see [37]

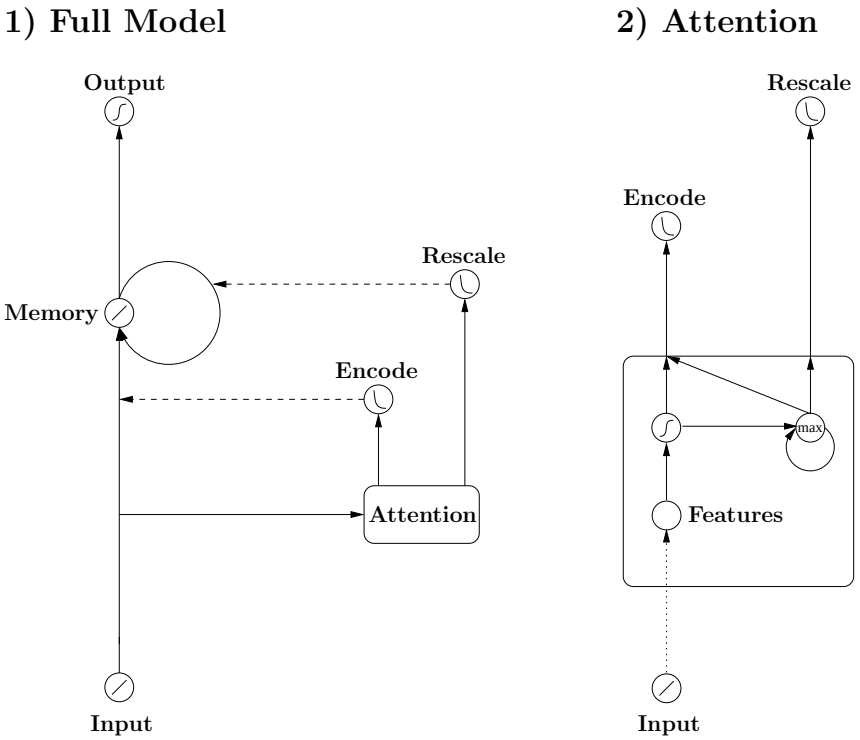


Figure 5.2: 1) Full Model: Modified LSTM structure. Like in the LSTM the central element is the memory cell and the input to the memory is controlled by an encoding unit. In difference to the LSTM the encode and rescale units are driven by an attention process which weights the input according to its relevance and reweights the memory when new constellations come up (e.g. a new signal attracts most attention). Encode and rescale units have exponential activation functions which are used to emphasize strong attended input and downscale input of low attention value. 2) A detailed view on the attention box of the left part. The input activates a set of features which lead to an attention value. This attention value is compared with the stored maximal attention value and replaces this when it is higher. It further drives together with the maximal attention value the encode unit. The rescale unit on the other hand is driven by the difference between the new and the old maximal attention value.

thermore, storing information selectively reduces the needed memory and this way it is easier to detect a relation between past attended and present events. A disadvantage is, that when a relation exists between past and present events, but attention is not drawn to the relevant past events then the relation will hardly be detected. Encoding in the LSTM varies in these points from human encoding. The LSTM massively stores information. Therefore problems arise with the memory capacity. Additionally, through this massive storage the LSTM is on the one hand able to detect a wide variety of relations, however, on the other hand it takes a long time to extract relations. Thus, a well working attention mechanism should be able to improve the LSTM performance, make its encoding behavior more human like and thus make it a memory structure

whose processing is more similar to human information processing.

We start from a basic and general assumption of how the memory content should be computed out of input and attention values at different time steps. It shows up that critical elements of the attention process are predetermined from this first assumption and thus at the core of the attention structure no variations are possible when the defined equation (5.1) should hold.

Basically we want the memory content to "concentrate" on few strongly attended information and to downscale the impact of information which has a low attention value relative to the strongly attended signals. Due to the vastly reduced amount of stored information that must be processed, detecting relations between attended signals should be much easier. Furthermore, we don't want the storage of information to rely on the special time step when the information arrives. So basically the memory content at time T should look like this.

$$mem_i(T) = \sum_{t=1}^{T} in_i(t) \cdot \phi(A_{max}(T) - A(t)). \tag{5.1}$$

$\phi$ denotes a function which operates on the relative value between the strongest and the actual attention value at step t. If the relative value is high the input should be strongly downgraded.

When storage is limited and thus not every $A(T)$ and $in_i(t)$ can or should be stored it is not simply possible to rescale the stored information when $A_{max}$ changes. What we need is a function $\phi$ that makes it possible to rescale the stored information in the way that

$$\begin{aligned} mem_i(T) = & \overbrace{K \cdot \ mem_i(T-1)}^{\text{rescale}} + \\ & \underbrace{in_i(T) \cdot \ \phi(A_{max}(T) - A(T))}_{encode}. \end{aligned}$$

To do so it must be possible to transform the '-' into a multiplicative factor, in other words $\phi(A_{max}(T) - A(t)) = \phi(A_{max}(T)) \cdot \ \phi(-A(t))$. This only holds for the exponential function thus $\phi = exp$. After introducing an additional minus and a scaling factor $\sigma$ we get

$$mem_i(T) = \sum_{t=1}^{T} in_i(t) \cdot exp(-\sigma(A_{max}(T) - A(t))).$$

The rescaling factor K computes now to

$$K = exp(\sigma(A_{max}(T-1) - A_{max}(T))).$$

To guarantee that the equation holds in every step we only need to store the memory value $mem_i(T)$ and the highest activation value $A_{max}(T)$ that has come up till the time step T.

In the next section we present a model which is based on these equations.

### 5.3.1 Model Description

Figure 5.2 shows the model structure. The input is passed to the attention process and to the memory. The attention process computes an attention value

for the momentary time step due to the input. According to this attention value it drives the encode and rescale units which control if the new input is stored, respectively how "strong" it is stored, and how the memory content must be reweighted due to changed conditions.

Diverse factors exist which drive our attention in one special moment. An example is our physical condition. When we are hungry or thirsty totally different objects attract our attention than when we are tired. Also totally different conditions like a task we are working on at the moment or personal preferences affect what attracts our attention. The study of the influence of these points is beyond the scope of this work. Instead of realizing an attention mechanism which is driven by all these factors we used some abstract variables $F_i(t)$ which might be used to represent a variety of features which are relevant to control attention. The only condition the variables must fulfill is that the values the variables can take lie in a bounded interval.

Due to these features the attention value at time step t calculates in our method as follows:

$$A(t) = f(max(\hat{w}_i F_i(t) + b_i)).$$

Here max is the maximum function and f is a logistic sigmoidal function. Variable $F_i(t)$ is dependent on the input and as stated above simulates a feature detector that signals if a feature i is present at time step t (in simulations the range of $F_i(t)$ was chosen to be in $[0, 1]$). A short remark: the function to calculate $A(t)$ must not have the upper form, one might for example also use a weighted sum of the different feature activations. Finally, in the above equation $\hat{w}_i$ is the weight of feature i and $b_i$ a bias. Both are adaptive and modulated through learning.

Contrary to the choice of $A(t)$ the following variables are predefined through our initial assumption (equation 5.1).

As stated above the maximal attention value which appears until step t is needed to encode and rescale:

$$A_{max}(t) = max(A(t), A_{max}(t-1)).$$

Building up on the value $A(t)$, $A_{max}(t)$ and $A_{max}(t-1)$ the encode and the rescale value can be calculated:

$$encode(t) = exp(-\sigma(A_{max}(t) - A(t))),$$
$$rescale(t) = exp(\sigma(A_{max}(t-1) - A_{max}(t))).$$

Finally the activation of the memory at time step t calculates as follows:

$$mem_i(t+1) = in_i(t) \cdot encode(t) + mem_i(t) \cdot rescale(t).$$

And the output of the network:

$$out(t+1) = f(\sum_i w_i mem_i(t) + b).$$

In this simple setup adaptive parameters that must be learned are the $w_i$ values which weight the impact of the memory content to the output of the network and the $\hat{w}_i$ values which weight the impact of the different features to

the attention value at one time step. The latter ones are the interesting ones, because they define how the network uses its memory. Another remark: in the LSTM input is bundled through a weighted sum and thus concentrated in one memory unit. This can also be done here through introducing another weight layer between the input and the memory. The attention at one special moment will in this case have a large impact on the learning of the input weights. Input units that are active when attention is high and thus the encode unit stores information will be strongly affected by a learning process.

Training the network can be done with any gradient descent algorithm which does not rely on higher derivatives (s. below). We used back-propagation through time [37] for training. The derivatives are straight forward to calculate. We only want to single out two of them. First, the derivative of $\frac{\partial A_{max}(t)}{\partial A(t)}$ because the maximum function is involved. The maximum function is differentiable, however its derivative is not continuous:

$$\frac{\partial A_{max}(t)}{\partial A(t)} = \begin{cases} 1, & A(t) \geq A_{max}(t-1) \\ 0, & otherwise \end{cases}.$$

This makes no problem for calculating the gradient and thus to train with a gradient descent method which does not rely on the second or higher order derivatives.

The other derivative which we want to single out is

$$\frac{\partial mem_i(t+1)}{\partial mem_i(t)} = rescale(t).$$

This one is important because of the vanishing gradient problem ([39], [40] and [12]) which makes it hard to process temporal information. Like in the original LSTM in our model this problem does not apply. The reason for this is that the rescaling for stored information is bounded, as long as the range of A(t) lies in a bounded interval (for our case $[0, 1]$), information, respectively the gradient, cannot vanish. The upper bound for rescaling is 1, which is trivial and the lower bound can be inferred in the following way.

Because $A_{max}(j) - A_{max}(i) \geq -1$ for $i > j$ due to the choice of the function f it holds that

$$\begin{aligned} \prod_{k=i+1}^{k=j} rescale(t) =& exp(\sigma(A_{max}(i+1) - A_{max}(i))) \cdot \\ & \cdot \ldots \cdot \ exp(\sigma(A_{max}(j) - A_{max}(j-1))) \\ =& exp(\sigma(A_{max}(j) - A_{max}(i))) \\ \geq& exp(-\sigma). \end{aligned}$$

the overall rescale value summed over all time steps is lower bounded by $e^{-\sigma}$. Thus stored information cannot be rescaled arbitrary low and thus will in all situations still have a noticeable effect. This leads to the statement that the vanishing gradient problem does not apply.

## 5.4 Simulation

We evaluate our model in category learning tasks. The tasks we consider here are rule-based category learning tasks [5]. In the simulations we presented the

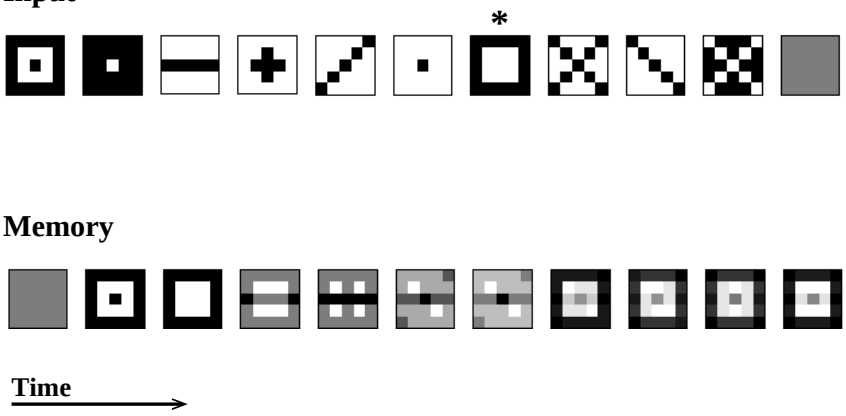


Figure 5.3: We presented a sequence of 15 steps to a network. In the first 10 steps cards were drawn randomly with equal probability from a pool of 20 cards, whereas each card could appear only once. In the final 5 steps no card was presented. At step 15 the network got an error signal for it's classification response. In the upper part one such sequence is shown. The sequence is presented from left to right with one card per time step. The relation of the sequence and the class was deterministic. If the black square card (* in the figure) was presented the class of the sequence was 1 otherwise the class was 0. The variables $F_i(t)$ where chosen to be reactive to only one of the cards, so we got 20 variables (one for each card). This way the calculation of $A(t)$ was trivial and the emphasis laid on the weighting over the different time steps. The according weights of $F_i(t)$ were initialized with a value of 0.2 whereas the weights between the memory and the output layer were chosen randomly in $[-0.1, 0.1]$. The bias values where negatively initialized (-2) to initially hold attention values low. The lower part shows the memory of a trained network (20 000 training steps, mean squared error $< 0.1$). At the position of the correct card the memory is rescaled and the stored values are strongly downgraded. At the same moment the input is stored with a high attention value and overshadows the other input signals.

network a set of cards (see Figure 5.3) whereas at each point in time only one card was presented. Thus the networks had to learn to memorize information about the seen cards to solve the categorization task. The relation between the presented cards and the category was deterministic, if a special card was present in the sequence then the according category was one, otherwise zero. Each card was represented through a 5x5 matrix of input activation. To each input unit directly one memory unit corresponded, thus we also had a matrix of 5x5 memory units. Input to the memory, respectively the rescale of the memory, was, however, controlled by just one unit. So a single attention mechanism was responsible for the inflow of information to the 5x5 memory units.

In the first simulation we were interested in how the memory is used from a network which had learned to solve the categorization task (see Figure 5.3). It shows up that the network has learned to extract the relevant feature and stores it in the memory whereas the other input elements are downscaled when

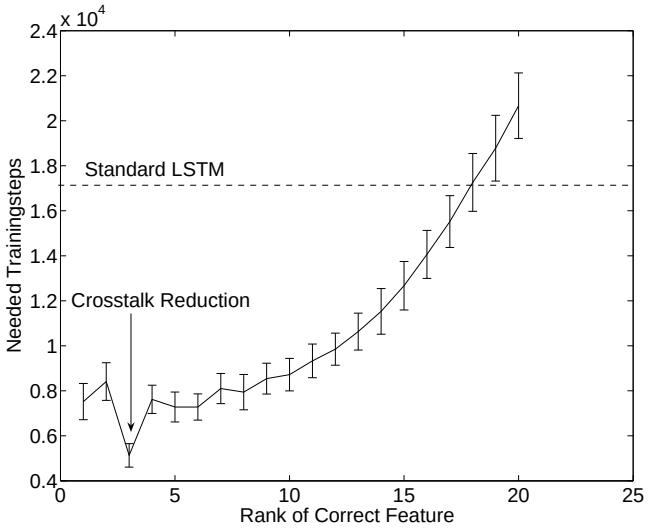


Figure 5.4: We measured how the initial weighting of the correct feature and its relative weighting to other features effects the performance of the model. The signals and features were chosen like in simulation described in Figure 5.3. We used the number of training steps which are needed to bring the mean squared error below 0.1 as a measurement for the performance of the network. We initialized the feature weights in the following way: $\hat{w}_i = 0.1 \cdot i$ and changed the position of the correct feature. So in the worst case the initial weight was 0.1 and in the best 2.1 for the correct feature. For each setup we made 100 runs. Notice that beside the effect of the rank also interference effects came up because the different cards we presented had similarities. Especially the cards which were similar to the black square card (this was the relevant feature) had an impact on the performance. This becomes most obvious in the plot at position 3. Here the feature responsible to detect the black square card with a dot in center (in figure 5.3 the card in the upper left) was initialized with a low weight which resulted in a low attention value when the card came up. Because this card is very similar to the relevant feature the reduced attention to this card boosted the performance of the network. We also made a run with the standard LSTM to compare the performance. We have chosen the same learning rate as in our model and initialized the other parameters similar to [38]. The mean number of training steps was 17640 and standard deviation was 781.28 (dashed line in the figure).

the relevant feature is seen.

We made a second experiment to study how the performance of the network changes when attention is directly drawn towards the correct feature against the case when attention is high for irrelevant features and low for the correct one. Figure 5.4 shows the results. An interesting side effect became apparent in this simulation. The cards we have chosen are in parts similar. During the simulation it became clear that this has a considerable impact on the performance of the network. Especially, as the performance was strongly boosted when a card which was nearly the same as the relevant card attracted no attention (the arrow in

the figure).

## 5.5 Conclusion

An important factor in categorization is time. Categorizing events which are spread over time puts humans in a similar position as machine learning models which have to classify time series. A prominent machine learning method, the Long Short Term Memory, which successfully processes time series has some similarities to recent neuropsychological models of the working memory. However different points make its processing behavior vary from human processing. Especially, it tends to massively memorize information and is not able to reweight memorized information due to changing constellations. We showed that these points can be tackled through introducing an attention mechanism which drives the encoding and the storage process. Especially, when the attention mechanism is working well, the performance of the network increases and learning becomes faster than with standard LSTM. However, the network performance strongly relies on the quality of the attention process (in our simulations an exponential relation shows up), thus when attention is attracted by irrelevant signals and the relevant signals are omitted then performance dramatically decreases. Beside this point the simulations revealed that the network is very sensitive to the effects of interference between input signals.

CHAPTER 6

# MDP Unfolding through Hidden State Extraction


## Abstract

Reinforcement learning has a long history in studies of Markov Decision Processes (MDPs) and has produced a variety of algorithms. In recent times the focus of attention has switched towards partially observable MDPs (POMDPs), which are more general but harder to handle. Consequently, algorithms constructed for MDPs are not applicable or do not perform well. In this work we consider POMDPs for which the environment is Markov, but the learner has restricted access to the state information. In particular, we address the problem where observations are available, which contain information about the true MDP state. For these problems we propose a method which resolves the Markov structure and enables us to apply arbitrary MDP methods to solve the control problem. Our approach builds up on studies about long term dependencies. We show in simulations that our method is able to resolve MDPs even if a critical observation is about 100 000 steps before a related effect.


## 6.1 Introduction

*Markov Decision Processes* (MDPs) are a popular and successful approach for modeling stochastic environments in which actions must be performed. In this work we are concerned with MDPs in which not all properties of the environment are encoded in the observable states. As an example one might think of a camera equipped household robot which has to cook, wash the cloth, clean the room and so on. The camera image can be used as an observation corresponding to the MDP states. Yet, this observation will in general be ambiguous, because: (1) the image will be distorted by noise (light, etc.) and (2) because hidden states might exist in the environment. For example, the robot might have washed the cloth and put it outside on a line or it might have started cooking water and is cleaning the room in between. In these cases the robot cannot infer these

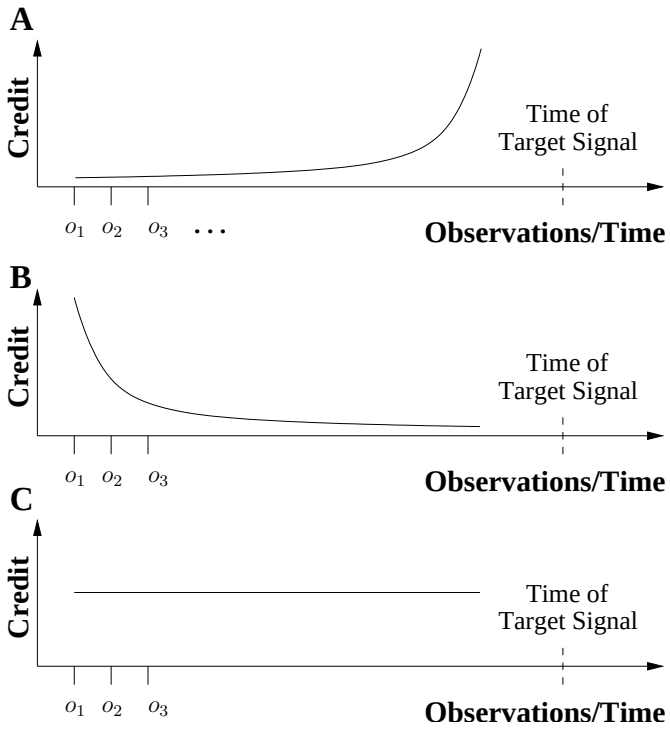


Figure 6.1: The Figure shows the vanishing gradient problem schematically. The three plots show how credit is distributed to observations over time. A,B,C are cases for three different dynamical systems $f$. The plots differ in the volume transformation of the system (A: $|\det J| < 1$,B: $|\det J| > 1$,C: $|\det J| = 1$).

properties from the current observation (camera image) but must remember them. In nearly any everyday setting hidden-states like these are present and a system that should be able to interact in human environments needs to deal with these.

In this work we address this topic of hidden states extraction. The setting is a special case of a *Partially Observable Markov Decision Process* (POMDP). The definition of a POMDP includes a number of other cases like fuzzy sensors (case 1 above) that we do not address here. We are essentially interested in scenarios in which critical observations are available at some time which make it possible to infer the environment's true state. To derive a sensible control strategy it is needed to store, respectively memorize these observations. A control strategy can then be applied to the directly available observation, e.g. the camera image, and the memorized signals.

The problem of memorizing information is strongly related to problems in time-series analysis, where critical signals are often only shortly available. Typically, such signals do not occur immediately before the target and learning systems must memorize the information for a longer period. Problems of learning can be significantly hindered by the presence of such long-term dependencies [13, 10, 40]. In general, a sequence of observations $o_1, \ldots, o_T$ is said to exhibit

long-term dependencies if the observations at a time $t$ are significantly dependent on observations $o_{t_0}$ at much earlier time $t_0 \ll t$. Long-term dependencies introduce learning problems, because *credit assignment* over time becomes increasingly complicated with longer sequences. The term credit refers to the "impact" an input signal has onto an observed signal. In differentiable systems the assigned credit is related to the partial derivative of the costs with respect to the input signal. Essentially, the problem with credit assignment for long sequences is that dynamical systems need to store information about all former observations in order to learn dependencies. If, for example, at time $t$ no "information" about an observation $o_{t_0}$ is left in the system then it will be impossible for a learning algorithm to detect a dependency between $o_t$ and $o_{t_0}$ at time $t$.

Standard methods for time series processing facilitate this problem, because they "weight" information not uniformly over time, but assign an exponentially higher weight to recent information. Such system are therefore unable to detect dependencies that are more than a few steps apart. For continuous systems the effect can be analyzed by utilizing the Jacobian $J$ of the dynamical system, where $J_{ij} := \frac{\partial s_{t,i}}{\partial s_{t-1,j}}$ and $s_t$ is the $n$-dimensional state of the system at time $t$. The Jacobian is a measure for the influence of the state of the system at time $t-1$ to the new state at time $t$. If the Jacobian is in a given matrix norm smaller than 1 for each possible state $s$, i.e. $\sup_s ||J_s|| \leq c < 1$ then the influence of the state $s_0$ onto the state $s_t$ vanishes exponentially fast due to the chain rule:

$$||D_{s_0} f^{(t)}|| = ||D_{f^{(t-1)}s_0} f D_{f^{(t-2)}s_0} f \dots D_{s_0} f|| \leq \sup_s ||D_s f||^n = \sup_s ||J_s||^n \leq c^n,$$

where $f$ describes the system evolution, $f^{(t)}$ denotes the $t$-times application of $f$ and $D_s$ denotes the differential at point $s$. In the vanishing gradient literature the volume transformation from the Jacobian is usually measured with the determinant and not with a norm [13, 40]. In dependence of this measure one can distinguish three cases: (A) $|\det J| < 1$, (B) $|\det J| > 1$, (C) $|\det J| = 1$. In Figure 6.1 the three cases are displayed schematically. On the $x$-axis the observations at different time steps are drawn. The $y$-axis describes the credit, which is assigned to the observations. Case $A$ corresponds to the so called vanishing gradient problem. This is the usual case for dynamical systems. The signals close to the target time get the most credit whereas far apart signals get exponentially less credit. Therefore, it is hard to extract important observations if they are far apart from the target time. Case B is the contrary. In this case the first signals get a high credit. Case C is an important special case which is utilized to solve the vanishing gradient problem [38, 41]. In this case each observation gets the same credit and far apart events can be related.

In [10] it is shown that similar problems occur for *Hidden Markov Models* (HMMs, [62]). The effect is the same as with other dynamical systems, however, it is needed to argue differently for HMMs. A different approach is needed because the sup-norm of $|J| = 1$. $|J| = 1$ holds, because the system evolution is given by a *stochastic matrix*. Therefore, there is in general "no loss" of information, but a diffusion of information. In [10] the problem is broken down to the ergodicity of the transition probability matrices and it is shown that this diffusion of context and credit information hampers both the representation and the learning of long-term context in the state variables.

The effect of the vanishing gradient problem is tremendous, because of the exponential information loss. For example, if a classical recurrent neural network

is used then the system is able to learn to relate signals which have a gap of about $\sim$ 15 time steps between them at best. In [38] a neural network structure named *Long Short Term Memory* (LSTM) was proposed that solves the vanishing gradient problem. The idea of the LSTM network is quite simple. To circumvent the information loss a subsystem is introduced for which $|\det J| = 1$ holds. This subsystem is called a memory unit. The memory unit enables the LSTM network to relate far apart signals. Despite the simplicity of this approach it has strong practical implications. The LSTM network is, for example, able to relate signals that are up to 1000 steps apart.

Nearly all approaches to handle POMDPs do not address the vanishing gradient problem (e.g. [22, 56, 57, 86, 1, 47]). As a consequence these methods have problems with environments in which critical signals must be memorized for a longer period (whereas longer means for more than about 10-20 steps). Actually, many of the methods are designed to handle other problems related to POMDPs, especially the problem of noisy sensory and the problem to construct an underlying MDP from scratch [22, 56, 57, 86]. The latter problem is a very hard one. Some of the methods try to solve it through modeling the MDP structure with a HMM. The methods try to learn the state structure of the HMM with the help of statistical tests. If the used test says that an observation actually corresponds to two different MDP states then the HMM structure is expanded about one node (for more details see [86] where these methods are summarized). As a consequence of the complexity of the problem the methods are very slow and not usable for larger state spaces, respectively for the detection of dependencies with a longer time gap.

Another approach is taken in [6]. In this work the LSTM network is used to predict the value function. However, this approach has a major disadvantage. Learning the value for each state action combination is a quite hard task for a network. The network parameters must be adjusted to give the correct value in each state (regression). It is known that for LSTM and other recurrent networks the regression problem leads to difficulties if too many outputs must be predicted. This can easily be seen with a look at the empirical risk function, which is minimized (typically: $1/2 \sum_i \sum_t^T (y_{i,t} - f(x_{i,t}))^2$, with $y_{i,t}$ being the observed output in sequence $i$ at time $t$, $T$ being the length of the sequences, $x_{i,t}$ being the input and $f$ the network). If $T$ is large, then a single term $y_{i,t} - f(x_{i,t})$ will have a negligible influence. The result is that a system $f$ *concentrates* on the mean response $y_{i,t}$. It is often unable to learn a different behavior in non standard cases (unbalanced data). In [6] this results in a reduced performance, which makes the method only usable for small size systems. In simulations the proposed system was able to solve tasks with 60-70 steps of time gap between the signals, respectively 60-70 observations in the POMDP (in contrast to 1000 time steps, which have been reported for other tasks [38]). For special cases where the MDP allows only actions in a few states (in [6] it was one state) the performance gets better (between 100-1000 states/steps).

Essentially, the problem of the approach from [6] is that the LSTM is used for controlling. If the usage of the LSTM would be restricted to detecting important patterns and a second method would be used for the controlling itself, then the performance could be dramatically improved. However, coupling the LSTM with a state space based method like a MDP based algorithm is not easy, because neural networks and state space models are in a sense quite different systems. In this work we make another approach with a state space system,

which accounts for the vanishing gradient problem. Due to the natural relation to standard reinforcement learning approaches the coupling of the systems is straight forward and the performance improves in the order of magnitudes (in simulations we solved tasks with up to 100 000 states, respectively steps between the signals). The major advantage against the LSTM approach is that the controlling can be performed by any of the methods constructed for MDPs. In simulations we used *Temporal Difference Learning* (TD(0), [77]), *Least-Squares Temporal Difference Learning* (LSTD, [19, 18]) and MDP based exploration strategies [78].

The system we use is a so called Memory based Input-Output Factorial Hidden Markov Model (MIOFHMM,[41]). It is like the LSTM constructed to solve the vanishing gradient problem. We use the most probable state of the MIOFHMM (Viterbi) to expand the observation vector. The approach has some similarity to [47]. In [47] the agent has the possibility to set a number of memory bits through actions. However, the approach has a major disadvantage. It does not account for the vanishing gradient problem. Hence it is only able to relate signals which come close to each other.

In our opinion the lack of knowledge of the vanishing gradient problem and the corresponding literature hampers the development of methods for POMDPs. The main goals of this work are therefore: (1) To demonstrate the importance of the subject for POMDPs. We do so by solving two POMDPs: One in which we increase the solvable state-space size from roughly 100 to 100 000 and one in which we solve a complicated maze task that currently cannot be solved by other methods. (2) To provide an approach that accounts for the vanishing gradient problem for solving POMDPs. The usual tool to address the vanishing gradient problem is the LSTM network. Yet, the LSTM network and the state-space structure from (PO)MDPs do not fit well together. We use the MIOFHMM as an alternative because it fits naturally to the POMDP setting. However, the MIOFHMM has not been the focus of research as it has few advantages over the LSTM for time-series processing. Consequently, the structure is not well studied. For example, in [41] the exact *Expectation-Maximization algorithm* (EM, [26]) is used. However, as the MIOFHMM is a factorial HMM, the exact EM is very costly and most often intractable [35]. Due to the usage of the exact EM the usefulness of the MIOFHMM is very limited. We address this point, we furthermore describe two problems of the MIOFHMM and we provide a solution for them.

## 6.2 Preliminaries

A *Partially Observable Markov Decision Process* (POMDP) consists of a finite state space $\mathbb{S} = \{s_1, s_2, \ldots, s_{n_\mathbb{S}}\}$, a finite set of actions $\mathbb{A} = \{a_1, a_2, \ldots, a_{n_\mathbb{A}}\}$, a finite set of observations $\mathbb{O} = \{o_1, o_2, \ldots, o_{n_\mathbb{O}}\}$, transition probabilities $p^a_{ss'}$ which denote the probability to move from state $s$ to $s'$ if action $a$ is taken, probabilities $O^a(s, o)$ for observing $o$ while choosing action $a$ in state $s$, and a reward function $R^a(s) \in \mathbb{R}$ which denotes the reward for performing action $a$ in state $s$. The transition probabilities and the probabilities for the observations sum to one, $\sum_{s' \in \mathbb{S}} p^a_{ss'} = 1$ and $\sum_{o \in \mathbb{O}} O^a(s, o) = 1$. The system evolves due to the transition probabilities $p^a_{ss'}$ and the chosen actions $a$. However, an agent which interacts with the system does not observe the states but observes only

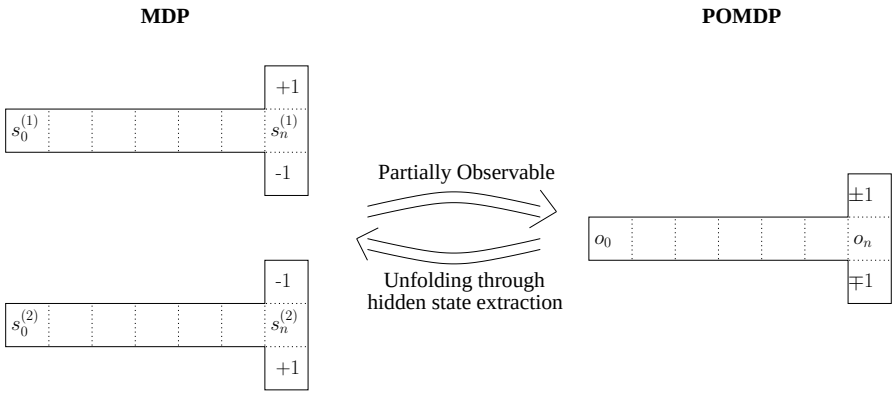


Figure 6.2: The left side of the figure shows a Markov Process (T-Maze). This one is transformed implicitly through partial observability to a partially observable Markov Process ($s_i^{(1)}, s_i^{(2)} \rightarrow o_i$, right side). Our method recovers the original Markov Process through extracting the hidden states.

the observations. Observations are typically ambiguous and the agent does not know the state of system.

The task for the agent is to determine a control *policy* that maximizes the *value*. The value is the expected sum of discounted rewards (infinite horizon), $\mathbb{E}_\pi[\sum_{i=1}^{\infty} \gamma^{i-1} r_i]$, where the discount is denoted with $\gamma \in (0, 1]$, $r_i$ is the received reward in step $i$ and $\pi$ a policy. The policy is a mapping from a sequence of observations and actions to an action, $\pi(o_1, a_1, o_2, a_2, \ldots, o_n) = a_n$. In the case that the state of the system is fully observable it is enough to consider only the last observation. For a "real" POMDP it is needed to consider the past actions and observations. Instead of the full history it is enough to store the *belief vector* [74]. The belief vector encodes the belief about the state of the system, e.g. $b(s)$ is the probability to be in state $s$. The probabilities sum to one, $\sum_{s \in \mathbb{S}} b(s) = 1$.

Standard methods for solving a POMDP use the belief vector representation [74]. A problem with this approach is that the space of belief states is continuous and algorithms are computationally expensive. A natural approach for determining the belief state and to learn the transition model is to apply a *Hidden-Markov Model* (HMM) [62]. The states of the HMM represent in this case the states of the POMDP and the observations of the HMM are the observations of the POMDP. Actions can be incorporated through using the *Input-Output Hidden Markov Model* (IOHMM) [11]. The IOHMM is in principal a HMM where the state transitions depend on an input signal. To model a POMDP one uses the actions as the input. In this way the state transitions of the HMM depend on the chosen actions of the policy. The IOHMM needs to be trained. This can be done by running the POMDP and collecting the input and output sequences. The transition model can then be learned by applying the *Expectation-Maximization* algorithm [9, 26].

A different way to apply an IOHMM, which is particularly suited to hidden state extraction, is to predict the reward sequence given the observations of the

POMDP. The inputs to the IOHMM are in this case the POMDP observations. The basic idea from this approach is that observations which influence the reward must be represented in the state space of the IOHMM to achieve a good performance.

Often a system consists of more or less independent sub-systems and each of these systems evolve by its own dynamics. Such cases can be modeled by using a *Factorial Hidden-Markov Model* [35], respectively a Factorial IOHMM. We make use of this approach in Section 6.4 by introducing factors for hidden states of the system.

### 6.2.1 Policy Iteration and Value Estimators

For our setting we assume that the true state of the system can be inferred from the observation sequence. In this case it is possible to apply standard methods from the MDP setting. In our simulations we used a *policy-iteration* approach [15]. The policy-iteration algorithm alters between a policy evaluation and a policy improvement step. The policy is evaluated with a value estimator and the estimate is used to calculate a new policy. We used two value estimators, the *Temporal Difference Learning* value estimator (TD(0), [77]) and the *Least-Squares Temporal Difference Learning* value estimator (LSTD, [19, 18]).

The TD(0) estimator is defined by the following update equation which is applied after each step

$$V(s) \leftarrow V(s) + \alpha(r + \gamma V(s') - V(s)),$$

where $V(s)$ is the value estimate for state $s$, $\alpha$ is the learning rate and $r$ the observed reward in that step.

For a MDP the LSTD value estimator coincides with a the maximum likelihood value estimate [18]. The value estimate applies the value equation onto estimates of the MDP parameters. The maximum likelihood parameter estimates $\bar{p}^a_{ss'}$ are given by

$$\bar{p}^a_{ss'} := \frac{T^a(s, s')}{N^a(s)},$$

where $T^a(s, s')$ denotes the number of times the transition $s \to s'$ has been taken while performing action $a$ and $N^a(s)$ is the number of times action $a$ has been performed in state $s$. The parameter estimate is simply the observed frequency of the transition.

The parameter estimates define another MDP that converges to the original MDP with growing sample size. In our simulations we applied *dynamic programming* algorithms [15] to calculate the optimal policy for this estimated MDP. The advantage of this approach against TD(0) is that the value estimate is strongly superior and that fewer samples are needed to yield a reasonable policy. Further, there are no free parameters like the learning rate that must be adjusted. The disadvantage is the higher computation time which is $O(|\mathbb{S}|^3)$ (TD(0) has $O(|\mathbb{S}|)$)).

### 6.2.2 Exploration

Exploration is an important aspect of the initial control process. This is especially true for environments with hidden states, because the hidden relation

must be observed in a number of trials to be detectable for a learning system. In our simulations we applied a very simple counter-based exploration strategy from [78]. This strategy chooses the action that has been chosen the fewest times in each state. Despite the simplicity, the exploration strategy typically visits every state in the state space for a number of times and allows a learning system to infer hidden relations.

## 6.3 Hidden State Representation

We are interested in detecting hidden states of the environment. These hidden states are only of interest if they affect the distribution of the reward. The hidden states that affect the reward distribution are important for modeling the reward distribution and a generative model will produce a better fit to the reward distribution if it incorporates the hidden states. We use a IOHMM in our approach and hidden environment states are most naturally represented through hidden states of the IOHMM. The details of the IOHMM that we use are explained in the next section. The input to the IOHMM are the observations and the target sequences are the gained rewards.

### 6.3.1 Expanded MDP

Our goal is to remove the ambiguities induced by the observation of the POMDP with the help of the hidden environment states. The hidden environment states are represented through the IOHMM states and the ambiguities can be removed by expanding the observation vector with the values of the different hidden states. If all hidden states have been extracted then the POMDP is transformed into a MDP and MDP algorithms can be used to solve the control problem.

For a Factorial IOHMM with $M$ factors the new states are given by

$$\tilde{s} := (o, h^{(1)}, \ldots, h^{(M)}),$$

where $o \in \mathbb{O}$ and $h^{(i)}$ is a state of the $i$th factor of the Factorial IOHMM. The size of the new state space is $|\mathbb{O}| \times |\mathbb{S}_1| \times \ldots \times |\mathbb{S}_M|$, where $\mathbb{S}_i$ is the state space of the $i$th factor. In our each factor consists of two states. The state space size is therefore $2^M|\mathbb{O}|$.

The transition probabilities of the IOHMM factors depend not only on the last IOHMM state, but also on the input (the observations from the POMDP). The transition probabilities for the new MDP are therefore given by:

$$\begin{aligned}&\tilde{P}^a((o_i, h_i^{(1)}, \ldots, h_i^{(M)}), (o_j, h_j^{(1)}, \ldots, h_j^{(M)}))\\ &= P^a(o_i, o_j) P(h_i^{(1)}, h_j^{(1)}|o_i) \ldots P(h_i^{(M)}, h_j^{(M)}|o_i),\end{aligned}$$

where $P(\cdot, \cdot|o)$ denotes the transition probability of the IOHMM. The transition probabilities of the observations are induced by the original MDP: $P^a(o_i, o_j) = \sum_{b=1}^{n_{\mathbb{S}}} \sum_{c=1}^{n_{\mathbb{S}}} P(s_b) p^a_{s_b s_c} P^a(s_b, o_i) P^a(s_c, o_j)/P(o_i)$, where $P(s_b)$ denotes the prior probability to be in state $s_b$ and $P(o_i)$ the probability to observe $o_i$.

In this work we use the most probable IOHMM state (Viterbi algorithm [62]) to determine the MDP state. An alternative would be to use the probabilities for being in the different IOHMM states. This would result in a belief vector representation (see [2, ch. 6]). We did not try this approach.

### 6.3.2 Example

Let us consider the example given in Figure 6.2. The environment is a so called T-maze [6]. In one state (typically the first) a sign is shown which gives the spatial position of the reward. In state $n$ the agent must decide whether to move north or south. If the correct action is taken, a reward of $+1$ will be received, otherwise $-1$. The agent must memorize the seen sign for at least $n$ steps to achieve a good performance. The problem can be interpreted as a POMDP with an underlying MDP structure. The original MDP structure consists actually of two T-mazes, each with a fixed sign and a fixed reward distribution (left side of Figure 6.2). Given the full state information, it is easy to learn a good policy. However, due to the partially observability, the situation results in a POMDP (right side). The agent is only able to observe ambiguous information $o_2, ..., o_n$ with $o_1$ being the only unambiguous information (sign up or down). Our goal is to reconstruct the original MDP structure using information like the sign (right-left arrow in Figure 6.2).

Now, let us further assume that a Factorial IOHMM with one factor ($M = 1$) has been successfully trained and assume that the sign is shown in state $k$. The agent would have the following observation vector in the beginning $o_1^{(0)}, ..., o_{k-1}^{(0)}$ (upper index is the IOHMM state). In state $k$ the IOHMM would switch to state 1 or stay in state 0 according to the coding of the sign. Assume that the shown sign is coded as 1, then the agents final observation sequence looks like $o_k^{(1)}, ..., o_n^{(1)}$. The extracted MDP corresponds to a split in two paths (and T-Mazes) at the sign. In this case the MDP structure of Figure 6.2 varies from the extracted one, because no information gives a hint that the first $k-1$ observations correspond to different states. Nevertheless the extracted MDP problem is easy to solve for any MDP learning algorithm, e.g. *value-policy iteration* with a TD(0) value estimator [77].

### 6.3.3 Training

HMMs are typically trained with the EM or an approximate EM algorithm[1]. The EM is a batch and not an online algorithm. Therefore, it is not possible to train the IOHMM in parallel to an reinforcement learning algorithm. We used two different approaches in order to overcome this problem and to incorporate the IOHMM in a reinforcement learning architecture. The first approach was to train the IOHMM ones with randomly sampled trajectories. This method works fine, as long as the random sampling covers the state space and as long as the important signals are included in a reasonable number of trajectories (in simulations the signals were present in a quarter or less of the total number of trajectories). The second approach combines the IOHMM with the MDP based method and performs an iteration between the methods, similar to a value-policy iteration. In this way the IOHMM is trained multiple times and with a sample distribution that is policy dependent. The advantage of this approach is that the IOHMM adapts to the needs of the current control policy. We used

[1] As an alternative to the EM we directly maximized the likelihood in a number of simulations. We did this with a batch algorithm and the performance was comparable to the EM. The maximization can also be done with a gradient ascent and can thus can be performed online. We did not follow this direction, because the EM approach worked very well for our tasks.

the following algorithm in simulations:

**Algorithm 4** IOHMM-Value/Policy Iteration

```
while IOHMM Transition Matrix Changed? do
  for i = 1 to batchsize do
    while Not in terminal state do
      Step 1: Update IOHMM state with current observation (Viterbi)
      Step 2: Choose action for current observation and IOHMM state
      Step 3: Update value estimates
      Step 4: Collect observations and given reward
    end while
  end for
  Step 5: Train IOHMM
end while
```

In our simulations we stopped when no transition probability changed more than a given threshold $\theta$, i.e. if for all inputs $x$ and IOHMM states $i, j$ it holds that $|P_x(i,j) - \tilde{P}_x(i,j)| < \theta$, where $P_x(i,j)$ denotes the new transition probability of the IOHMM and $\tilde{P}_x(i,j)$ the old transition probability. A drawback of the multiple training is that in a factorial HMM the factors can interchange their meaning and the control policy will get invalid. In our simulations we used the old IOHMM solution as the initialization for the new EM procedure. If the new batch of examples is drawn from a similar distribution like the old one, then few learning steps are needed (often 1 or 2). Furthermore, the coding of the states remains typically the same and the value estimates stay valid.

## 6.4 MIOFHMM

We use a special IOHMM in this work, the so called Memory Input Output Factorial HMM (MIOFHMM) from [41]. The MIOFHMM is the HMM analog to the LSTM network. The MIOFHMM is an input output Hidden Markov Model [11] which consists of M different factors [35]. Each factor is equivalent to a two state HMM model with the following transition matrix depending on the input $x$ and on an input and factor dependent parameter $c_x^{(m)} \in [0,1]$:

$$P_x^{(m)} = \begin{pmatrix} c_x^{(m)} & 1 - c_x^{(m)} \\ 0 & 1 \end{pmatrix} \tag{6.1}$$

Each of the factors represents a memory latch with the catch state being state 2. The $c$'s play the role of diffusion factors. If a parameter $c$ is close to 1 then the system remains in state 1, while for $c$ close to 0 the system "diffuses" from state 1 to state 2. The intuition of the memory latch is that the system should adjust the diffusion probabilities in the way that state 2 is only accessed if a critical input information is observed. In this way state 2 memorizes the occurrence of signals. We can increase the memory capacity to $2^M$ by using multiple factors/latches.

The MIOFHMM was proposed as a solution to the "vanishing gradient" problem. However, some care must be taken with the application of the MIOFHMM, because two problems exist that are related to diffusion of state 1 to state 2. We discuss these problems and a simple solution in Appendix 6.A.

### 6.4.1 Model Details

We use a factorial IOHMM with M factors [35]. Each of the factors has the transition probability defined by (6.1). In [41] the output was modeled in analogy to traditional HMMs through multinomial distributions for each possible complex state, hence it was represented through matrices of dimension $D \times 2^M$, where D is the number of output signals. Such a representation has some drawbacks (see [35] for details). In our case it is especially problematic because we do not calculate the EM directly but we approximate it (s. below). The strong coupling due to this matrix representation is unsuitable for the approximation. We therefore use an alternative approach from [46]. In this approach the output distribution is given by a linear combination of the expected values of the hidden states on which a monotonic and differentiable function $h$ is applied. $h$ is chosen according to the output distribution. For example, if the output is binomial then $h$ is the logistic function. The setting is very flexible and allows to model a variety of continuous and discrete distributions. This is also the advantage to approximations suggested in [35] which are, for example, not usable for discrete distributions.

### 6.4.2 Training

The MIOFHMM is trained with the EM algorithm [26]. In our work we use an approximate EM which is suggested in [46]. The approximation is based on the backfitting algorithm for generalized additive models [36]. The approximation updates the different factors sequentially. The first factor is adapted to account as good as possible to the overall distribution. Then the second factor is adapted to account for the remaining uncertainty, and so on. The method is very similar to boosting. For details see [46]. Like in the variational approximations from [35] a direct coupling between the states of different factors does not exist.

### 6.4.3 Variations of the Memory Latch

It is possible to use other structures then the 2-state memory latch. Instead one can use for example multiple catch states. This increases the representation power of one factor. It is further possible to account for a series of events, where the different events have a long time gap in-between. A structure that would be suitable for this case is a chain of states with a final catch state and high probabilities for staying in the current state. The structure of the MIOFHMM can in this sense be adapted to the task at hand.

## 6.5 Experiments

We tested the capability of the MIOFHMM to detect hidden states in the environment and to reconstruct the underlying MDP in two experiments. In the first experiment we used the T-Maze from [6] to test the maximal distance for which our method is able to relate a source to a target signal. The control problem in the T-Maze is quite simple (mainly one decision must be learned). In the second experiment we used the 986-State Maze from [85] to test the capability of the method in a non-trivial control problem.

### 6.5.1 T-Maze

The T-maze problem has been described in Section 6.4. We used the following setup to solve the T-maze problem. We first generated a set of 1000 trajectories with a counter-based exploration strategy from [78] (Section 6.2.2). This exploration strategy takes in state $s$ the action which has been taken the fewest times. For the T-maze, this sampling leads to a trajectory which walks from the sign to the goal states and makes no step back. In fact, the needed number of steps to reach the goal in case of pure random sampling, increases exponentially with the length of the corridor. We then trained the MIOFHMM with the samples. For the T-maze, the MIOFHMM consisted of just one factor, which is enough to code the observed sign. Notice that in this case the approximate EM is equivalent to the exact EM. After the training of the MIOFHMM we applied the TD(0) estimator on the extended observation vector. In this step we used again the counter-based exploration strategy for sampling. In a final step we used a greedy policy on the TD(0) estimates and we verified the performance of the method in 100 greedy runs. In all trials (10/10) the final policy achieved in all runs a reward of +1 for a maximal corridor length of 100 000. In contrast the current best method for this task from [6] can handle a length of about 70 and for a simpler task a corridor length below 1000.

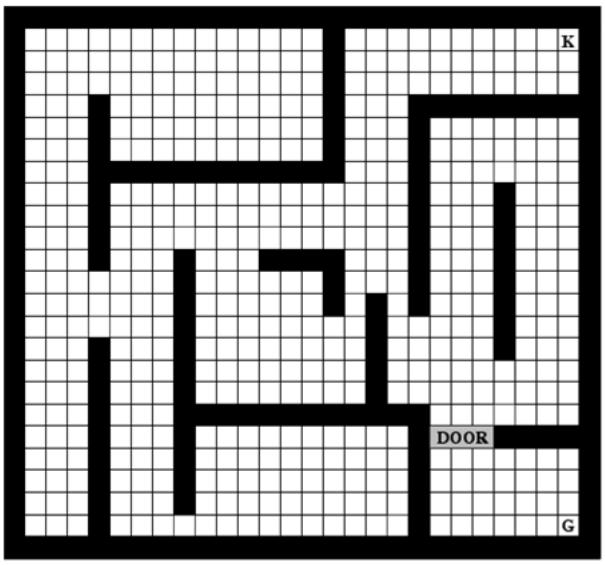


Figure 6.3: 986-State Maze. (G) denotes the goal state and (K) the key. See text for details.

### 6.5.2 986-State Maze.

The second POMDP is a modified version of the so called 986-State Maze from [85]. The POMDP is shown in Figure 6.3. The original setup is the following: The POMDP contains one goal state (G) in the lower right corner. The goal is

blocked by a door, which can only be opened by a key. The key is located in the upper right corner (K).

The difficulty for an agent is to memorize if it visited the key state or not. This is difficult for a MDP algorithm since it must decide for a given state if it should walk to the key or to the door without knowing whether it has the key or not. However, in the original setup a solution for MDP algorithms is possible and one can bypass the memory task. This is the case because one can construct a path from the starting position to the key and a path from the key to the goal which does not cross each other (in this way the states of the path contain the information if the key has been visited or not). The following MDP policy solves the task: first walk from the starting position (for example in the left area of the map) along the left wall until the key room and then from the Key state walk along the right wall to the door and to the goal.

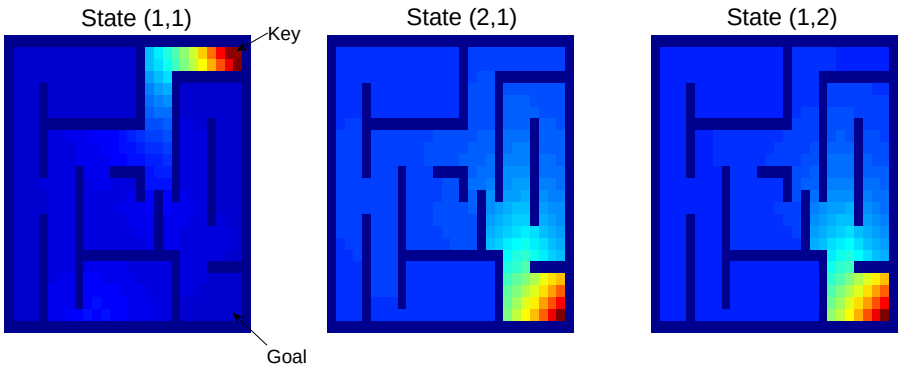


Figure 6.4: The figure shows the value maps which correspond to the different HMM states. The left map corresponds to the case that both factors are in state 1. This case represents the situation that the signal has not yet been observed. The map in the middle corresponds to the case that the first factor is in state 2 and the right map to the case that the second factor is in state 2. Both cases encode the situation that the signal has been observed, whereas one encodes signal one and the other signal two. The color denotes the value (dark blue = low value, red = high value). For state (2,1) and (1,2) the value gets higher from the key to the goal, while for state (1,1) the value is highest at the key.

We changed the original setup to overcome this effect. In our setup the key was associated with a signal (like in the T-Maze). Furthermore, in the goal the agent can take two possible actions, one with positive reward and one with negative reward. The door was removed. The signal at the key state denotes the correct action. This task is not solvable for a MDP algorithm, which will have an equal chance for positive or negative reward. The task is considerably harder than the T-Maze task because the trajectories differ strongly for different runs and the control task is not trivial.

We solved the 986-State Maze problem with a similar setting as for the T-Maze. This time we used a 2-factor MIOFHMM and sampled again with the counter-based exploration strategy. We tested the method for three different cases: (1) deterministic transitions, (2) 1/4 of the action result in a random

movement, (3) 1/2 of the actions result in a random movement. In all cases (10/10) the MIOFHMM extracted the relevant signal, whereas a high number of EM steps was needed (up to 500). Instead of TD(0) we used the LSTD based approach as discussed in Section 6.2.1 for control (5000 trajectories). The LSTD approach solved the problem nearly perfectly. In a few cases it happened that from some states the goal was not reached. Figure 6.4 shows value maps for the different MIOFHMM states (in this case 1/4 of the actions were random). State (1,1) means that both MIOFHMM factors are in state 1. State (2,1) and (1,2) denote the case where one of the factors switched to state 2. The case that both factors switched to state 2 does not occur.

We did not try alternative methods on this task, as a method should at least solve the T-maze for a length of $> 1000$. The only candidate would in principle be the method from [6] but this method achieves such a performance only if few actions must be taken.

## 6.6 Summary

Environments with hidden states are the common case in everyday life. The knowledge of such hidden states is often cruicial for a successful control strategy. Extraction of hidden states is, however, not trivial. An indication for this point is the shortcoming of most reinforcement learning methods in quite simple environments like the T-maze. One reason for the difficulties is the vanishing gradient problem, which states that it is getting exponentially harder in time to extract relations between signals. In a complex control task in which a sequence of actions must be performed to achieve a goal, it is typically the case that a long time gap exists between the first actions and the final result (e.g. the reward). It is needed to account for the vanishing gradient problem to be successful in such scenarios.

In this work we used the MIOFHMM from [41] to deal with the vanishing gradient problem. The MIOFHMM is able to relate far apart signals and is therefore well suited for large environments. In contrast to [41] we did not apply the exact EM but an approximation suggested in [46]. The advantage of our approach is that it is computationally feasible to use multiple factors and in this way to increase the size of the memory for environmental signals. We showed in simulations that in the extreme case our method can reconstruct MDPs if the signals are up to 100 000 steps apart. This is a significant improvement in comparison with other approaches, which are mostly only able to solve POMDPs where the signals are fewer than 100 steps apart. Our method is explicitly constructed for extracting hidden signals and does not handle other sources of partial observability. We think that it is an interesting line for future research to combine our method with approaches that handle other sources like fuzzy sensors.

## 6.A Diffusion and Credit Assignment

We discuss in this appendix two problems of the MIOFHMM which have to do with diffusion from state 1. For simplification we assume in the following that the output must only be generated at the last time step.

The probability to be in state 1 depends exponentially on $t$, due to the diffusion. With $s_t$ being the state at time $t$, $\{x_t\}$ being the input sequence and $k$ being a point in time with $k < t$:

$$P(s_t = 1|s_k = 1, \{x_t\}) = \prod_{i=k}^{t} c_{x_i} \sim c^{t-k} \tag{6.2}$$

The factor of interest in the system is the probability for a transition from state 1 to state 2. The learning method (in our case EM, [26, 9]) must adjust this parameter to be high for important signals and low otherwise. The update rule for this parameter depends on the difference between $\xi_t(1,2)$ and $\xi_t(1,1)$, where $\xi_t(i,j)$ denotes the probability of being in state $i$ at time $t$ and state $j$ at time $t+1$ (we use the notation of [62]). These probabilities are

$$\xi_t(1,1) = (1/Z)\alpha_t(1)a_{11}\beta_{t+1}(1) \sim a_{11}\beta_{t+1}(1),$$
$$\xi_t(1,2) = (1/Z)\alpha_t(1)a_{12}\beta_{t+1}(2) \sim a_{12}\beta_{t+1}(2),$$

where $Z$ is a normalization factor, $\alpha_t(i)$ the probability to be in state $i$ at time $t$ given the observation sequence until time $t$, $\beta_t(i)$ the probability to be in state $i$ given the observations from time $t+1$ to the end of the sequence and $a_{ij}$ the probability for a state transition from state $i$ to $j$. Crucial in these equations are the $\beta$'s. These should be as different as possible to detect important information at time $t$. However, due to the diffusion these are:

$$\beta_{t+1}(1) \approx c^{T-t-1}b_1(o_T) + (1 - c^{T-t-1})b_2(o_T),$$
$$\beta_{t+1}(2) = b_2(o_T),$$

where $o_T$ is the observation at time $T$ and $b_1(o_T), b_2(o_T)$ are the observation probabilities, i.e. the probabilities to observe $o_T$ given state 1, respectively 2. $c^k$ goes exponentially in $k$ to zero and $\beta_{t+1}(1)$ approaches $\beta_{t+1}(2)$. Hence, it gets exponentially more difficult to separate the states in dependence of the time and adjustments will get marginal. This results in an effect similar to the vanishing gradient (Case A in Figure 6.1).

The second weakness becomes obvious when taking a closer look at the probability to be in state 2 at time $T$ if the important signal comes either at time $t$ or time $l$ (for simplification we assume l=0). Due to equation (6.2) the probability is exponentially lower for the later point in time (here $t$): Using

$$P(s_T = 2|\{x_T\}) = \sum_{t=1}^{T-1} P(s_{t+1} = 2|s_t = 1, x_t)P(s_t = 1|\{x_{t-1}\}),$$

assuming that the transition probability for signal $I$ is $d$ and setting $\tilde{c} := 1 - c$:

$$P(s_T = 2|x_0 = I) - P(s_T = 2|x_t = I) = \tilde{c}c^t + d - dc^t - \tilde{c} = (d - \tilde{c}) - (d - \tilde{c})c^t.$$

For $t \to \infty$ the value approaches exponentially the value $(d - \tilde{c})$. Hence the probability to be in state 2 is exponentially less affected by the later signal (e.g. for $d = 0.5$, $\tilde{c} = 0.1$ and $t = 10$ the probability difference is 0.36). The probability directly affects the change in the output distribution. For the multinomial case

$$b_2(k) = \frac{\sum_{i=1}^{n} P(s_T = 2|\{x_{T,i}\})\delta_{o_T,k}}{\sum_{i=1}^{n} P(s_T = 2|\{x_{T,i}\})},$$

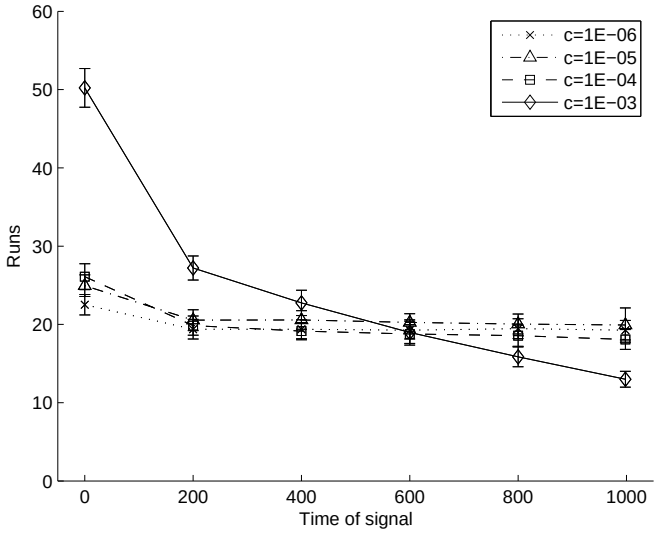


Figure 6.5: The Figure shows the effect of the initialization of the diffusion factor c. If it is to high a vanishing gradient like problem is present (The number of training runs needed to correctly predict the reward for a T-Maze of length 1000 is plotted).

where $\delta$ denotes the Dirac delta function. The output distribution affects the values of the $\beta$'s. Hence, the detection of the signal is inhibited. Therefore, we have a problem that corresponds to case B in Figure 6.1.

In simulations the first disadvantage was dominant, however, both can lead to considerable problems. Yet, being aware of the problems it is easy to circumvent them. Despite the exponential relation of the critical factor $c^t$ to the time $t$ it is possible to control the "minimal" influence through a thorough initialization. If we know, for example, that we have to cope with sequences of maximal length 100 then we can bound this influence factor easily (for example by 0.9): $0.9 < c^t < c^{100} \Rightarrow c > 0.9^{1/100}$. This way the transition probability is extremly low in the beginning, but all signal times are nearly equally important. The low initial value did not result in problems in our simulations. In Figure 6.5 the effects of different initial values of $c$ are plotted. The x-axis denotes the signal time and the y-axis the number of training runs needed to solve a prediction problem for the T-maze ([6], see also Section 6.4) in dependence of the time of the signal. The task was to predict the given reward for the last transition (time $t = 1000$).

# Bibliography


[1] D. Aberdeen and J. Baxter. Scaling internal-state policy-gradient methods for pomdps. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 2002.

[2] D. A. Aberdeen. *Policy-Gradient Algorithms for Partially Observable Markov Decision Processes*. PhD thesis, The Australian National University, 2003.

[3] M. Aigner. *A Course in Enumeration*. Springer, 2006.

[4] Leola A. Alfonso-Reese, F. Gregory Ashby, and David H. Brainard. What makes a categorization task difficult? *Perception & Psychophysics*, 64(4):570–583, 2002.

[5] F. Gregory Ashby and W. Ell Shawn. The neurobiology of human category learning. *TRENDS in Cognitive Sciences*, 5(5):204–210, 2001.

[6] Bram Bakker. Reinforcement learning with long short-term memory. In *NIPS*, 2001.

[7] H. Bauer and R. B. Burckel. *Probability Theory*. Walter de Gruyter, 1995.

[8] H. Bauer and R. B. Burckel. *Measure and Integration Theory*. Walter de Gruyter, 2001.

[9] L. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41, 1970.

[10] Y. Bengio and P. Frasconi. Diffusion of context and credit information in markovian models. *Journal of Artificial Intelligence Research*, 1995.

[11] Y. Bengio and P. Frasconi. An input/output hmm architecture. In *Advances in Neural Information Processing Systems 7*, 1995.

[12] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5:157–166, 1994.

[13] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 1994.

[14] D. P. Bertsekas and S. E. Shreve. *Stochastic Optimal Control (The Discrete Time Case)*. Academic Press, New York, 1978.

[15] D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.

[16] L. E. Bourne. Knowing and using concepts. *Psychological Review*, (77):546–556, 1970.

[17] J. Boyan. *Learning Evaluation Functions for Global Optimization*. PhD thesis, School of Computer Science Carnegie Mellon University, 1998.

[18] J. Boyan. Least-squares temporal difference learning. In *ICML*, 1999.

[19] S. J. Bradtke and A. G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22, 1996.

[20] Alistair Bray and Dominique Martinez. Kernel-based extraction of slow features: Complex cells learn disparity and translation invariance from natural images. *NIPS*, 15:253–260, 2002.

[21] L. Cesató and M. Opper. Sparse on-line gaussian processes. *Neural Computation*, 2002.

[22] L. Chrisman. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *Proceedings of the Tenth International Conference on Artificial Intelligence*, 1992.

[23] L. Clemente, A. Davison, I. Reid, J. Neira, and J. Tardos. Mapping large loops with a single hand-held camera. In *Robotics: Science and Systems Conference*, 2007.

[24] A. J. Davison. *Mobile Robot Navigation Using Active Vision*. PhD thesis, University of Oxford, 1998.

[25] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *IEEE International Conference on Computer Vision*, 2003.

[26] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 1997.

[27] R. A. DeVore and G.G. Lorentz. *Constructive Approximation*. Springer, 1993.

[28] E. Eade and T. Drummond. Scalable monocular slam. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 2006.

[29] Y. Engel. *Algorithms and Representations for Reinforcement Learning*. PhD thesis, Hebrew University, 2005.

[30] Y. Engel, S. Mannor, and R. Meir. Bayes meets bellman: The gaussian process approach to temporal difference learning. In *ICML*, 2003.

[31] Y. Engel, P. Szabo, and D. Volkinshtein. Learning to control an octopus arm with gaussian process temporal difference methods. In *NIPS*, 2005.

[32] C. Estrada, J. Neira, and J.D. Tardos. Hierarchical slam: Real-time accurate mapping of large environments. In *IEEE Transactions on Robotics*, volume 21, 2005.

[33] J. A. Filar, L. C. M. Kallenberg, and H. Lee. Variance-penalized markov decision processes. *Mathematics of Operations Research*, 1989.

[34] M. Franzius, H. Sprekeler, and L. Wiskott. Unsupervised learning of place cells, head-direction cells, and spatial-view cells with slow feature analysis on quasi-natural videos. In preparation, 2007.

[35] Z. Ghahramani and M. I. Jordan. Factorial hidden markov models. *Machine Learning*, 1997.

[36] T.J. Hastie and R.J. Tibshirani. *Generalized additive models*. London: Chapman and Hall, 1990.

[37] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2 edition, 1999.

[38] J. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[39] S. Hochreiter. Untersuchung zu dynamischen neuronalen netzen. Master's thesis, Technische Universität München, Germany, 1991.

[40] S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(2):107–116, 1998.

[41] S. Hochreiter and M. C. Mozer. A discrete probabilistic memory model for discovering dependencies in time. In *Proceedings of the International Conference on Artificial Neural Networks*, 2001.

[42] R. Horst and H. Tuy. *Global Optimization. Deterministic Approaches*. Springer, Berlin, 1998.

[43] Y. Huang and L.C.M. Kallenberg. On finding optimal policies for markov decision chains: A unifying framework for mean-variance-tradeoffs. *Mathematics of Operations Research*, 1994.

[44] Tommi Jaakkola, Michael I. Jordan, and Satinder P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 1994.

[45] D. Jackson. *Ueber die Genauigkeit der Annäherung stetiger Funktionen durch ganze rationale Funktionen gegebenen Grades und trigonometrische Summen gegebener Ordnung*. PhD thesis, Göttingen, 1911.

[46] R. A. Jacobs, W. Jiang, and M. A. Tanner. Factorial hidden markov models and the generalized backfitting algorithm. *Neural Computation*, 2002.

[47] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in paritally observable stochastic domains. *Artificial Intelligence*, 101, 1998.

[48] Steven M. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall International, Inc., 1993.

[49] M. Kearns and S. Singh. Bias-variance error bounds for temporal difference updates. In *Conference on Computational Learning Theory*, 2000.

[50] M.G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journalof Machine Learning Research*, 2003.

[51] E. L. Lehmann and G. Casella. *Theory of Point Estimation*. Springer Texts in Statistics, 1998.

[52] J.D.C. Little. The use of storage water in a hydroelectric system. *Journal of the Operations Research Society of America*, 3(2), 1955.

[53] G. G. Lorentz. *Approximation of Functions*. Holt, Rinehart and Winston, 1966.

[54] G. Lugosi. Concentration-of-measure inequalities - lecture notes, 2006.

[55] S. Mannor, D. Simester, P. Sun, and J. N. Tsitsiklis. Bias and variance approximation in value function estimates. *Management Science*, 53, 2007.

[56] R. A. McCallum. Overcoming incomplete perception with utile distinction memory. In *Proceedings of the Tenth International Conference on Machine Learning*, 1993.

[57] R. A. McCallum. Instance-based utile distinctions for reinforcement learning with hidden state. In *Proceedings of the Twelfth International Conference on Machine Learning*, 1995.

[58] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fastslam: a factored solution to the simultaneous localization and mapping problem. In *Eighteenth national conference on Artificial intelligence*, 2002.

[59] R. Munos. Performance bounds in lp-norm for approximate value iteration. *SIAM Journal on Control and Optimization*, 2007.

[60] K. Murphy. Bayesian map learning in dynamic environments. In *NIPS*, 1999.

[61] R. C. O'Reilly, T. S. Braver, and J. D. Cohen. A biologically based computational model of working memory. In A. Miyake and P. Shah, editors, *Models of Working Memory: Mechanisms of Active Maintenance and Executive Control*, pages 375–411. New York: Cambridge University Press, 1999.

[62] L. R. Rabiner. A tutorial on hidden markov models and selected applications inspeech recognition. *Proceedings of the IEEE*, 1989.

[63] Rhadhakrishna C. Rao. *Linear Statistical Inference and Its Applications*. John Wiley & Sons:London, 1973.

[64] C. E. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. In *NIPS 16*, 2004.

[65] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[66] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[67] R. N. Shepard, C. L. Hovland, and H. M. Jenkins. Learning and memorization of classifications. *Psychological Monographs*, 75(13), 1961.

[68] R. Sim, P. Elinas, M. Griffin, and J. J. Little. Vision-based slam using the rao-blackwellised particle filter. In *IJCAI Workshop on Reasoning with Uncertainty in Robotics*, 2005.

[69] S. Singh and R. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22(1):123–158, 1996.

[70] Satinder Singh and Peter Dayan. Analytical mean squared error curves for temporal difference learning. *Machine Learning*, 32, 1998.

[71] R. Smith, M. Slef, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*. Springer-Verlag, 1990.

[72] M. J. Sobel. The variance of discounted markov decision processes. *J. Appl. Probability*, 19:794–802, 1982.

[73] M.J. Sobel. Mean-variance tradeoffs in an undiscounted mdp. *Operations Research*, 42(1), 1994.

[74] E. J. Sondik. The optimal control of partially observable markov processes over the infinite horizon: discounted case. *Operations Research*, 1978.

[75] Stuart and Ord. *Kendall's Advanced Theory of Statistics*. Edward Arnold, fifth edition, 1991.

[76] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[77] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[78] S. Thrun and K. Müller. Active exploration in dynamic environments. In *Advances in Neural Information Processing Systems*, 1992.

[79] C.H. von Lanzenauer. Optimal claim decisions by policyholders in automobile insurance with merit-rating structures. *Op. Res.*, 1974.

[80] Chris Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8, 1992.

[81] D. Werner. *Funktionalanalysis*. Springer, 2005.

[82] D.J. White. Real applications of markov decision processes. *Interfaces*, 15(6), 1985.

[83] D.J. White. Further real applications of markov decision processes. *Interfaces*, 18(5), 1988.

[84] D.J. White. Survey of applications of markov decision processes. *JORS*, 44(11), 1993.

[85] M. Wiering and J. Schmidhuber. Hq-learning. *Adaptive Behavior*, 6(2):219–246, 1997.

[86] D. Wierstra and M. Wiering. Utile distinction hidden markov models. In *Proceedings of the Twenti-first International Conference on Machine Learning*, 2004.

[87] Laurenz Wiskott. Slow feature analysis: A theoretical analysis of optimal free responses. *Neural Computation*, 15(9):2147–2177, September 2003.

[88] Laurenz Wiskott and Terrence Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, 2002.