Feature Extraction for Inverse Reinforcement Learning

Feature-Extraktion für Inverse Reinforcement Learning
Master-Thesis von Oleg Arenz aus Wiesbaden
Tag der Einreichung:

1. Gutachten:
2. Gutachten:
3. Gutachten:

TECHNISCHE UNIVERSITÄT DARMSTADT

Feature Extraction for Inverse Reinforcement Learning
Feature-Extraktion für Inverse Reinforcement Learning

Vorgelegte Master-Thesis von Oleg Arenz aus Wiesbaden

1. Gutachten:
2. Gutachten:
3. Gutachten:

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 17. Dezember 2014

(Oleg Arenz)

Abstract

Equipping an agent with the ability to infer intentions behind observed behavior is a prerequisite for creating truly autonomous robots. By deducing the purpose of the actions of other agents the robot would be able to react in a sensible way and, furthermore, to imitate the strategy on a high level. Inverse Reinforcement Learning (IRL) can be applied to learn a reward function that is consistent with the observed behavior and is, thus, a step towards this overall goal.
Some strategies can only be modeled properly by underlying a time-dependent reward function. Maximum Causal Entropy Inverse Reinforcement Learning (MaxCausalEnt-IRL) is a method that can be applied to learn such non-stationary functions. However, it depends on gradient-based optimization and its performance can therefore suffer if too many parameters have to be learned. This can be problematic, since the number of parameters increases significantly if separate reward functions are learned for each time step. Furthermore, since only few time steps might be relevant for the observed task, a second challenge of applying IRL for learning non-stationary reward functions consists in properly extracting such sparseness. This thesis investigates how to meet these practical requirements.
A novel approach, IRL-MSD, is developed for that purpose. Unlike some previous IRL methods, Inverse Reinforcement Learning by Matching State Distributions (IRL-MSD) does not aim to match feature counts but instead learns a reward function by matching state distributions. This approach has two interesting properties. Firstly, the features do not have to be defined explicitly but arise naturally from the structure of the observed state distribution. Secondly, it does not require gradient-based optimization.
The experiments show that it converges faster than existing IRL methods and properly recovers the goals of a sparse reward function. Therefore, IRL-MSD suggests itself for future research.

Acknowledgments

I would like to thank Prof. Dr. Jan Peters for founding the Intelligent Autonomous Systems lab at TU Darmstadt. The IAS concentrates a lot of expertise in the field of robot learning from which I could benefit. I also want to thank my supervisors Prof. Dr. Gerhard Neumann and M.Sc. Christian Daniel for the countless hours they invested during our regular meetings. I learned a lot from you during my work on this thesis. I especially want to thank Prof. Dr. Gerhard Neumann, who came up with the optimization problem for IRL-MSD. I am grateful for having gotten the opportunity to work on this method.

Contents

1. Introduction **2**
1.1. Thesis Statement 2
1.2. Motivation for Imitation Learning by Inverse Reinforcement Learning 2
1.3. Motivation for Learning Time-Dependent Reward Functions 3
1.4. Preliminaries 4
1.4.1. Information Theory 4
1.4.2. Markov Decision Process 5
1.4.3. Optimal Control 5
1.5. Outline 6

2. Related Work **7**
2.1. Learning From Demonstrations 7
2.1.1. Perceiving Demonstrations 7
2.1.2. Imitation Learning 8
2.1.3. Inverse Reinforcement Learning 10
2.2. Relative Entropy Policy Search 13

3. Maximum Causal Entropy Inverse Reinforcement Learning **14**
3.1. Insights 15
3.2. Implementation 17
3.2.1. Learning a Non-Stationary Reward Function 17
3.2.2. Towards a Compact Representation Using Radial Basis Functions 18
3.2.3. Ensuring a Positive Semi-Definite Reward Function 18
3.2.4. Adapting the Stepsize Based on the Dual Function 19

4. Inverse Reinforcement Learning by Matching State Distributions **20**
4.1. The Optimization Problem 20
4.2. The Proposed Algorithm 22
4.2.1. Using the Policy KL for Learning Action Costs 22
4.2.2. Using the Policy KL for Regularization 22
4.2.3. Ensuring a Positive Definite Reward Function 23
4.2.4. Disregarding Features by Matching Marginals 23
4.2.5. A Pseudo-code Implementation of IRL-MSD 23

5. Evaluation **25**
5.1. Assuming Known Statistics of the Expert 25
5.1.1. IRL by Matching State Distributions 25
5.1.2. Comparison with Maximum Causal Entropy IRL 27
5.2. Learning Based on Samples 29
5.2.1. IRL by Matching State Distributions 29
5.2.2. Comparison with Maximum Causal Entropy IRL 31
5.3. Discussion 32

6. Future Work 34
6.1. Learning Context Dependent Reward Functions 34
6.2. Dropping the LQG Assumption 34
6.3. Bounding the Policy-KL 34
6.4. Learning State Independent Action Costs 34

7. Conclusion 35

References 37

A. Derivations for Maximum Causal Entropy Inverse Reinforcement Learning 41

B. Derivations for Inverse Reinforcement Learning by Matching State Distributions 44

C. The Difference between $\tilde{V}_t^\pi(\mathbf{s})$ and $V_t^\pi(\mathbf{s})$ 46
C.1. Derivation of $V_t^\pi(\mathbf{s})$ 46
C.2. Derivation of $\tilde{V}_t^\pi(\mathbf{s})$ 47
C.3. Comparison of $V_t^\pi(\mathbf{s})$ and $\tilde{V}_t^\pi(\mathbf{s})$ 48

Figures, Tables and Algorithms

List of Figures

1.1. Motivating Example 3

5.1. First Dimension of Expert State Distribution in Task Space 26
5.2. Expected Reward of IRL-MSD for Known Expert Statistics 26
5.3. First Dimension of MSD State Distribution in Task Space 27
5.4. Comparison of MSD and MaxCausalEnt for Known Statistics 28
5.5. First Dimension of MaxCausalEnt State Distribution in Task Space 29
5.6. Plots of Learned Reward Parameters 30
5.7. Expected Reward of MSD-ACL Based on 7 Samples 31
5.8. Comparison of Maximum Causal Entropy IRL and IRL-MSD Based on 7 Samples 32
5.9. Plots of Learned Reward Parameters Based on 7 Samples 33

List of Tables

5.1. Learned Goal Position for Known Expert Statistics 28
5.2. Expected Reward for MSD-REG Based on 7 Samples 30
5.3. Learned Goal Position Based on 7 Samples 31

List of Algorithms

1. Pseudo-code Implementation of IRL-MSD 24

Abbreviations and Symbols

List of Abbreviations

Notation	Description
DMP	Dynamic Movement Primitive
IRL	Inverse Reinforcement Learning
IRL-MSD	Inverse Reinforcement Learning by Matching State Distributions
KL	Kullback-Leibler divergence
LfD	Learning from Demonstrations
LQG	Linear Quadratic Gaussian
MAP	Maximum-A-Posteriori
MaxCausalEnt-IRL	Maximum Causal Entropy Inverse Reinforcement Learning
MaxEnt-IRL	Maximum Entropy Inverse Reinforcement Learning
MCMC	Markov-Chain-Monte-Carlo
MDP	Markov Decision Process
MSD-ACL	MSD with Action Cost Learning
MSD-REG	MSD with KL-based Regularization
PbD	Programming by Demonstrations
PoWER	Policy Learning by Weighting Exploration with the Returns
ProMP	Probabilistic Movement Primitive
RBF	Radial Basis Function
REPS	Relative Entropy Policy Search
RL	Reinforcement Learning

1 Introduction

Autonomous robots need to adapt their planning based on observations of their environment. Hence, when observing other agents, the autonomous robot should decide how to react on the observed behaviors. For example, if during a collaborative assembly task one agent picks up a screw, a sensible reaction of a second agent might consist in handing over the screwdriver. Especially if the environment is not well-defined, a meaningful reaction to the observed behavior might only be possible by inferring the intentions of the other agents.
Learning the intention behind an observed behavior is also helpful for Imitation Learning. Imitation Learning (Schaal, 1999; Argall et al., 2009) allows to teach an agent how to perform a task by providing demonstrations. By inferring the goals of the demonstrations, the agent learns a task representation that is generalizable and not affected by different embodiments and system dynamics.
IRL (Ng et al., 2000) is a step towards learning the intentions of an observed behavior by learning a reward function from expert demonstrations. Reward functions define task goals by rating actions and states with respect to these goals and hence can represent intention. This representation is suitable for Imitation Learning, because the agent can learn to perform the desired task by applying Reinforcement Learning (RL) (Sutton and Barto, 1998).

1.1 Thesis Statement

The aim of this thesis is to apply Inverse Reinforcement Learning in order to learn a time-dependent reward function. Furthermore, it is investigated how well the sparseness can be recovered, if the true reward function of the expert is sparse in time.

MaxCausalEnt-IRL (Ziebart et al., 2010) is an IRL approach that can properly treat situations where the agent adapted its plan based on information that has been revealed to him during execution. This method therefore suggests itself for learning time-dependent reward functions. MaxCausalEnt-IRL applies gradient descent for learning the parameters of the reward function. Therefore, the speed of convergence can suffer if many parameters have to be optimized.
A novel approach, IRL-MSD, is developed in order to avoid gradient based optimization. This method does not require an explicit definition of the features, but chooses them automatically based on the structure of the target distribution.

1.2 Motivation for Imitation Learning by Inverse Reinforcement Learning

Teaching a task by combining IRL and RL seems laborious because it involves two steps of learning. Hence, it could be argued that a more direct approach should be preferred by (1) learning the policy directly from the demonstrations or (2) by providing the reward function directly. However, both of these approaches have limitations that shall be discussed in the following.
Imitating a movement from demonstration without learning a reward function is usually achieved by learning a parametrized target policy or target trajectory using regression. Learning the policy directly was used by Sammut et al. (2002) to create an autopilot for a flight simulator. However, such approaches can suffer from the correspondence problem, fail if the dynamics change and can not be generalized. Learning target trajectories is more promising and was successfully applied on a humanoid

robot to learn a forehand tennis swing (Ijspeert et al., 2003). Nevertheless, learning a trajectory is still less flexible than learning a reward function, which gets evident when obstacles occur along the desired path.
Reinforcement Learning on a provided reward function has been used with great success, e.g. for learning fast quadrupedal walking (Kohl and Stone, 2004). However, defining an appropriate reward function can be cumbersome and difficult even for experts. For example, when assessing how well a dishwasher has been loaded, several different features have to be taken into account (e.g. for describing the types of the dishes as well as their positions and orientations) with respect to several different subgoals, e.g. efficient usage of space and high expected cleanness. As it is usually not obvious which subgoals are relevant and how they have to be weighted against each other, defining an appropriate reward function requires experience and often involves hand-tuning by trial and error. In contrast to that, demonstrations can often be provided substantially easier and even by non-experts.

1.3 Motivation for Learning Time-Dependent Reward Functions

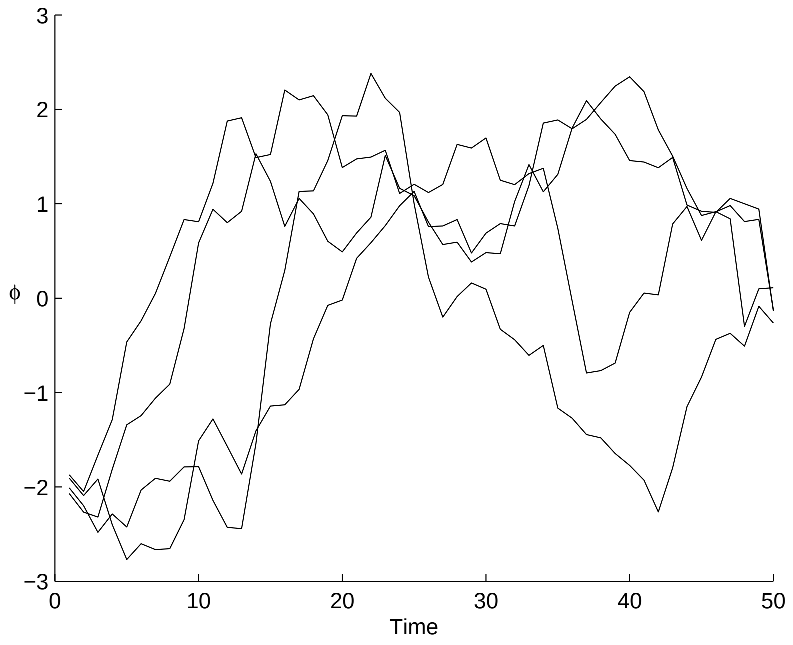

Figure 1.1.: Motivating Example. The demonstrations can be explained with a time-dependent policy based on the only observed feature.

Learning a single, stationary (time-independent) reward function can be inappropriate for learning tasks, that decompose into several subtasks. For example, assume that the demonstrations of an expert led to the trajectories depicted in figure 1.1, where ϕ shall be some arbitrary feature plotted over time. The demonstrated behavior can be explained using a non-stationary reward function, that rewards the agent for achieving $\phi = 1$ at time step 25 and $\phi = 0$ at time step 50. Even if both subtasks served an overall task and could be explained using a stationary reward function, that reward function might possess higher complexity than the non-stationary one and would depend on additional, unobserved features.

1.4 Preliminaries

This section provides background information on Information Theory, Markov Decision Processes and Optimal Control Theory.

1.4.1 Information Theory

Entropy

The entropy (Shannon, 2001) $H(P)$ of a discrete probability distribution $P(X)$ measures the uncertainty of that distribution and, thus, also the amount of encoded information. It is defined as

$$H(X) = -\sum_x P(x)\log(P(x)).$$

This thesis will make use of several related entropy measures for continuous probability distribution that are defined as follows.
The continuous entropy of a probability distribution $p(\mathbf{s})$, denoted $H(\mathbf{S})$, is defined as

$$H(\mathbf{S}) = -\int_{\mathbf{s}} p(\mathbf{s})\log p(\mathbf{s})\, d\mathbf{s}.$$

If an agent chooses its actions according to a time-depend policy, the entropy of that distribution should only take into account the information that has already been revealed to the agent. The entropy is then denoted as causal entropy.

Causal Entropy

For a continuous, conditional distributions $p(\mathbf{a}|\mathbf{s})$, the conditional entropy $H(\mathbf{A}|\mathbf{S})$ is defined as the expected continuous entropy of the conditional distribution, i.e.,

$$H(\mathbf{A}|\mathbf{S}) = -\int_{\mathbf{s}} p(\mathbf{s}) \int_{\mathbf{a}} p(\mathbf{a}|\mathbf{s})\log p(\mathbf{a}|\mathbf{s})\, d\mathbf{a}\, d\mathbf{s}.$$

Let the conditional distribution $p(\mathbf{a}_1,\dots,\mathbf{a}_T|\mathbf{s}_1,\dots,\mathbf{s}_T)$ denote the probability of observing $\mathbf{a}_t$ given $\mathbf{s}_t$ at each discrete time step $t \in [1, T]$. If this distribution only depends on past observations and, thus, marginalizes to $p(\mathbf{a}_1,\dots,\mathbf{a}_T|\mathbf{s}_1,\dots,\mathbf{s}_T) = \prod_{t=1}^{T} p(\mathbf{a}_t|\mathbf{s}_1,\dots,\mathbf{s}_t)$ it will be referred to as "**a** causally conditioned on **s**". The entropy of such a causally conditioned distribution is then denoted as *causal entropy* and defined as

$$H(\mathbf{A}||\mathbf{S}) = -\sum_{t=1}^{T} \int_{\mathbf{s}_1,\dots,\mathbf{s}_t} p(\mathbf{s}_1,\dots,\mathbf{s}_t) \int_{\mathbf{a}_t} p(\mathbf{a}_t|\mathbf{s}_1,\dots,\mathbf{s}_t)\log(p(\mathbf{a}_t|\mathbf{s}_1,\dots,\mathbf{s}_t))\, d\mathbf{a}_t\, d\mathbf{s}_1,\dots,\mathbf{s}_t.$$

For the special case where $p(\mathbf{a}_1,\dots,\mathbf{a}_T|\mathbf{s}_1,\dots,\mathbf{s}_T) = \prod_{t=1}^{T} p(\mathbf{a}_t|\mathbf{s}_t) = \prod_{t=1}^{T} p_t(\mathbf{a}|\mathbf{s})$, the causal entropy simplifies to

$$H(\mathbf{A}||\mathbf{S}) = -\sum_{t=1}^{T} \int_{\mathbf{s},\mathbf{a}} p_t(\mathbf{s})p_t(\mathbf{a}|\mathbf{s})\log(p_t(\mathbf{a}|\mathbf{s}))\, d\mathbf{a}\, d\mathbf{s}.$$

This equation can be used to compute the entropy of first order Markovian policies, i.e. policies that do not depend on past states if the current state is given.

Relative Entropy

The relative entropy $\mathrm{KL}_{\mathbf{s}}(P||Q)$ between two distributions $p(\mathbf{s})$ and $q(\mathbf{s})$, also known as Kullback-Leibler divergence, measures the information loss that results when $p(\mathbf{s})$ is employed as approximation of $q(\mathbf{s})$. It is defined as

$$\mathrm{KL}_{\mathbf{s}}(P||Q) = \int_{\mathbf{s}} p(\mathbf{s}) \frac{\log p(\mathbf{s})}{\log q(\mathbf{s})}\, d\mathbf{s}.$$

The conditional relative entropy $\mathrm{KL}_{\mathbf{a}|\mathbf{s}}(P||Q)$ and the causal relative entropy $\mathrm{KL}_{\mathbf{a}||\mathbf{s}}(P||Q)$ are defined, analogous to the conditional and causal entropy,

$$\mathrm{KL}_{\mathbf{a}|\mathbf{s}}(P||Q) = \int_{\mathbf{s},\mathbf{a}} p(\mathbf{s})p(\mathbf{a}|\mathbf{s}) \frac{\log p(\mathbf{a}|\mathbf{s})}{\log q(\mathbf{a}|\mathbf{s})}\, d\mathbf{s}\, d\mathbf{a},$$

$$\mathrm{KL}_{\mathbf{a}||\mathbf{s}}(P||Q) = \sum_{t=1}^{T} \int_{\mathbf{s},\mathbf{a}} p_t(\mathbf{s})p_t(\mathbf{a}|\mathbf{s}) \frac{\log p_t(\mathbf{a}|\mathbf{s})}{\log q_t(\mathbf{a}|\mathbf{s})}\, d\mathbf{s}\, d\mathbf{a}. \tag{1.1a}$$

Equation 1.1a can be used as measure of similarity between two time-dependent policies while properly taking into account causality.

1.4.2 Markov Decision Process

A Markov Decision Process (MDP) is a mathematical framework that is often employed for modeling the environment of an agent in the context of Reinforcement Learning. MDPs model time by using discrete time steps. Within this thesis, only finite horizon MDPs are discussed, i.e. it is assumed that the process always ends after a fixed number T of time steps. The state of the environment and the state of the agent are modeled together using the set $\mathcal{S}$ of states $\mathbf{s}$. At each time step, the agent chooses an action $\mathbf{a}$ from the set $\mathcal{A}$ of actions. After an action has been chosen, the process transitions into the next time step by randomly choosing a new state according to the system dynamics $p_t(\mathbf{s}'|\mathbf{s},\mathbf{a})$. A MDP assumes the Markovian property, i.e. it assumes that the probability distribution over the next state $\mathbf{s}'$ does not depend on past states or actions, if the current state $\mathbf{s}$ and the current action $\mathbf{a}$ are known. Additionally, for any given time step t, the state $\mathbf{s}_t$ and action $\mathbf{a}_t$ are rated using a reward function $r_t(\mathbf{s},\mathbf{a})$. Therefore, a finite horizon MDP can be defined using a five-tuple $\mathcal{M} = \big(\mathcal{S}, \mathcal{A}, p_t(\mathbf{s}'|\mathbf{s},\mathbf{a}), T, r_t(\mathbf{s},\mathbf{a})\big)$.
For the remainder of this thesis, the states and actions are represented using vectors $\mathbf{s}$ and $\mathbf{a}$ of size N_s and N_a respectively. The state and action for time step t are denoted by $\mathbf{s}_t$ and $\mathbf{a}_t$ and their continuous element at index i by $s_{i,t}$ and $a_{i,t}$ respectively. The system dynamics are assumed to be linear with Gaussian noise, e.g.

$$p_t(\mathbf{s}'|\mathbf{s},\mathbf{a}) = \mathcal{N}(\mathbf{A}_t\mathbf{s} + \mathbf{B}_t\mathbf{a} + \mathbf{b}_t, \mathbf{\Sigma}_{dyn,t}).$$

Furthermore, the reward function is assumed to be convex quadratic in $\mathbf{s}$ and $\mathbf{a}$. Thereby, the MDP complies with the Linear Quadratic Gaussian (LQG) assumption.

1.4.3 Optimal Control

An agent, acting in a MDP, chooses its actions according to a policy $\pi_t(\mathbf{a}|\mathbf{s})$ that is assumed to be consistent over time and thereby (and due to the Markovian property) depends only on the current state and time step. Reinforcement Learning addresses the problem of learning a policy that maximizes the expected reward of the agent. For finite horizon MDPs that comply with the LQG assumption, the optimal

control policy can be computed recursively, even for continuous (and thereby infinite) state and action spaces, by using backward induction.
Backward induction makes use of the concept of Value functions and state-action Value functions. The Value function for a given policy π at time step t is denoted by $V_t^{\pi}(\mathbf{s})$ and corresponds to the expected reward of an agent that starts at time step t in state $\mathbf{s}$, assuming that the agent will act according to policy π for the current and all remaining time steps. The state-action Value function $Q_t^{\pi}(\mathbf{s},\mathbf{a})$ is defined accordingly, however, it assumes that the agent chooses action $\mathbf{a}$ at time step t and only afterwards acts according to the policy π. The Value function for the last time step does not have to take future rewards into account and is thereby given by the reward function of the last time step, $V_T^{\pi}(\mathbf{s}) = r_T(\mathbf{s})$. The reward at the last time step does not depend on an action, because it is assumed that the MDP ends immediately after reaching the final state. The state action Value function is consequently only defined for $t < T$. It can be expressed in terms of the expected Value function of the next time step using

$$Q_t^{\pi}(\mathbf{s},\mathbf{a}) = r_t(\mathbf{s},\mathbf{a}) + \mathbb{E}_{p_t(\mathbf{s}'|\mathbf{s},\mathbf{a})}\left[V_{t+1}^{\pi}(\mathbf{s}')\right], \tag{1.2}$$

where the distribution over the next state $\mathbf{s}'$ is given by the system dynamics $p_t(\mathbf{s}'|\mathbf{s},\mathbf{a})$.
Similarly, the Value function for a time step $t < T$ can be expressed in terms of the expected state action Value function of that same time step using

$$V_t^{\pi}(\mathbf{s}) = \mathbb{E}_{\pi_t(\mathbf{a}|\mathbf{s})}\left[Q_t^{\pi}(\mathbf{s},\mathbf{a})\right], \tag{1.3}$$

where the distribution of the current action is given by the policy $\pi_t(\mathbf{a}|\mathbf{s})$. These recursive equations (1.2 and 1.3) are known as the Bellman equations for finite horizon MDPs. Starting with $V_T^{\pi}(\mathbf{s})$ they can be employed to compute the Value function and the state-action Value function of any given policy π for all time steps t by deducing backwards in time.
The optimal policy $\pi_t^{\star}(\mathbf{a}|\mathbf{s})$ chooses at each time step t an action that maximizes the state-action Value function $Q_t^{\star\pi}(\mathbf{s},\mathbf{a})$. Hence, the corresponding state-action Value function $Q_t^{\star\pi}(\mathbf{s},\mathbf{a})$ and the Value function $V_t^{\star\pi}(\mathbf{s})$ can be computed using equation 1.2 and

$$V_t^{\star\pi}(\mathbf{s}) = \max_{\mathbf{a}} \; Q_t^{\star\pi}(\mathbf{s},\mathbf{a}).$$

The recursive equations are then known as Bellman optimality equations for finite horizon MDPs.
For LQG systems, $V_t^{\pi}(\mathbf{s})$ and $Q_t^{\pi}(\mathbf{s},\mathbf{a})$ are convex quadratic for all time steps. The involved expected values and maximum values can thus be derived analytically. The optimal policy for such systems takes the form of a deterministic, linear controller $\pi_t(\mathbf{s}) = \mathbf{K}_t\mathbf{s} + \mathbf{k}_t$, where the controller gains $\mathbf{K}_t$ and $\mathbf{k}_t$ can be computed using backward induction.

1.5 Outline

The remainder of this thesis is structured as follows: Chapter 2 discusses related work, by providing an overview of Imitation Learning and Inverse Reinforcement Learning. Studying MaxCausalEnt-IRL led to insights that hopefully assist in better understanding the mechanics of the approach. These insights are presented in chapter 3 along with the discussion of implementation specific details. Chapter 4 will cover the novel approach of IRL-MSD. Chapter 5 evaluates this approach and compares it to MaxCausalEnt-IRL. Finally, Chapter 6 presents an outlook on future research and Chapter 7 recapitulates the findings of this thesis.

2 Related Work

2.1 Learning From Demonstrations

This section provides a brief overview of Learning from Demonstrations (LfD) in the field of robotics. Robotic LfD is an active field of research since it was first covered in depth in 1984 (Halbert, 1984). Since then it has evolved into several different directions and produced overlapping terminology (Argall et al., 2009). This thesis distinguishes between the following terms:

- *Learning from Demonstrations* is used for all robot learning methods that make use of demonstrations. This definition is strictly more general than the one given by Argall et al. (2009), that only applies to approaches that learn a policy based on demonstrations.
- *Programming by Demonstrations* (PbD) is used in its original meaning, when demonstrations are used to produce code in a programming language (Cypher and Halbert, 1993).
- *Imitation Learning* is used when the robot should learn to imitate the demonstrator. *True Imitation Learning* is used when, following the definition of Tomasello et al. (1993), an "understanding of the intentional state underlying the behavior" is additionally required.
- *Behavioral Cloning* is used when the agent learns state-action mappings that closely match the observed ones without regarding the implications of the actions. It is thereby a subfield of Imitation Learning but can not provide true imitation.

A natural question regarding LfD is how to demonstrate. The different ways of demonstrating a task can be also described from the perspective of the learning agent, leading to the question: How can a robot perceive demonstrations?

2.1.1 Perceiving Demonstrations

Within this thesis, three different possibilities of perceiving demonstrations are discussed. For the first one, the agent perceives the demonstrations merely via exteroceptive sensors, i.e. sensors that measure quantities concerning the agent's environment, e.g. via vision. The second one also incorporates proprioception, i.e. directly sensing quantities that are related to the robot itself, e.g. joint positions. The third possibility of perceiving a demonstration includes proprioception and exteroception and furthermore recordings of the issued control commands.

Perception Based on Exteroceptive Sensing

When perceiving demonstrations based on exteroception, the agent passively observes the expert performing a task. Even though the robot might be able to measure using its proprioceptive sensors, this information is not considered to be part of the demonstration. Kuniyoshi et al. (1994) used *learning by watching* to learn a high-level assembly plan by watching human demonstrations. Marker-based motion capturing is often used by humanoid robots to mimic the whole-body movement of human demonstrations (Ude et al., 2004; Kulić et al., 2011; Kim et al., 2009).
Such demonstrations have the advantage, that they often can be performed naturally and without the

need of interacting with the robot. However they have the drawback, that they give the least information to the agent. Especially, they do not contain any information about how the robot is supposed to perform the task but only about how the expert performs it. If the expert and the robot have similar embodiments, the robot might be able to perform the desired motion by imitating the expert. However, if they have different embodiments, it is not clear how to imitate the expert which is known as the correspondence problem (Nehaniv and Dautenhahn, 2002; Alissandrakis et al., 2002).

Additional Perception Based on Proprioceptive Sensing

The robot can also perceive demonstrations using its proprioceptive sensors if the expert demonstrates the task by physically moving its limbs. Providing such demonstrations to the robot is known as kinesthetic teaching. If the robot is lightweight and compliant, the desired robot motion can be induced solely by forces created by the expert. Such demonstrations have been used successfully for learning a large variety of tasks, including Ball-Paddling, Ball-in-a-Cup, Pendulum Swing-Ups and Pancake-Flipping (Kober and Peters, 2009; Kormushev et al., 2010).

Additional Perception of the Controls

Demonstrations can also be performed by using the actuators of the robot directly, for example via teleoperation. Thereby, the robot is controlled from distance via a remote control, e.g. a joystick or exoskeleton. Other examples for this kind of demonstrations include programming the desired motor commands and some forms of active kinesthetic teaching, i.e. kinesthetic teaching where the robot senses touches of the expert and reacts by issuing corresponding control commands.
Demonstrations on a radio-controlled helicopter where used by Abbeel et al. (2010) to learn a controller that was capable of producing maneuvers at the edge of the physical feasible, including in-place flips and auto-rotational landing (emergency landing with an unpowered main rotor).

2.1.2 Imitation Learning

Imitation Learning is an important application of LfD, as it makes programming the robot easier for the expert and accessible for non-expert users. Imitation Learning can be divided into three different approaches (Schaal, 1999), (1) approaches that are only concerned with matching the demonstrated policy, (2) approaches that try to reproduce the demonstrated trajectory and (3) approaches that use the demonstrated trajectories in order to reduce the search space for Reinforcement Learning.

Policy-Based Imitation Learning

Already in the early years of robotics, demonstrations were used to ease the task of robot programming. The manipulator was guided by a human expert in order to perform a desired movement and was able to repeat that motion by recording the via points. This simple approach of robot programming by demonstration does not incorporate sensory inputs and is thereby only applicable to few, industrial tasks like painting component parts (Lozano-Perez, 1983).
Behavioral Cloning (Bain and Sammut, 1995; Pomerleau, 1989) additionally records the sensory information to map the observed states to actions. This approach was used to create an autopilot for a flight simulator that exceeded the performance of the expert by smoothing out the human control noise (Sammut et al., 2002). This clean-up effect is often encountered with behavioral cloning (Michie et al., 1990). However, as behavioral cloning merely tries to produce the same actions as the expert, mindless of the resulting trajectory, it is often fragile with respect to changes in the environment.

Trajectory-Based Imitation Learning

The policy-based imitation learning approach can fail, if the robot and the expert differ in their kinematics or dynamics, because the same actions then lead to different trajectories. Trajectory-based imitation learning methods circumvent this shortcoming by trying to match the expert's trajectory instead of its policy. In order to make learning of the trajectory feasible, a representation is required that can generate the desired trajectories while at the same time depending on a tractable number of parameters. The choice of representation is crucial for the performance of the imitation learning task. An example for such a representation are via points. Miyamoto et al. (1996) extract via points from demonstrations by first adding a via point at the goal position and then iteratively adding via points at the positions of maximal squared error between the learned trajectory and the demonstrated one until the demonstration is matched sufficiently well. However, this representation can not be generalized straightforwardly and might not recover from perturbations in the environment.

Schaal et al. (2000) proposed to represent a movement by using a nonlinear dynamic system as an attractor. This led to the development of Dynamic Movement Primitives (DMPs), a flexible representation of a motion that can be learned from demonstrations and is therefore suitable for Imitation Learning (Ijspeert et al., 2002, 2003; Schaal et al., 2003, 2004). A DMP is actually not a trajectory representation, but a representation of a control policy. This policy corresponds to a PD-controller that is perturbed with a nonlinear forcing function. The PD-controller has the purpose of reaching a given goal position whereas the forcing function controls which path should be chosen to reach that goal. The forcing function is defined as a radial basis function network and converges to zero so that it does not prevent the PD-controller from reaching the goal state (when assuming appropriate controller gains). Furthermore, a phase variable is introduced that can be used to change the speed of execution. Additional advantages include robustness against perturbations, capability of producing rhythmic movements, and ease of learning, which is achieved by learning the weights of the radial basis functions. Ijspeert et al. (2002) demonstrated the effectiveness of their approach by learning a forehand tennis swing from demonstration.

The trajectories can also be represented as Gaussian distributions by using Probabilistic Movement Primitives (ProMPs) (Paraschos et al., 2013). This approach uses weighted radial basis functions in order to encode the means for each time step and encodes the covariance matrices by using a prior on these weights. Hence, by marginalizing out the weight vector, the trajectories are given by the parameterization of the prior. Similar to DMPs, temporal modulation is achieved by introducing a phase variable and rhythmic motions can be produced by using von-Mises basis functions instead of Gaussian ones. ProMPs can be used for Imitation Learning, by learning the parameters of the prior that maximize the likelihood of the demonstrated trajectories. This probabilistic approach has several interesting capabilities: The movement primitive can be adapted to reach different via points, by conditioning the distribution accordingly. Also, several primitives can be co-activated and blended by computing their product while weighting them with time-dependent activation functions. ProMPs were recently applied for learning interactions between multiple agents in a collaborative assembly task (Maeda et al., 2014).

Reinforcement Learning From Demonstration

For many practical applications, the state space is large and the dynamic model has to be approximated. If the dynamics are not learned, Imitation Learning can be insufficient to fulfill the task. Reinforcement Learning can be applied to learn the task by trial-and-error while at the same time learning the dynamic model. However, exploring the large state space can be too costly, especially if the reward function is sparse. Learning can be boosted by first learning a policy from demonstrations and afterwards improving it using Reinforcement Learning (Schaal, 1997). Following up Imitation Learning with Reinforcement Learning, may not only enable the robot to accomplish the task but also to improve upon the expert demonstration (Atkeson and Schaal, 1997). Furthermore, by applying Reinforcement Learning the robot is able to handle new situations for which the demonstrated behavior would fail, for example, when new obstacles occur (Guenter et al., 2007). Kober and Peters (2009) used Imitation Learning based on DMPs in order to initialize their policy search method Policy Learning by Weighting Exploration with the Returns (PoWER). This approach enabled the robot to reliably succeed in the game of Ball-in-a-Cup. A similar approach was used by Kormushev et al. (2010) for learning the task of pancake-flipping.

2.1.3 Inverse Reinforcement Learning

Inverse Reinforcement Learning constitutes a different application of Learning from Demonstration by aiming at learning the reward function of the expert based on its demonstration. By using the learned reward function for Reinforcement Learning, IRL can be used as a method of Imitation Learning. However, the IRL-problem should be clearly distinguished from the problem of Imitation Learning, because learning a reward function can also serve other purposes than imitation. For example, in human-robot collaboration tasks, the robot might try to infer the intention of the human, not in order to overtake his task but in order to assist him in achieving it. In a less cooperative way, as mentioned by Ramachandran and Amir (2007), the reward function can be used to model the opponent in adversarial games like poker, in order to exploit its strategy. More generally, IRL aims at inferring the goals underlying the observed behavior.
Nevertheless, surveying current research in IRL, one may conclude that it is indeed most often used for Imitation Learning tasks, both, for experimental evaluations as well as for practical applications. This is not surprising, because describing a (nearly) optimal behavior in terms of its intention is succinct and allows for generalization. Furthermore, unlike a policy, the reward function remains valid, if the dynamics change.
In the following, an overview will be given of research in the field of Inverse Reinforcement Learning, starting with the first MDP formulation of the problem, given by Ng et al. (2000).

The MDP Formulation of Inverse Reinforcement Learning

Similar to Reinforcement Learning, Inverse Reinforcement Learning assumes that the agent is acting in a Markov Decision Process. However, the reward function of that MDP is not known to the agent. In exchange, it is given demonstrations of an expert, that tries to maximize this unknown reward. The goal of IRL is now to learn that reward function from the demonstrations. Hence, instead of addressing the problem of finding a near-optimal policy for a given reward function, it addresses the problem of recovering a reward function from demonstrations of a near-optimal policy.
Unfortunately, this problem formulation is ill-posed, because the demonstrations do not suffice to deduce the underlying intention of the expert with certainty. For example, there is always the, usually extremely unlikely but always non-zero, possibility that the expert chooses its actions completely randomly. This would correspond to a constant reward function, for which any behavior would be optimal. Clearly, such degenerated solutions should be discarded and a more reasonable assumption of the underlying reward function should be found. However, what would make an assumption *reasonable*?
Demanding that the expert performs (near) optimal on the learned reward function is a necessary but not sufficient condition. Ng et al. (2000) proposed to further constrain the solution space by demanding that any single-step deviation from the observed policy should be maximally punished. Additionally, an L1-regularization was added to prefer simple solutions. They demonstrated that this approach can be used to approximately recover the true expert policy on a simple grid world task and the mountain-car problem.

Bayesian Inverse Reinforcement Learning

Ramachandran and Amir (2007) propose a Bayesian approach in order to avoid having to decide for a particular reward function. For that purpose the posterior is computed based on a prior on the reward parameters and the likelihood of the demonstrations. Possible priors on the reward parameters include the Gaussian-, Laplace- and Beta-distribution as well as the uniform distribution over a finite interval. The likelihood of a demonstration for given reward parameters is computed under the assumption that the expert chooses its action proportional to the exponential of the corresponding expected reward. The reward function can be inferred from the posterior distribution by computing its mean or Maximum-A-Posteriori (MAP). More interestingly for Imitation Learning, the policy can also be inferred from the posterior directly. However, due to the complexity of the posterior distribution, Markov-Chain-Monte-Carlo (MCMC) (Gilks, 2005) has to be applied for both cases whereby the approach can suffer from the curse of dimensionality (Bellman, 1957).

Matching Feature Counts

Abbeel and Ng (2004) proposed a new approach to discard degenerated solutions, by demanding that the optimal policy with respect to the learned reward function should match the observed policy in behavior. For that purpose, the reward function is assumed to be a linear combination of given time-dependent features $\boldsymbol{\phi}_t(\mathbf{s},\mathbf{a})$, i.e.,

$$r_t(\mathbf{s},\mathbf{a}) = \boldsymbol{\phi}_t(\mathbf{s},\mathbf{a})^\top \boldsymbol{\theta},$$

and a reward function is learned, such that the expected feature counts $\tilde{\boldsymbol{\phi}}$ match the empirical feature counts of the expert $\hat{\boldsymbol{\phi}}$,

$$\frac{1}{N_D}\sum_{i=1}^{N_D}\sum_{t=1}^{T}\boldsymbol{\phi}_t(\mathbf{s}_t^{(i)},\mathbf{a}_t^{(i)}) = \hat{\boldsymbol{\phi}} \overset{!}{=} \tilde{\boldsymbol{\phi}} = \sum_{t=1}^{T}\mathbb{E}_{p_t(\mathbf{s},\mathbf{a})}\left[\boldsymbol{\phi}_t(\mathbf{s}_t,\mathbf{a}_t)\right],$$

where N_D denotes the number of demonstrations and $\mathbf{s}_t^{(i)}$ and $\mathbf{a}_t^{(i)}$ denote the state and action of the i-th demonstration at time step t. The expected feature counts $\tilde{\boldsymbol{\phi}}$ are computed based on the joint distributions $p_t(\mathbf{s},\mathbf{a})$ that would result from the optimal policy with respect to the learned parameters. Since the total reward $r(\mathbf{s},\mathbf{a})$ can be written

$$\begin{aligned} r(\mathbf{s},\mathbf{a}) &= \sum_{t=1}^{T} r_t(\mathbf{s},\mathbf{a}) \\ &= \sum_{t=1}^{T}\boldsymbol{\phi}_t(\mathbf{s},\mathbf{a})^\top\boldsymbol{\theta} \\ &= \left[\sum_{t=1}^{T}\boldsymbol{\phi}_t(\mathbf{s},\mathbf{a})\right]^\top \boldsymbol{\theta}, \end{aligned}$$

a policy that produces the same feature counts as the expert would achieve the same reward on the true reward function independent of its parameters $\boldsymbol{\theta}$.
However, if the joint distribution $p_t(\mathbf{s},\mathbf{a})$ for computing $\tilde{\boldsymbol{\phi}}$ is based on an optimal, deterministic policy, it is often not possible to match the empirical feature counts, because they are usually based on samples and, furthermore, on suboptimal demonstrations. Therefore, IRL approaches that are based on matching feature counts use a stochastic, suboptimal policy for computing the feature expectations. However, when the expected feature counts are based on a policy, that is suboptimal with respect to the learned

parameters $\boldsymbol{\theta}$, different reward functions could be learned, depending on which policy has been chosen. And even though all these reward functions could be used to match the empirical feature counts by employing their respective policy, they might not infer the correct goals and the *optimal* policies for these reward functions thus might perform badly.

Maximum Entropy Inverse Reinforcement Learning

Maximum Entropy Inverse Reinforcement Learning (Ziebart et al., 2008) applies the principle of maximum entropy (Jaynes, 1957) by choosing the policy that leads to the maximum entropy joint distribution for matching the feature counts. This is the most principled approach, as it does not presume any ungrounded constraint on the joint distribution and thereby minimizes the worst-case prediction log-loss (Grünwald and Dawid, 2004).
Under the maximum entropy model, the likelihood of a given path ζ_i is proportional to the exponential of its reward, i.e.,

$$p(\zeta_i|\boldsymbol{\theta}) \propto \exp \boldsymbol{\theta}^\top \tilde{\boldsymbol{\phi}}_{\zeta,i}.$$

The reward function $\boldsymbol{\theta}^\star$ that leads to the maximum entropy state distribution matching the feature counts, can be found by maximizing the log-likelihood of the demonstrations,

$$\boldsymbol{\theta}^\star = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, L(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{i=1}^{N_d} \log p(\zeta_i|\boldsymbol{\theta}),$$

where N_d denotes the number of demonstrations. The maximum likelihood can be found using gradient descent, where the gradient is given by the difference between the empirical feature count and the expected feature count for the current estimate of the reward function, i.e.,

$$\nabla L(\boldsymbol{\theta}) = \hat{\boldsymbol{\phi}} - \tilde{\boldsymbol{\phi}}.$$

Several recent advances in Inverse Reinforcement Learning are based on Maximum Entropy Inverse Reinforcement Learning (MaxEnt-IRL), e.g.

- Levine et al. (2011) apply Gaussian Process Regression in order to learn nonlinear reward functions.

- Levine and Koltun (2012) apply a Laplace approximation in order to locally approximate the policy by a Gaussian along the demonstrated trajectories in a deterministic MDP.

- Boularias et al. (2011) actually use the *relative* entropy between the empirical distribution and expected distribution. By estimating the subgradient via Importance Sampling they derive a model-free method.

In its original formulation, MaxEnt-IRL assumes that all side information was available to the expert at the beginning of its demonstration. If this is not the case, e.g. whenever the system dynamics are noisy, Maximum Causal Entropy Inverse Reinforcement Learning (Ziebart et al., 2010) can be used instead, which chooses the stochastic policy that has the maximum causal entropy while matching the feature counts. This modification is of main interest for this thesis and is discussed in detail in Chapter 3.

2.2 Relative Entropy Policy Search

Policy Search is an approach to Reinforcement Learning that does not aim for learning the Value function but instead learns a policy directly. Policy gradient methods achieve this by using gradient-based optimization to iteratively update the parameters of the policy. However, because the previous experience is encoded within the current policy, policy updates destroy parts of that experience leading to an information loss. Relative Entropy Policy Search (Peters et al., 2010) is based on the optimization problem of maximizing the expected reward (Equation 2.2a) while bounding the information loss (relative entropy) between the current and previous iteration (Equation 2.2b), i.e.,

$$\underset{\pi(\mathbf{a}|\mathbf{s}),\mu(\mathbf{s})}{\text{maximize}} \quad \sum_{\mathbf{s},\mathbf{a}} \mu(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})r(\mathbf{s},\mathbf{a}) \tag{2.2a}$$

$$\text{subject to} \quad \sum_{\mathbf{s},\mathbf{a}} \mu(\mathbf{s})\pi(\mathbf{a}|\mathbf{s}) \log \frac{\mu(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})}{q(\mathbf{s},\mathbf{a})} \leq \epsilon, \tag{2.2b}$$

$$\sum_{\mathbf{s}'} \mu(\mathbf{s}')\boldsymbol{\phi}(\mathbf{s}') = \sum_{\mathbf{s},\mathbf{a}} \mu(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})p(\mathbf{s}'|\mathbf{a},\mathbf{s})\boldsymbol{\phi}(\mathbf{s}'), \tag{2.2c}$$

$$\sum_{\mathbf{s},\mathbf{a}} \mu(\mathbf{s})\pi(\mathbf{a}|\mathbf{s}) = 1, \tag{2.2d}$$

where $\mu(\mathbf{s})$ is the stationary (Equation 2.2c) state distribution that is eventually reached when executing the policy $\pi(\mathbf{a}|\mathbf{s})$ in an infinite horizon MDP. Equation 2.2d is necessary to ensure that $\pi(\mathbf{a}|\mathbf{s})$ and $\mu(\mathbf{s})$ are probability distributions. $q(\mathbf{s},\mathbf{a})$ represents the (estimated) joint distribution of the previous iteration, thus, the policy that maximizes the problem,

$$\pi_{\max}(\mathbf{a}|\mathbf{s}) \propto q(\mathbf{s},\mathbf{a}) \exp\left(\frac{1}{\eta}\left(r(\mathbf{s},\mathbf{a}) + \sum_{\mathbf{s}'} p(\mathbf{s}'|\mathbf{s},\mathbf{a})\boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s}') - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s})\right)\right),$$

corresponds to the best policy update with bounded information loss ϵ. The Lagrangian Multipliers $\boldsymbol{\theta}$ and η can be found using gradient based optimization.
Encoding the iterative nature of a learning approach by bounding the information loss between the current and last iteration is a major inspiration for IRL-MSD.

3 Maximum Causal Entropy Inverse Reinforcement Learning

The action choices $\mathbf{a}_t$ of the expert for different time steps t might not be consistent with respect to the complete trajectory, because it might have adapted its planning based on the outcomes of the stochastic system dynamics $p_t(\mathbf{s}'|\mathbf{s},\mathbf{a})$. Maximizing the entropy of the whole trajectory distribution does not take this causality into account.
Maximum *Causal* Entropy IRL differs from Maximum Entropy IRL by aiming at maximizing the causal entropy of the policy $\pi_t(\mathbf{s},\mathbf{a})$ instead of the entropy of the joint distribution $p(\mathbf{s},\mathbf{a})$.
The resulting Gaussian policy chooses actions with a probability that increases exponentially with the expected future reward,

$$\pi_t^{\mathrm{MCE}}(\mathbf{a}|\mathbf{s}) \propto \exp\left(Q_t^{\pi}(\mathbf{s},\mathbf{a})\right). \tag{3.1}$$

For LQG systems, this leads to a noisy linear controller, that differs from the optimal controller only by its noise.
The state-action Value function $Q^{\pi}(\mathbf{s},\mathbf{a})$ and the Value function $V_t^{\pi}(\mathbf{s})$ can be computed using the equations 1.2 and 1.3 described in section 1.4.3. However, Ziebart et al. (2010) provide a different way to compute a Value function $\tilde{V}_t^{\pi}(\mathbf{s})$, that is similar to the computation of the optimal Value function (Equation 1.4.3) except that the maximum-operator has been replaced by the softmax-operator,

$$\begin{aligned}\tilde{V}_t^{\pi}(\mathbf{s}) &= \log \int_{\mathbf{a}} \exp\left(Q_t^{\pi}(\mathbf{s},\mathbf{a})\right)\, d\mathbf{a} \qquad (3.2a)\\ &= \operatorname*{softmax}_{\mathbf{a}} Q_t^{\pi}(\mathbf{s},\mathbf{a}).\end{aligned}$$

The difference between $\tilde{V}_t^{\pi}(\mathbf{s})$ and $V_t^{\pi}(\mathbf{s})$ will be investigated in section 3.1.
MaxCausalEnt-IRL learns the reward function for which $\pi_t^{\mathrm{MCE}}(\mathbf{a}|\mathbf{s})$ matches the empirical feature counts $\hat{\boldsymbol{\phi}}$ in expectation. This is achieved by minimizing the dual of the optimization problem of maximizing the causal entropy of the policy constrained on matching the feature counts.
The partial derivative of the dual function with respect to the parameterization of the reward function is given in (Ziebart et al., 2010) as the difference between the empirical feature count $\hat{\boldsymbol{\phi}}$ of the expert and the expected feature count $\tilde{\boldsymbol{\phi}}$ of $\pi_t^{\mathrm{MCE}}(\mathbf{a}|\mathbf{s})$ for the current estimation of $\boldsymbol{\theta}$, i.e.,

$$\frac{\partial \mathcal{G}}{\partial \theta} = \hat{\boldsymbol{\phi}} - \tilde{\boldsymbol{\phi}}. \tag{3.3}$$

Hence, MaxCausalEnt-IRL proposes to compute the reward parameters iteratively using gradient descent. Each iteration thereby involves a backward pass and a forward pass. The backward pass computes $\tilde{V}_t^{\pi}(\mathbf{s})$ for all time steps t starting at the last time step T (based on the current estimate of $\boldsymbol{\theta}$) as described in chapter 1.4.3. For that purpose, it applies Equation 1.2 to compute $\tilde{Q}_t^{\pi}(\mathbf{s},\mathbf{a})$ and Equation 3.2a to compute $\tilde{V}_t^{\pi}(\mathbf{s})$. For LQG systems, $\tilde{Q}_t(\mathbf{s},\mathbf{a})$ is a convex quadratic function for all time steps t. Hence, the policies $\pi_t^{\mathrm{MCE}}(\mathbf{a}|\mathbf{s})$ can be computed using Equation 3.1. The forward pass starts at the (known) initial state distribution $p_1(\mathbf{s})$ and uses the policies $\pi_t^{\mathrm{MCE}}(\mathbf{a}|\mathbf{s})$ to compute the joint distributions $p_t(\mathbf{s},\mathbf{a})$ for all time step based on the system dynamics $p_t(\mathbf{s}'|\mathbf{s},\mathbf{a})$. These joint distributions are then used to compute the expected feature counts $\tilde{\boldsymbol{\phi}}$ for the current estimate of θ. The feature expectations can then be used

to compute the gradient using Equation 3.3.
Since the performance of gradient descent depends heavily on the chosen stepsize, it is sensible to look for ways to adapt it during optimization. In the context of MaxCausalEnt-IRL, the evaluations of the dual function can serve as a basis for stepsize adaption. Unfortunately, the dual function has not been published along with the algorithm and, therefore, had to be derived for this thesis (see appendix A). Fortunately, however, this derivation provides insights into the intrinsics of the algorithm that shall now be discussed.

3.1 Insights

Based on the description in (Ziebart et al., 2010), the optimization problem can be formulated as

$$\underset{\pi_t(\mathbf{a}|\mathbf{s})}{\text{maximize}} \sum_{t=1}^{T-1} \int_{\mathbf{s},\mathbf{a}} -p_t(\mathbf{s})\pi_t(\mathbf{a}|\mathbf{s}) \log \pi_t(\mathbf{a}|\mathbf{s})\, d\mathbf{s} d\mathbf{a} \tag{3.4a}$$

$$\text{subject to} \sum_{t=1}^{T-1} \int_{\mathbf{s},\mathbf{a}} p_t(\mathbf{s})\pi_t(\mathbf{a}|\mathbf{s})\boldsymbol{\phi}_t(\mathbf{s},\mathbf{a}) d\mathbf{s} d\mathbf{a} + \int_{\mathbf{s}} p_T(\mathbf{s})\boldsymbol{\phi}_T(\mathbf{s},\mathbf{0})\, d\mathbf{s} = \hat{\boldsymbol{\phi}}, \tag{3.4b}$$

$$\forall_{t>1} \forall_{\mathbf{s}'} \int_{\mathbf{a}} p_t(\mathbf{s}')\pi_t(\mathbf{a}|\mathbf{s}')\, d\mathbf{a} = \int_{\mathbf{s},\mathbf{a}} p_{t-1}(\mathbf{s})\pi_{t-1}(\mathbf{a}|\mathbf{s}) p(\mathbf{s}'|\mathbf{s},\mathbf{a})\, d\mathbf{s} d\mathbf{a}, \tag{3.4c}$$

$$\forall_{\mathbf{s}}\, p_1(\mathbf{s}) = \mu_1(\mathbf{s}), \tag{3.4d}$$

$$\forall_{t<T} \forall_{\mathbf{s}} \int_{a} \pi_t(\mathbf{a}|\mathbf{s}) d\mathbf{a} = 1, \tag{3.4e}$$

where the causal entropy of the policy $\pi_t(\mathbf{a}|\mathbf{s})$ should be maximized (3.4a) subject to the constraints of matching the feature counts (3.4b) and keeping the state distribution consistent (3.4c), whereas the initial state distribution shall be provided by $\mu_1(\mathbf{s})$ (3.4d). Furthermore, equation (3.4e) ensures that the policy is a probability distribution.
The Lagrangian multiplier of the constraint (3.4b) for matching the feature counts will be denoted as $\boldsymbol{\theta}$ as it corresponds to the weight vector of the reward function. The Lagrangian multipliers of the constraints (3.4c) and (3.4d) will be denoted by $\tilde{V}_1^\pi(\mathbf{s})$ and $\tilde{V}_2^\pi(\mathbf{s}) \ldots \tilde{V}_T^\pi(\mathbf{s})$, respectively, as they relate to the Value function of the policy $\pi_t(\mathbf{a}|\mathbf{s})$.

The optimization problem is solved using Lagrange optimization (Boyd and Vandenberghe, 2009) as demonstrated in Appendix A. The dual function $\mathcal{G}\left(p_t(\mathbf{s}), \boldsymbol{\theta}, \tilde{V}_t^\pi(\mathbf{s})\right)$ is thereby minimized using the partial derivatives

$$\frac{\partial \mathcal{G}\left(p_t(\mathbf{s}),\boldsymbol{\theta},\tilde{V}_t^\pi(\mathbf{s})\right)}{\partial p_t(\mathbf{s})} = \begin{cases} -\tilde{V}_t^\pi(\mathbf{s}) + \log \int_{\mathbf{a}} \exp\left(-\boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s},\mathbf{a}) + \int_{\mathbf{s}'} \tilde{V}_{t+1}^\pi(\mathbf{s}') p(\mathbf{s}'|\mathbf{s},\mathbf{a}) ds\right) d\mathbf{a} & \text{, if } t < T \\ -\tilde{V}_T^\pi(\mathbf{s}) - \boldsymbol{\theta}^\top \boldsymbol{\phi}_T(s,\mathbf{0}) & \text{, if } t = T \end{cases}, \tag{3.5a}$$

$$\frac{\partial \mathcal{G}\left(p_t(\mathbf{s}),\boldsymbol{\theta},\tilde{V}_t^\pi(\mathbf{s})\right)}{\partial \tilde{V}_t^\pi(\mathbf{s})} = \begin{cases} -p_1(\mathbf{s}) + \mu_1(\mathbf{s}) & \text{, if } t = 1 \\ -p_t(\mathbf{s}) + \int_{\mathbf{s},\mathbf{a}} \pi_{t-1}(\mathbf{a}|\mathbf{s}) p_{t-1}(\mathbf{s}) p_{t-1}(\mathbf{s}'|\mathbf{s},\mathbf{a}) & \text{, if } t > 1 \end{cases}, \tag{3.5b}$$

$$\frac{\partial \mathcal{G}\left(p_t(\mathbf{s}),\boldsymbol{\theta},\tilde{V}_t^\pi(\mathbf{s})\right)}{\partial \boldsymbol{\theta}} = \hat{\boldsymbol{\phi}} - \tilde{\boldsymbol{\phi}}. \tag{3.5c}$$

Setting Equation 3.5a equal to zero leads to an update equation for $\tilde{V}_t^\pi(\mathbf{s})$ (backward pass),

$$\tilde{V}_t^\pi(\mathbf{s}) = \begin{cases} \log \int_{\mathbf{a}} \exp\left(-\boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s},\mathbf{a}) + \int_{\mathbf{s}'} \tilde{V}_{t+1}^\pi(\mathbf{s}') p(\mathbf{s}'|\mathbf{s},\mathbf{a}) ds \right) d\mathbf{a} & \text{, if } t < T \\ -\boldsymbol{\theta}^\top \boldsymbol{\phi}_T(s, \mathbf{0}) & \text{, if } t = T \end{cases}. \tag{3.6}$$

Setting Equation 3.5b equal to zero leads to an update equation for $p_t(\mathbf{s})$ (forward pass),

$$p_t(\mathbf{s}) = \begin{cases} \mu_1(\mathbf{s}) & \text{, if } t = 1 \\ \int_{\mathbf{s},\mathbf{a}} \pi_{t-1}(\mathbf{a}|\mathbf{s}) p_{t-1}(\mathbf{s}) p_{t-1}(\mathbf{s}'|\mathbf{s},\mathbf{a}) & \text{, if } t > 1 \end{cases}.$$

As mentioned at the beginning of this chapter, the backward pass and the forward pass are performed consecutively during each iteration in order to compute the expected feature counts $\tilde{\boldsymbol{\phi}}$ of the current approximation of $\boldsymbol{\theta}$, which can then be used for a single step of gradient descent by using equation (3.5c). The common way of computing the Value function $V_t^\pi(\mathbf{s})$ of π_t^{MCE} would involve computing the expectation of the state-action Value function $Q_t^\pi(\mathbf{s},\mathbf{a})$ using Equation 1.3. It is therefore interesting to compare the Value function $V_t^\pi(\mathbf{s})$ that is computed using Equation 1.3 with the Value function $\tilde{V}_t^\pi(\mathbf{s})$ that is computed using Equation 3.6. It turns out, that both Value functions have the same state-dependent part, but differ in an offset that depends on $\boldsymbol{\theta}$,

$$\tilde{V}_t^\pi(\mathbf{s}) = V_t^\pi(\mathbf{s}) - \sum_{i=t}^{T-1} \frac{1}{2}(N_a + \log|2\pi \boldsymbol{\Sigma}_{\mathbf{a}|\mathbf{s},i}^{-1}|), \tag{3.7}$$

where $\boldsymbol{\Sigma}_{\mathbf{a}|\mathbf{s},t}$ denotes the covariance matrix of $\pi_t^{\text{MCE}}(\mathbf{a}|\mathbf{s})$. The proof is given in Appendix C. As the offset does not depend on the state, both backward passes lead to the same policy.

When evaluating the dual function for the current estimates of $p_t(\mathbf{s})$, $\tilde{V}_t^\pi(\mathbf{s})$ and $\boldsymbol{\theta}$, equation (3.5a) and (3.5b) equate to zero for all time steps t. In that case, the dual function simplifies greatly and is given by

$$\mathcal{G}^\star(\boldsymbol{\theta}) = \hat{\boldsymbol{\phi}}^\top \boldsymbol{\theta} - \mathbb{E}_{\mathbf{s}'}\left[\tilde{V}_1^\pi(\mathbf{s}')\right]. \tag{3.8}$$

The expected total reward $\mathbb{E}_{\mathbf{s}'}\left[\tilde{V}_1^\pi(\mathbf{s}')\right]$ can also be computed based on the expected feature count $\tilde{\boldsymbol{\phi}}$,

$$\mathbb{E}_{\mathbf{s}'}\left[\tilde{V}_1^\pi(\mathbf{s}')\right] = \boldsymbol{\theta}^\top \tilde{\boldsymbol{\phi}}. \tag{3.9}$$

Using Equation 3.7, 3.8 and 3.9, the dual function can be expressed in terms of $\tilde{\boldsymbol{\phi}}$,

$$\begin{aligned} \mathcal{G}^\star(\boldsymbol{\theta}) &= \hat{\boldsymbol{\phi}}^\top \boldsymbol{\theta} - \mathbb{E}_{\mathbf{s}'}\left[\tilde{V}_1^\pi(\mathbf{s}')\right]. \\ &= \hat{\boldsymbol{\phi}}^\top \boldsymbol{\theta} - \mathbb{E}_{\mathbf{s}'}\left[V_t^\pi(\mathbf{s}') - \sum_{i=t}^{T-1} \frac{1}{2}(N_a + \log|2\pi \boldsymbol{\Sigma}_{\mathbf{a}|\mathbf{s},i}^{-1}|)\right]. \\ &= \hat{\boldsymbol{\phi}}^\top \boldsymbol{\theta} - \mathbb{E}_{\mathbf{s}'}\left[V_t^\pi(\mathbf{s}')\right] + \sum_{i=t}^{T-1} \frac{1}{2}(N_a + \log|2\pi \boldsymbol{\Sigma}_{\mathbf{a}|\mathbf{s},i}^{-1}|). \\ &= \hat{\boldsymbol{\phi}}^\top \boldsymbol{\theta} - \tilde{\boldsymbol{\phi}}^\top \boldsymbol{\theta} + \sum_{t=1}^{T-1} \log|2\pi \boldsymbol{\Sigma}_{\mathbf{a}|\mathbf{s},i}^{-1}| + \text{const.} \end{aligned} \tag{3.10}$$

The empirical feature count $\hat{\boldsymbol{\phi}}$ and the expected feature count $\tilde{\boldsymbol{\phi}}$ have to be computed anyway in order to evaluate Equation (3.5c), and the covariances $\boldsymbol{\Sigma}_{\mathbf{a}|\mathbf{s},t}$ of the stochastic policies $\pi_t(\mathbf{a}|\mathbf{s})$ are a side product of the backward pass. Therefore, the dual function can be evaluated at each iteration for the current approximation of $\boldsymbol{\theta}$ without any noticeable computational overhead.

3.2 Implementation

This section discusses the practical challenges that arose when employing MaxCausalEnt-IRL for the purpose of learning a non-stationary reward function and how they have been coped with. Applying the algorithm for learning time-dependent reward functions is straightforward and is achieved by introducing independent sets of features for each time step. Thereby, however, the number of features is increased significantly, and learning becomes infeasible if the duration of the demonstrations is discretized into too many partitions. Radial Basis Functions (RBFs) are employed in order to learn a smooth, time-continuous reward function based on a coarse discretization. Additionally, the insights discussed in section 3.1 are utilized by adapting the stepsize based on the dual function (3.10).

3.2.1 Learning a Non-Stationary Reward Function

MaxCausalEnt-IRL is applied to learn T independent, quadratic reward functions

$$\begin{aligned} r_t(\mathbf{s},\mathbf{a}) &= -(\mathbf{s}-\mathbf{g}_t)^\top \mathbf{R}_t(\mathbf{s}-\mathbf{g}_t) - \mathbf{a}^\top \mathbf{H}\mathbf{a} \\ &= -\mathbf{s}^\top \mathbf{R}_t \mathbf{s} + \mathbf{s}^\top \mathbf{r}_t - \mathbf{a}^\top \mathbf{H}\mathbf{a} + \text{const}, \end{aligned}$$

where $\mathbf{R}_t$ must be positive semi-definite and $\mathbf{H}$ must be positive definite. The goal positions $\mathbf{g}_t$ relate to the linear coefficients $\mathbf{r}_t$ via $\mathbf{r}_t = 2\mathbf{R}_t\mathbf{g}_t$. The action penalty matrix $\mathbf{H}$ is assumed to be a time independent diagonal matrix in order to reduce the amount of features.
The features $\boldsymbol{\phi}$ are divided into $T+1$ subsets $\boldsymbol{\phi} = [\boldsymbol{\phi}_0^\top, \boldsymbol{\phi}_1^\top, \dots, \boldsymbol{\phi}_T^\top]^\top$, where $\boldsymbol{\phi}_0$ aggregates the quadratic actions, i.e. $\boldsymbol{\phi}_0 = \sum_{t=1}^{T-1} [a_{1,t}^2, \dots, a_{N_a,t}^2]^\top$. The remaining subsets encode the linear, quadratic and mixed state features for their corresponding time step, i.e.,

$$\forall t \in [1,T] : \boldsymbol{\phi}_t = [s_{1,t}, \dots, s_{N_s,t}, s_{1,t}\cdot s_{1,t}, \dots, s_{1,t}\cdot s_{N_s,t}, s_{2,t}\cdot s_{2,t}, \dots, s_{2,t}\cdot s_{N_s,t}, \dots, s_{N_s,t}\cdot s_{N_s,t}]^\top.$$

The total reward can then be expressed as a linear combination of these features,

$$\begin{aligned} r(\mathbf{s},\mathbf{a}) &= \sum_{t=1}^{T} r_t(\mathbf{s},\mathbf{a}), \\ &= -\boldsymbol{\theta}^\top \boldsymbol{\phi} + \text{const}. \end{aligned}$$

The entries of $\boldsymbol{\theta}$ thereby correspond to the entries of the parameters $\mathbf{r}_t$, $\mathbf{R}_t$ and $\mathbf{H}$ depending on the feature they weight:

- if the feature is a quadratic action term, it corresponds to the respective diagonal element of $\mathbf{H}$.
- if the feature is a quadratic state term, it corresponds to the respective diagonal element of $\mathbf{R}_t$.
- if the feature is a mixed state term, it corresponds to the respective off-diagonal entry times two.
- if the feature is a linear state term, it corresponds to the respective element of $\mathbf{r}_t$.

Where t corresponds to the index of the subset that contains the feature.
Thus, the parameters of the time dependent reward functions $r_t(\mathbf{s},\mathbf{a})$ can be learned, by learning $\boldsymbol{\theta}$ via MaxCausalEnt-IRL as described in chapter 3.

3.2.2 Towards a Compact Representation Using Radial Basis Functions

Learning an independent set of features for each time step increases the number of parameters significantly. In order to soften this increase, the T time dependent reward functions are expressed using N_φ radial basis functions $\varphi_i(t)$, i.e.

$$r_t^\varphi(\mathbf{s},\mathbf{a}) = \sum_{i=1}^{N_\varphi} \varphi_i(t)\left(-\mathbf{s}^\top \mathbf{R}_i \mathbf{s} - \mathbf{s}^\top \mathbf{r}_i\right) - \mathbf{a}^\top \mathbf{H}\mathbf{a},$$

where $\varphi_i(t)$ are normalized gaussian radial basis functions

$$\varphi_i(t) = w_i e^{-\frac{1}{2}\left(\frac{t-c_i}{\sigma_i}\right)^2},$$

with equally spaced centers c_i and variances σ_i. The weights w_i are chosen to normalize the radial basis functions on the interval $[1, T]$, i.e.,

$$\forall i \in [1, N_\varphi] : \sum_{t=1}^{T} \varphi_i(t) = 1.$$

By employing radial basis functions, only $N_\varphi + 1$ subsets of features are required instead of $T + 1$. Again, the first subset, $\boldsymbol{\phi}_0$, shall encode the quadratic actions as shown in section 3.2.1. The remaining subsets, however, now aggregate the linear, quadratic and mixed state features, with respect to the responsibilities of the respective radial basis function, i.e.,

$$\forall i \in [1, N_\varphi] : \boldsymbol{\phi}_i^\varphi = \sum_{t=1}^{T} \varphi_i(t)[s_{1,t}, \ldots, s_{N_s,t}, s_{1,t} \cdot s_{1,t}, \ldots, s_{1,t} \cdot s_{N_s,t}, s_{2,t} \cdot s_{2,t}, \ldots, s_{2,t} \cdot s_{N_s,t}, \ldots, s_{N_s,t} \cdot s_{N_s,t}]^\top.$$

By composing the feature vector ϕ^φ of these subsets, the total reward is again a linear combination of the features,

$$\begin{aligned} r(\mathbf{s},\mathbf{a}) &= \sum_{t=1}^{T} r_t^\varphi(\mathbf{s},\mathbf{a}) \\ &= -\boldsymbol{\theta}^\top \boldsymbol{\phi}^\varphi, \end{aligned}$$

and the parameters of the time dependent reward functions can thus be learned via $\boldsymbol{\theta}$.
Besides decreasing the number of parameters, the RBF-based representation leads to time continuous reward functions by assuming a reward function that changes smoothly over time.

3.2.3 Ensuring a Positive Semi-Definite Reward Function

As the entries of θ directly correspond to the elements of the parameters $\mathbf{r}_x$, $\mathbf{R}_x$ and $\mathbf{H}$, the gradient based optimization may lead to non-positive semidefinite state costs $\mathbf{R}_x$, which would violate the LQG assumption and thereby lead to failure.
This can be avoided by applying gradient descent for learning the entries of a lower triangular matrix $\mathbf{L}_x$ instead of learning the entries of $\mathbf{R}_x$ directly. $\mathbf{R}_x$ can then be constructed by

$$\mathbf{R}_x = \mathbf{L}_x \mathbf{L}_x^\top.$$

As Cholesky has shown, any positive semi-definite matrix can be decomposed into such a matrix product and any such product yields a positive semi-definite matrix.
In order to learn the entries of $\mathbf{L}_x$ an additional weight vector $\tilde{\boldsymbol{\theta}}$ of the same size as $\boldsymbol{\theta}$ is introduced, such that each entry of that weight vector corresponds to a different entry of the respective triangular matrix $\mathbf{L}_x$, goal position $\mathbf{g}_x$ or action cost matrix $\mathbf{H}$. Gradient descent is then applied to update $\tilde{\boldsymbol{\theta}}$ instead of $\boldsymbol{\theta}$ using

$$\frac{\partial \mathscr{G}\left(p_t(\mathbf{s}), \boldsymbol{\theta}, \tilde{V}_t^{\pi}(\mathbf{s})\right)}{\partial \tilde{\boldsymbol{\theta}}} = \frac{\partial \mathscr{G}\left(p_t(\mathbf{s}), \boldsymbol{\theta}, \tilde{V}_t^{\pi}(\mathbf{s})\right)}{\partial \boldsymbol{\theta}} \frac{d\boldsymbol{\theta}}{d\tilde{\boldsymbol{\theta}}}.$$

The gradient $\frac{d\boldsymbol{\theta}}{d\tilde{\boldsymbol{\theta}}}$ can be solved in closed form and produces a block diagon matrix

$$\frac{d\boldsymbol{\theta}}{d\tilde{\boldsymbol{\theta}}} = \begin{pmatrix} \frac{d\boldsymbol{\theta}_0}{d\tilde{\boldsymbol{\theta}}_0} & 0 & \dots & 0 \\ 0 & \frac{d\boldsymbol{\theta}_1}{d\tilde{\boldsymbol{\theta}}_1} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \frac{d\boldsymbol{\theta}_N}{d\tilde{\boldsymbol{\theta}}_N} \end{pmatrix},$$

where $\boldsymbol{\theta}_i$ and $\tilde{\boldsymbol{\theta}}_i$ indicate the subsets of $\boldsymbol{\theta}$ and $\tilde{\boldsymbol{\theta}}$ that correspond to the respective subset $\boldsymbol{\phi}_i$ of $\boldsymbol{\phi}$.

3.2.4 Adapting the Stepsize Based on the Dual Function

As shown in section 3.1, the dual function can be evaluated very efficiently using Equation 3.10. Therefore, it makes sense to take these evaluations as a basis for stepsize adaption during gradient descent. This is done by checking after each gradient step, whether the dual function did indeed decrease. Only then, the gradient step is accepted and the stepsize is increased by a fixed factor α. Whenever the dual function did not decrease, the last gradient step is withdrawn and the stepsize is decreased by β. The parameters α and β are chosen such that the relative increase after a successful step is smaller than the relative decrease after an unsuccessful step in order to avoid having to withdraw too often.

4 Inverse Reinforcement Learning by Matching State Distributions

This chapter presents a novel approach for learning non-stationary reward functions from demonstrations. In contrast to Maximum Causal Entropy Inverse Reinforcement Learning it is not based on matching the feature expectations of the learned policy with the observed feature counts. Instead, it aims at matching the state distributions that result from the learned policies, $p_t^{\pi}(\mathbf{s})$ with the observed state distributions, $q_t(\mathbf{s})$. By defining the objective in terms of distributions instead of defining them in terms of feature counts, the features do not have to be explicitly defined. Instead, the structure of the reward is automatically determined based on the structure of the distributions. Matching the state distribution is achieved by minimizing the relative entropy between those probability distributions. Additionally, inspired by Relative Entropy Policy Search (REPS), the relative entropy of the current policy $\pi_t(\mathbf{a}|\mathbf{s})$ is minimized with respect to the last policy $q_{0,t}(\mathbf{a}|\mathbf{s})$ in order to reduce the loss of information between iterations.

4.1 The Optimization Problem

Inverse Reinforcement Learning by Matching State Distributions is based on the following optimization problem:

$$\underset{\pi_t(\mathbf{a}|\mathbf{s})}{\text{minimize}} \quad \sum_{t=1}^{T} \int_{\mathbf{s}} p_t(\mathbf{s}) \log \frac{p_t(\mathbf{s})}{q_t(\mathbf{s})} d\mathbf{s} + \sum_{t=1}^{T-1} \int_{\mathbf{s}} p_t(\mathbf{s}) \int_{a} \pi_t(\mathbf{a}|\mathbf{s}) \log \frac{\pi_t(\mathbf{a}|\mathbf{s})}{q_{0,t}(\mathbf{a}|\mathbf{s})} d\mathbf{a} d\mathbf{s} \tag{4.1a}$$

$$\text{subject to} \quad \forall_{t>1} \forall_{\mathbf{s}'} \int_{a} p_t(\mathbf{s}') \pi_t(\mathbf{a}|\mathbf{s}') d\mathbf{a} = \int_{\mathbf{s},\mathbf{a}} p_{t-1}(\mathbf{s}) \pi_{t-1}(\mathbf{a}|\mathbf{s}) p(\mathbf{s}'|\mathbf{s},\mathbf{a}) d\mathbf{s} d\mathbf{a}, \tag{4.1b}$$

$$\forall_{\mathbf{s}}\, p_1(\mathbf{s}) = \mu_1(\mathbf{s}), \tag{4.1c}$$

$$\forall_{t<T} \forall_{\mathbf{s}} \int_{a} \pi_t(\mathbf{a}|\mathbf{s}) d\mathbf{a} = 1. \tag{4.1d}$$

The constraints (4.1b, 4.1c, 4.1d) are exactly the same as their counterparts in the Maximum Causal Entropy optimization problem (3.4c, 3.4d, 3.4e). The goal of the optimization (4.1a), however, is different, as it aims at minimizing the two Kullback-Leibler divergences (KLs). It is to note, that the target policy $q_t(\mathbf{a}|\mathbf{s})$ does not have to be set to the policy of the last iteration, but could be set to a fixed distribution instead. In this case, the corresponding KL does not lead in learning action costs, but serves as regularization instead. Again, the Lagrangian multipliers for the constraints (4.1b) and (4.1c) relate to the Value function of the policy $\pi_t(\mathbf{a}|\mathbf{s})$ and will be therefore denoted by $\tilde{V}_t^{\pi}(\mathbf{s})$. As the constraint of matching the feature counts has been dropped, the optimization problem does not possess a Lagrangian multiplier that corresponds to the parameters of the reward function.
Indeed, it might look like the optimization problem does not refer to any form of reward at all, giving

rise to the question of how it can be employed to tackle the problem of IRL. This question can be best answered by looking at the partial derivatives of the dual function,

$$\frac{\partial \mathcal{G}(p_t(\mathbf{s}), \tilde{V}_t(\mathbf{s}))}{\partial p_t(\mathbf{s})} = \begin{cases} \tilde{V}_T(\mathbf{s}) + \log \frac{p_t(\mathbf{s})}{q_t(\mathbf{s})} + 1 & \text{, if } t = T \\ \tilde{V}_t(\mathbf{s}) - \log \int_{\mathbf{a}} \exp\left(\log q_{0,t}(\mathbf{a}|\mathbf{s}) + \frac{\log q_t(\mathbf{s})}{\log p_t(\mathbf{s})} - 1 + \int_{\mathbf{s}'} \tilde{V}_{t+1}(\mathbf{s}') p(\mathbf{s}'|\mathbf{s}, \mathbf{a}) d\mathbf{s}'\right) & \text{, if } t < T \end{cases} \tag{4.2a}$$

$$\frac{\partial \mathcal{G}(p_t(\mathbf{s}), \tilde{V}_t(\mathbf{s}))}{\partial \tilde{V}_t(\mathbf{s})} = \begin{cases} -p_1(\mathbf{s}) + \mu_1(\mathbf{s}) & \text{, if } t = 1 \\ -p_t(\mathbf{s}) + \int_{\mathbf{s},\mathbf{a}} \pi_{t-1}(\mathbf{a}|\mathbf{s}) p_{t-1}(\mathbf{s}) p_{t-1}(\mathbf{s}'|\mathbf{s}, \mathbf{a}) & \text{, if } t > 1 \end{cases}. \tag{4.2b}$$

The derivations are given in Appendix B. Setting the derivative with respect to the state distribution (4.2a) to zero leads to the backward pass to compute $\tilde{V}_t^{\pi}(\mathbf{s})$, while setting the derivative with respect to the Value function (4.2b) equal to zero leads to the forward pass to compute $p_t(\mathbf{s})$. Actually, the forward pass is exactly the same for MaxCausalEnt-IRL (3.5b) and IRL-MSD. More interestingly, however, the backward pass of IRL-MSD is the same as the backward pass of MaxCausalEnt-IRL (3.5a) except that the reward function $-\theta^\top \phi(\mathbf{s}, \mathbf{a})$ has been replaced by the term

$$r_t(\mathbf{s}, \mathbf{a}) = \log(q_{0,t}(\mathbf{a}|\mathbf{s})) + \log(q_t(\mathbf{s})) - \log(p_t(\mathbf{s})) - 1, \tag{4.3}$$

that serves as reward signal for the optimal policy $\pi_t^\star(\mathbf{a}|\mathbf{s})$, which is again, as in MaxCausalEnt-IRL, proportional to the exponential of the state action Value function $Q_t(\mathbf{s}, \mathbf{a})$,

$$\pi_t^\star(\mathbf{a}|\mathbf{s}) \propto \exp(Q_t(\mathbf{s}, \mathbf{a})) = \exp\left(r_t(\mathbf{s}, \mathbf{a}) + \int_{\mathbf{s}'} \tilde{V}_{t+1}(\mathbf{s}') p(\mathbf{s}'|\mathbf{s}, \mathbf{a}) d\mathbf{s}'\right). \tag{4.4}$$

Hence, the reward function depends directly on the target policy $q_{0,t}(\mathbf{a}|\mathbf{s})$, the target state distribution $q_t(\mathbf{s})$, and $p_t(\mathbf{s})$, the state distribution that results from the policy that minimizes the objectives (4.1a). As the state distribution $p_t(\mathbf{s})$ depends on the policy, and the policy depends on the Value function $\tilde{V}_t(\mathbf{s})$, the reward function is actually defined recursively. The features of the reward function thus evolve naturally, depending on the types of the involved probability distributions. Most notably, if all distributions are Gaussian, and the system dynamics are linear with Gaussian noise, the reward function is quadratic in states and actions. For LQG systems, $\pi_t^\star(\mathbf{a}|\mathbf{s})$ always takes the form of a linear PD controller with Gaussian noise. Therefore, in the following, the target policy is assumed to be of the same form, i.e. $q_{0,t}(\mathbf{a}|\mathbf{s}) = \mathcal{N}(\mathbf{a}|\mathbf{K}_{0,t}\mathbf{s} + \mathbf{k}_{0,t}, \Sigma_{q0,t})$. Then, the parameters of the reward function

$$r_t(\mathbf{s}, \mathbf{a}) = -\begin{pmatrix} \mathbf{s} \\ \mathbf{a} \end{pmatrix}^\top \begin{pmatrix} \mathbf{R}_t & \mathbf{F}_t \\ \mathbf{F}_t^\top & \mathbf{H}_t \end{pmatrix} \begin{pmatrix} \mathbf{s} \\ \mathbf{a} \end{pmatrix} + \begin{pmatrix} \mathbf{s} \\ \mathbf{a} \end{pmatrix}^\top \begin{pmatrix} \mathbf{r}_t \\ \mathbf{h}_t \end{pmatrix} + \text{const} \tag{4.5}$$

can be computed by

$$\mathbf{R}_t = \frac{1}{2}\left(\Sigma_{q,t}^{-1} + \mathbf{K}_{0,t}^\top \Sigma_{q0,t}^{-1} \mathbf{K}_{0,t} - \Sigma_{p,t}^{-1}\right), \tag{4.6a}$$

$$\mathbf{r}_t = \Sigma_{q,t}^{-1}\mu_{q,t} - \mathbf{K}_{0,t}^\top \Sigma_{q0,t}^{-1} \mathbf{k}_{0,t} - \Sigma_{p,t}^{-1}\mu_{p,t}, \tag{4.6b}$$

$$\mathbf{F}_t = -\frac{1}{2}\mathbf{K}_{0,t}^\top \Sigma_{q0,t}^{-1}, \tag{4.6c}$$

$$\mathbf{H}_t = -\frac{1}{2}\Sigma_{q0,t}^{-1}, \tag{4.6d}$$

$$\mathbf{h}_t = \Sigma_{q0,t}^{-1}\mathbf{k}_{0,t}. \tag{4.6e}$$

The derivations can be found in Appendix B.

4.2 The Proposed Algorithm

The partial derivatives of the dual function (equation 4.2a and equation 4.2b) lead to an iterative algorithm similar to MaxCausalEnt-IRL. Starting with an initial estimate of the reward function, the softmax Value functions $\tilde{V}_t^{\pi}(\mathbf{s})$ as well as the corresponding policies $\pi_t(\mathbf{a}|\mathbf{s})$ are computed via the backward pass (4.2a), and the resulting state distributions $p_t(\mathbf{s})$ are computed via the forward pass (4.2b). However, IRL-MSD utilizes these state distributions in order to compute the parameters of the reward function, that is optimal with respect to the current estimates, in one shot, whereas MaxCausalEnt-IRL utilizes them merely to update the parameters by a single step of gradient descent. It should be noted, that the computational cost of computing the optimal parameters (Equations 4.6a to 4.6e) does not exceed the cost of computing the gradient of the parameters via feature expectations.
The IRL-MSD optimization problem allows different interpretations regarding $q_{0,t}$, that lead to two different variants of the algorithm. The first variant treats $q_{0,t}$ as an estimate of the optimal policy, that is updated after each iteration, whereas the second variant treats $q_{0,t}$ as a regularization on the policy which is not changed during optimization. The effects of these different point of views shall be discussed in the following subsections.

4.2.1 Using the Policy KL for Learning Action Costs

The first variant of IRL-MSD, which will be referred to by MSD-ACL in the following, regards $q_{0,t}$ as an estimate of the optimal policy. It is updated after each iteration by setting it equal to the policy that was computed during that iteration. Thereby, similar to REPS, the current policy should stay close to the last one. However, in contrast to REPS the relative entropy is not bounded. This allows for faster convergence but does not give any guarantee, that the policy is indeed close to the last one. Therefore, if such a guarantee is needed, e.g. when learning a local linearization of the state dynamics, a bound on that KL should be introduced by reformulating it as a constraint. In the following, however, it will be assumed that no such bound is necessary. The relative entropy then merely serves the purpose of learning the policy that matches the target state distribution as close as possible.
MSD with Action Cost Learning (MSD-ACL) would per default not regularize the actions at all, however, it can be easily augmented with action regularization by adding a fixed offset $\mathbf{H}_{reg,t}$ to the action cost matrix $\mathbf{H}_t$ (equation 4.6d) whenever computing the reward function. This corresponds to adding a corresponding punishment term to the objective of the optimization problem.
A more serious disadvantage of MSD-ACL comes from the fact, that setting $q_{0,t}(\mathbf{a}|\mathbf{s})$ equal to a noisy PD-controller, leads to learning state-dependent action costs $\mathbf{F}_t$ and potentially non-zero action goals $\mathbf{h}_t$. While this might be fine for many cases, it has the drawback that it leads to a reward function that is difficult to interpret. When state dependent action costs should be avoided, the second variant of IRL-MSD can be employed.

4.2.2 Using the Policy KL for Regularization

The second variant, which is in the following named “MSD with KL-based Regularization (MSD-REG)”, uses $q_{0,t}$ for regularization. The controller gains $\mathbf{K}_{0,t}$ and $\mathbf{k}_{0,t}$ are set to zero, and its noise $\Sigma_{q0,t}$ serves as weight of the regularization. If the covariance is high, deviations from the zero action are punished less and the regularization is thus low. Respectively, a low covariance would result into strong regularization. However, misusing a KL for regularization comes at a cost, because minimizing the relative entropy might actually result in increasing the action costs artificially just for the purpose of matching $\Sigma_{q0,t}$ which might lead to suboptimal learning. Nevertheless, it allows to learn a state-dependent reward function for a given action cost matrix $\mathbf{H}_t$. Such a reward function has the advantage, that goal states as well as their relevance can be directly inferred.

The effects on the performance, that result from using the KL for regularization will be investigated in chapter 5.

4.2.3 Ensuring a Positive Definite Reward Function

Computing $\mathbf{R}_t$ or $\mathbf{H}_t$ according to equation 4.6a or 4.6d does not guarantee positive semi-definite cost matrices. Such reward functions should be avoided, because they break the LQG assumption and may lead to non-Gaussian policies. For IRL-MSD, a non-positive semi-definite reward function indicates that the variance with respect to the goal position is smaller than the target variance. The goal position then is no longer an attractor, but serves as detractor instead, i.e. the reward increases with the distance to that position. As it is not admissible to allow such reward functions, the cost matrices can be checked immediately after computation for positive definiteness. If they are not positive definite, spectral decomposition can be utilized in order to replace all negative eigenvalues by a small positive number and then to transform the matrix back into its original basis.

4.2.4 Disregarding Features by Matching Marginals

In some cases, it is not desirable to match the demonstrations with respect to all state trajectories. For example, kinesthetic teaching might be used in order to demonstrate the via points of a movement regarding only the task space positions, but not the corresponding velocities. It might seem to be a drawback of IRL-MSD, that the features cannot be chosen directly, but are rather implicitly defined by the target distribution. However, defining the type of the target distribution $p_t(\mathbf{s})$ does not need to be less intuitive than defining the type of the reward function. It could be argued, that defining the type of the target distribution is more intuitive, since it can be directly estimated based on the trajectories, whereas inferring the structure of the reward function from the trajectories is slightly less direct. However, for many practical applications it probably does not make any difference in the end and the problem of choosing the proper features and the problem of choosing the proper target distributions are just two different views on the same problem.
Ignoring certain state dimensions for MaxCausalEnt-IRL is achieved by ignoring the corresponding features when computing the feature counts, whereas for IRL-MSD it is achieved by ignoring the corresponding dimensions of the random variable when computing the KL. This leads to a slightly more general formulation of the objective of the optimization function, i.e. Equation 4.1a becomes

$$\underset{\pi_t(\mathbf{a}|\mathbf{s})}{\text{minimize}} \quad \sum_{t=1}^{T} \int_{\mathbf{s}} p_t(\mathbf{s}) \log \frac{\tilde{p}_t(\mathbf{s})}{\tilde{q}_t(\mathbf{s})} d\mathbf{s} + \sum_{t=1}^{T-1} \int_{\mathbf{s}} p_t(\mathbf{s}) \int_{a} \pi_t(\mathbf{a}|\mathbf{s}) \log \frac{\pi_t(\mathbf{a}|\mathbf{s})}{q_{0,t}(\mathbf{a}|\mathbf{s})} d\mathbf{a} d\mathbf{s}, \tag{4.7}$$

where $\tilde{p}_t(\mathbf{s})$ and $\tilde{q}_t(\mathbf{s})$ are the respective marginal distributions. The only difference that results from this small modification is that the entries of $\mathbf{R}_t$ and $\mathbf{r}_t$ that correspond to the disregarded states have to be filled up with zeros, i.e. they have to be computed by

$$\mathbf{R}_t = \frac{1}{2}\left(\mathbf{D}^\top(\Sigma_{\tilde{q},t}^{-1} - \Sigma_{\tilde{p},t}^{-1})\mathbf{D} + \mathbf{K}_{0,t}^\top \Sigma_{q0,t}^{-1} \mathbf{K}_{0,t}\right), \tag{4.8a}$$

$$\mathbf{r}_t = \mathbf{D}^\top(\Sigma_{\tilde{q},t}^{-1}\mu_{\tilde{q},t} - \Sigma_{\tilde{p},t}^{-1}\mu_{\tilde{p},t}) - \mathbf{K}_{0,t}^\top \Sigma_{q0,t}^{-1} \mathbf{k}_{0,t}, \tag{4.8b}$$

where $\mathbf{D}$ is constructed by removing all rows of the N_s-by-N_s identity matrix, that correspond that disregarded states.

4.2.5 A Pseudo-code Implementation of IRL-MSD

A pseudo-code implementation of IRL-MSD for LQG systems is given in Algorithm 1 for the variants discussed in section 4.2.1 and 4.2.2. The modifications given in section 4.2.3 and 4.2.4 can be incorporated by computing the rewards accordingly.

```
input : $q(\mathbf{s})$ ;                        /* target state distributions for all time steps */
        $q_0^{(0)}(\mathbf{a}|\mathbf{s})$ ;                        /* target policies for all time steps */
        $p(\mathbf{s}'|\mathbf{s},\mathbf{a})$ ;                        /* system dynamics for all time steps */
        $\mu_0(\mathbf{s})$ ;                     /* state distribution of the first time step */
        $T$ ;                                                    /* time horizon */
        learnActionCosts ; /* boolean indicating whether action costs should be learned */
output: $\mathbf{H}^{(i)}, \mathbf{h}^{(i)}, \mathbf{F}^{(i)}, \mathbf{R}^{(i)}, \mathbf{r}^{(i)}$ ;           /* reward parameters for all time steps */
/* Initialize reward parameters                                                    */
$[\mathbf{H}^{(0)}, \mathbf{h}^{(0)}, \mathbf{F}^{(0)}, \mathbf{R}^{(0)}, \mathbf{r}^{(0)}] \leftarrow$ initialize()
$i \leftarrow 0$
while not converged do
    /* compute the Value functions and policies using equation 4.2a and 4.4         */
    $[V^{(i)}(\mathbf{s}), \pi(\mathbf{a}|\mathbf{s})^{(i)}] \leftarrow$ backward_pass($\mathbf{H}^{(i)}, \mathbf{h}^{(i)}, \mathbf{F}^{(i)}, \mathbf{R}^{(i)}, \mathbf{r}^{(i)}$) ;     /* See Appendix B */
    /* compute the state distributions using equation 4.2b                          */
    $p^{(i)}(\mathbf{s}) \leftarrow$ forward_pass($\pi^{(i)}(\mathbf{a}|\mathbf{s}), p(\mathbf{s}'|\mathbf{s},\mathbf{a}), \mu_0(\mathbf{s})$)
    /* set next target policies                                                     */
    if learnActionCosts then
        $q_0^{(i+1)}(\mathbf{a}|\mathbf{s}) \leftarrow \pi^{(i)}(\mathbf{a}|\mathbf{s})$
    else
        $q_0^{(i+1)}(\mathbf{a}|\mathbf{s}) \leftarrow q_0^{(i)}(\mathbf{a}|\mathbf{s})$
    /* compute reward parameters for the next iteration using equation 4.6a to 4.6e
       */
    $[\mathbf{H}^{(i+1)}, \mathbf{h}^{(i+1)}, \mathbf{F}^{(i+1)}, \mathbf{R}^{(i+1)}, \mathbf{r}^{(i+1)}] \leftarrow$ compute_reward($q(\mathbf{s}), q_0^{(i)}(\mathbf{a}|\mathbf{s}), p^{(i)}(\mathbf{s})$)
    /* iterate                                                                      */
    $i \leftarrow i + 1$
```

Algorithm 1: Pseudo-code Implementation of IRL-MSD

5 Evaluation

The different approaches are evaluated on a double integrator as toy example. A double integrator is a dynamics model of an underactuated systems that possesses twice as many state dimensions as action dimensions, where these two states correspond to the first and second integration of the corresponding action. Thereby the actions can be interpreted as accelerations and the states as velocities and positions. The velocity at time step t, v_t, can be approximated using the velocity and acceleration of the last time step, i.e. $v_t = v_{t-1} + \Delta v \approx v_{t-1} + a_{t-1}\Delta t$, where Δt is the difference in time between two time steps. Similarly, the position at time step t, x_t can be approximated $x_t \approx x_{t-1} + v_{t-1}\Delta t$. For a single action and a time discretization $\Delta t = 0.1$, this corresponds to linear system dynamics $s_{t+1} = \mathbf{As} + \mathbf{Ba}$, with

$$\mathbf{A} = \begin{pmatrix} 1 & 0.1 \\ 0 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{B} = \begin{pmatrix} 0 \\ 0.1 \end{pmatrix}. \tag{5.1}$$

The dimensions can be increased by adding additional actions and their corresponding pairs of states. The matrices of such higher dimensional systems are block-diagonal and composed of the blocks given in Equation 5.1.
The approximation error is modeled as Gaussian noise, leading to non-deterministic system dynamics.

5.1 Assuming Known Statistics of the Expert

The evaluations within this section assume that the true policy of the expert is known. Therefore, the state distributions resulting from that policy are used in order to compute the empirical feature counts $\hat{\phi}$ for MaxCausalEnt-IRL and, respectively, are used as target distributions $q_t(\mathbf{s})$ for IRL-MSD. The effects of the sampling error are thereby removed when comparing the different approaches.
The system dynamics are given by a double integrator with N_a action dimensions and, thus, $N_s = 2N_a$ state dimensions. The MDP lasts for $T = 50$ time steps. The true reward function is quadratic in states and actions. The action costs are not correlated with the states and are time-independent. The action cost matrix is given by $\mathbf{H} = \sigma_a^2 \mathbb{I}^{N_a}$, where $\mathbb{I}^{N_a}$ denotes the N_a-by-N_a identity matrix and $\sigma_a^2 = 0.01$. State costs are only given at time step 25 and time step 50, and are based on a projection of the seven positions to a three-dimensional space. This relates to the problem of task space control of a N_a-link manipulator, whereat the forward kinematics are given by the projection matrix.
For the following experiments, the number of links was set to three. The target distribution was computed using optimal control on the true reward function. Figure 5.1 shows the resulting state distribution of the first dimension in task space. The shaded area corresponds to the 2σ confidence interval. The right side shows an estimate based on seven samples.

5.1.1 IRL by Matching State Distributions

MSD-REG and MSD-ACL are evaluated for different initial covariance matrices $\mathbf{\Sigma}_{q0,t}$. The mean of the target policy $q_0(\mathbf{a})$ is set equal to zero, independent of the state. Only a single parameter, σ_0, has to be chosen by setting $\mathbf{\Sigma}_{q0,t} = \sigma_0^2 \mathbb{I}^{N_a}$.
MSD-REG uses σ_0^2 to control regularization, whereat high values of σ_0^2 correspond to low regularization and vice versa. MSD-ACL uses σ_0^2 as initial guess of the target policy that is updated during optimization.

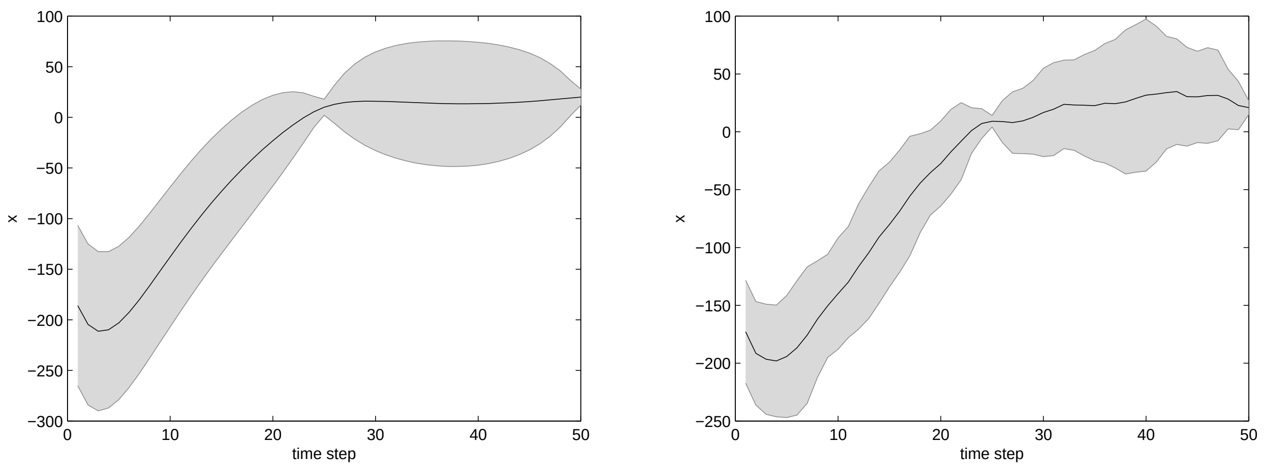

Figure 5.1.: The expert trajectory distributions of the first dimension in task space plotted over time. Left: true distribution, Right: estimate based on seven samples

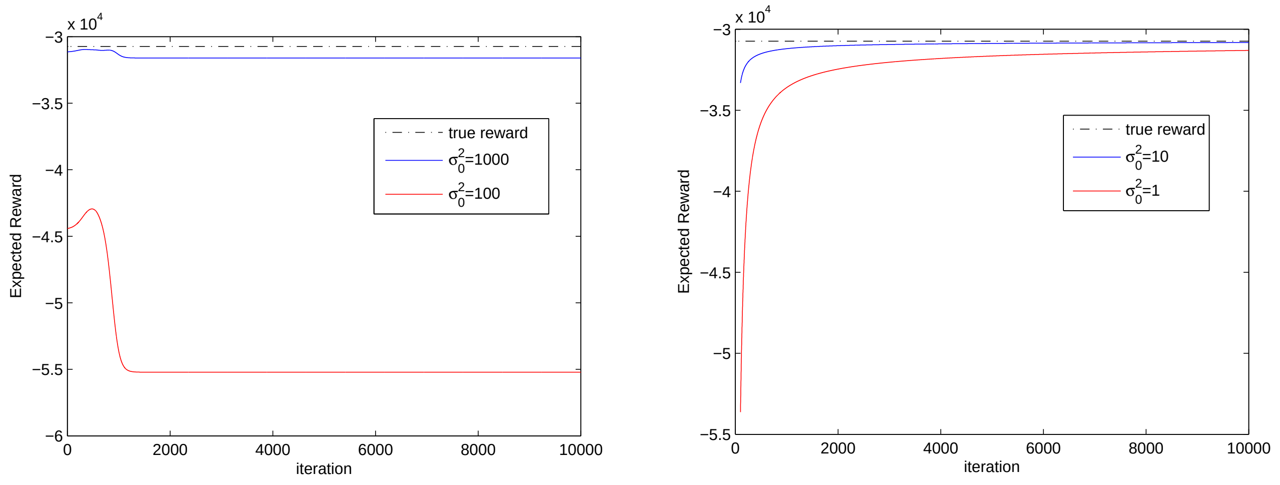

Figure 5.2.: The expected reward for MSD-REG and MSD-ACL are shown for different initializations. Left: MSD-REG achieves a good performance already after the first iteration, however converges to a solution that performs worse. Right: The expected reward of MSD-ACL converges to the one of the expert. The higher initialization has a faster speed of convergence.

Figure 5.2 shows the expected reward of the optimal controller with respect to the learned reward function for good values of σ_0^2. As for all other conducted experiments in this chapter, the expected reward is computed based on the true reward function. The values for σ_0^2 have been found in a previous experiment. Higher values have led to instabilities for both approaches.
Interestingly, MSD-REG attains the best performing reward functions after only a few iterations and subsequently converges to a solution that performs slightly worse.
MSD-ACL converges smoothly to the expert performance. The speed of convergence is faster for the initialization with higher variance.
The state distributions of the optimal controller on the learned reward functions are depicted in Figure 5.3. The initialization was set to $\sigma_0^2 = 1000$ for MSD-REG and to $\sigma_0^2 = 10$ for MSD-ACL. The plot on the left side shows the state distributions after one (red) and after five thousand iterations (blue) of MSD-REG. The distribution after a single iterations has clearly lower variance than the baseline distribution (black). The state distribution after five thousand iterations matches the baseline distribution better,

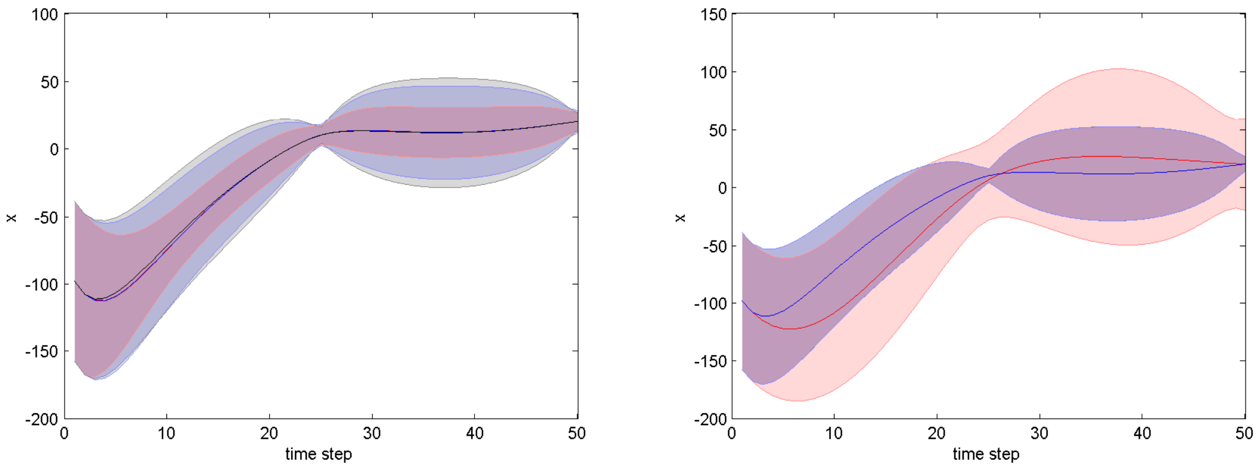

Figure 5.3.: The trajectories of the first dimension in task space are shown for MSD-REG (left) and MSD-ACL (right). The initial estimate of MSD-REG (red) has clearly lower variance than the baseline distribution. The state distribution of MSD-ACL after five thousand iterations (blue) is indistinguishable from the baseline distribution.

even though it performs worse in terms of expected reward.
The plot on the right side shows the corresponding distributions for MSD-ACL. The state distribution after a single iteration (red) approximates the target distribution worse than the respective distribution of MSD-REG because it started with a higher regularization. The state distribution after five thousand iterations (blue) matches the baseline distribution exactly.

5.1.2 Comparison with Maximum Causal Entropy IRL

MSD-ACL with $\sigma_0^2 = 10$ is compared to MaxCausalEnt-IRL. The modification based on radial basis function (Chapter 3.2.2) is not shown, because it does not lead to significant differences for the given task (except, of course, if the number of centers is chosen very low, which impairs the performance significantly). For MaxCausalEnt-IRL the action costs are assumed to be known in order to reduce the amount of features. The step size is adapted after each iteration of gradient descent by multiplying it with $\alpha = 1.01$ or $\beta = 0.5$ respectively, as described in chapter 3.2.4. The algorithms are compared with respect to the number of performed iterations. An additional comparison based on the computational time spent would be uninspiring, because both algorithms need approximately the same time per iteration.
Figure 5.4 shows the expected rewards after each iteration. The blue curve relates to MSD-ACL and is the same as the corresponding curve in Figure 5.2. The red curve shows the performance of MaxCausalEnt-IRL. It converges significantly slower than MSD-ACL.
Figure 5.5 shows the resulting state distributions for MaxCausalEnt-IRL after one thousand (red) and after ten thousand iterations (blue). These distributions are based on the greedy, optimal controller. The corresponding distribution for the true reward function is shown in black. MaxCausalEnt-IRL succeeds in matching the mean very accurately already after thousand iterations, but has too high variance at the goal positions. After ten thousand iterations the variances are matched more closely, however, they are still slightly to large at the goal positions.
It is also interesting to inspect the learned parameters directly. Both approaches learn reward function that only assign high rewards at time steps that are close to the critical time steps 25 and 50, thus, the remaining time steps can be neglected. MSD-ACL learns state-dependent action costs and is therefore difficult to interpret. Hence, MSD-REG and MaxCausalEnt-IRL are analyzed, based on the reward functions that have been learned after ten thousand iterations. The learned reward matrices $\mathbf{R}_t$ and goal

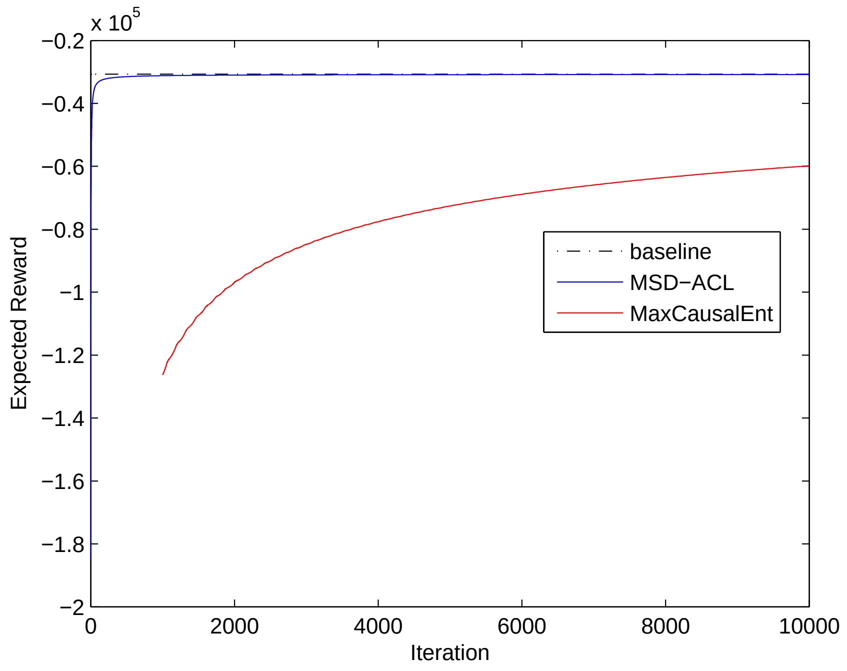

Figure 5.4.: The expected reward of MSD-ACL and MaxCausalEnt-IRL is compared after each iteration based on the true reward function. MSD-ACL converges significantly faster to the expected reward of the expert.

positions $\mathbf{g}_t$ are mapped to the task space using the projection matrix. This transformation yields the low dimensional reward parameters $\mathbf{R}_t^{\mathrm{TS}}$ and $\mathbf{g}_t^{\mathrm{TS}}$ that can be compared to the true reward function. For ease of illustration, only the first task space dimension is considered and reward correlations are ignored. Figure 5.6 shows the first dimension of the task space goal positions (solid line) and the corresponding entries of $\mathbf{R}_t^{\mathrm{TS}}$ (radius of shaded area) for the interesting time steps. Interestingly, MaxCausalEnt-IRL learns goal positions that are distant from the desired trajectory, especially at time steps shortly before the critical ones. The goal positions are given in table 5.1. MSD-REG succeeds in extracting the true goal positions at the critical time steps 25 and 50. The goal positions learned by MaxCausalEnt-IRL are very different from the true goal positions even for high-reward time steps.

time step	23	24	25	26	27	48	49	50
actual	0	0	10	0	0	0	0	20
MSD-REG	4.1	7.4	9.9893	11.5	12.2	17.9	19	19.9991
MaxCausalEnt-IRL	1.2E5	-174.6	238.8	12.2	10.5	7.4E3	-7.9E5	891

Table 5.1.: The table compares the goal positions around the critical time steps for the first dimension in task space. MSD-REG extracts the actual goal positions at the critical time steps (25 and 50) very accurately. These values are given with higher precision.
MaxCausalEnt-IRL learns wrong goal positions at the critical time steps and compensates this by choosing goal positions that are very different from zero for the preceding time steps.

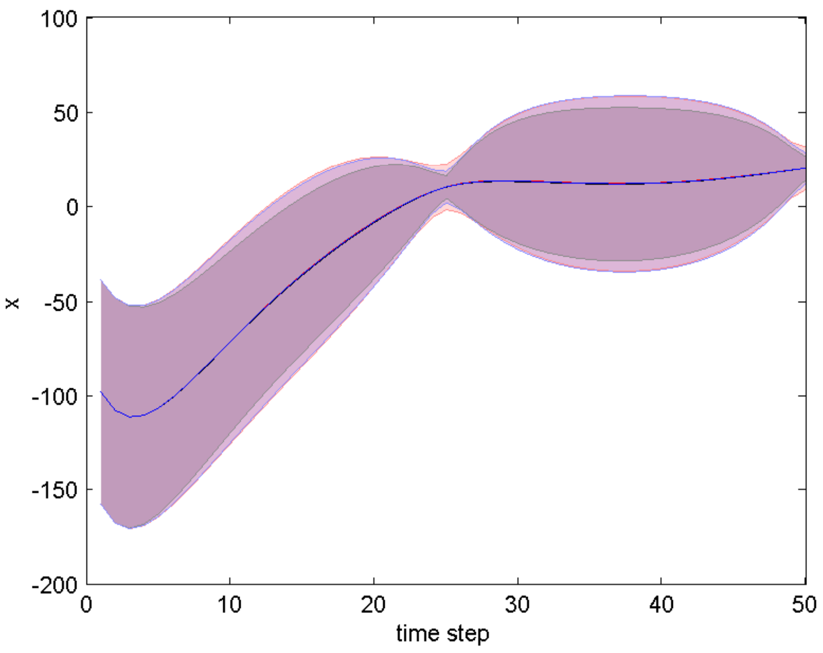

Figure 5.5.: The state distribution of the optimal controller based on the reward learned by Maximum Causal Entropy IRL is shown after 1000 iterations (red) and after 10000 iterations (blue). The state distribution for the true reward function is shown in black. The learned reward functions lead to a variance that is slightly too high at the goal positions.

5.2 Learning Based on Samples

This section covers the more realistic scenario where the policy and the resulting joint distribution of the expert are not known.
The experiments are based on the same system as the one described in section 5.1. The empirical feature counts and the target distribution respectively are estimated based on demonstrations. All experiments use the same twenty sets of seven demonstrations.

5.2.1 IRL by Matching State Distributions

The employed reward function does not depend on the velocities directly. As the velocities are usually very noisy, it is sensible to exclude them from the target distribution as described in chapter 4.2.4. The effect of removing the velocities from the target distribution was tested both, for MSD-ACL as well as for MSD-REG.
The version of MSD-ACL that tries to match the complete state distribution was initialized with $\sigma_0^2 = 10$; the version that ignores the velocities uses $\sigma_0^2 = 1$. These values were chosen based on preliminary experiments. The averaged expected rewards and their 2σ confidence intervals are shown in Figure 5.7. The variant that does not try to match the velocities converges slower, but seems to converge to a better solution.
Both variants of MSD-REG have been tested with $\sigma_0^2 = 10000$. Similar to the experiments based on the known expert distribution, the averaged expected rewards did not change much after the first iteration. The results of the optimization are therefore shown in a table (5.2). By ignoring the noisy velocities, the performance could be improved.

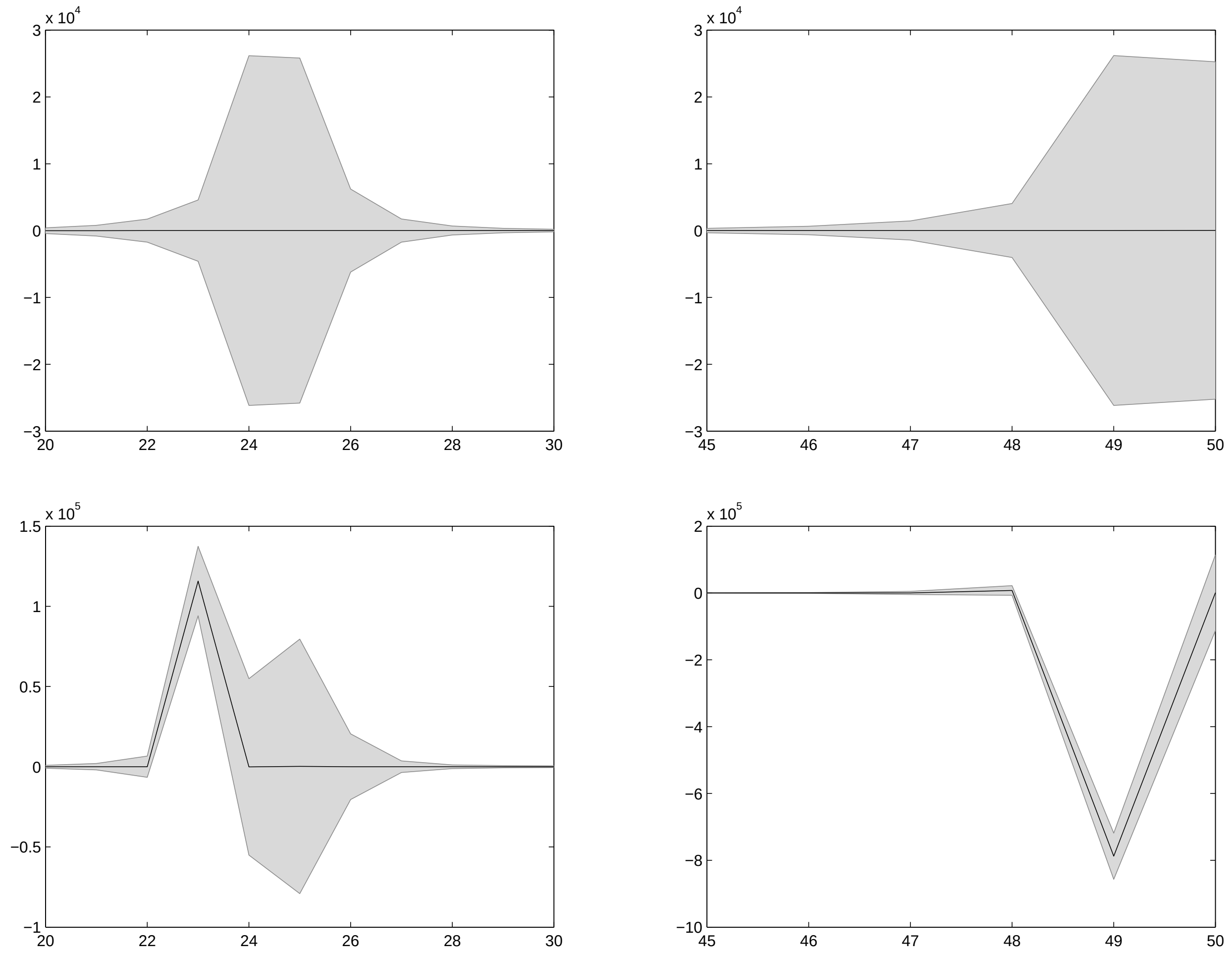

Figure 5.6.: The plots show the learned goal positions and their associated rewards for the interesting time steps at the middle and end of the demonstration. The top row illustrates the reward parameters learned by MSD-REG. The bottom row shows those learned by MaxCausalEnt-IRL. Both approaches associate high reward to the critical time steps 25 and 50. The learned goal positions by MaxCausalEnt-IRL are very different from zero at the time steps that precede the critical ones. The goal positions are shown in Table 5.1.

variant	averaged reward	2σ-confidence
matching velocities	-90884	± 24974
ignoring velocities	-69314	± 10583
expert demonstrations	-67418	± 26437

Table 5.2.: The expected averaged reward is shown after thousand iterations. The performance of MSD-REG is similar to the one demonstrated by the expert when the velocities are ignored.

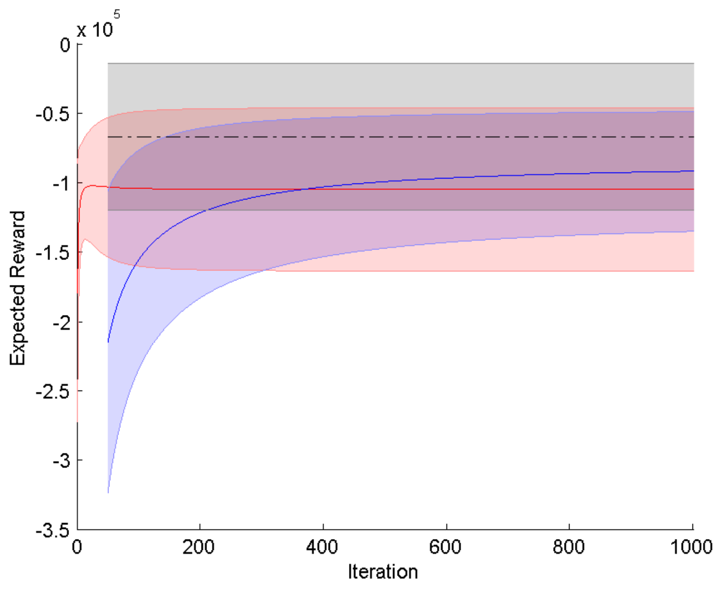

Figure 5.7.: The red curve shows the averaged expected rewards of MSD-ACL when matching both, positions and velocities. The blue curve shows averaged expected rewards of MSD-ACL when the velocities are ignored. The average expected rewards of the demonstrations is illustrated by the black line. The shaded areas correspond to 2σ confidence. The variant that ignores the velocities converges slower. However it seems to converge to a better solution. The slower convergence can be explained with the higher initial regularization.

time step	23	24	25	26	27	48	49	50
actual	0	0	10	0	0	0	0	20
MSD-REG	112.1	5.1	10.1015	4.6	111.9	-11.3	-119.1	20.7220
MaxCausalEnt-IRL	1.2E4	31.9	13.4	65.8	41.2	3.6	35.5	43.1

Table 5.3.: The table compares the goal positions around the critical time steps for the first dimension in task space. MSD-REG approximately learns the true goal positions at the critical time steps (25 and 50).

5.2.2 Comparison with Maximum Causal Entropy IRL

Figure 5.8 compares the expected rewards of MSD-ACL and MaxCausalEnt-IRL. Both approaches try to match all states. MSD-ACL is initialized with $\sigma_0^2 = 10$. As for the setting where the expert state distribution is known, MSD-ACL converges significantly faster.
Again, the reward parameters learned by MSD-REG are compared to those learned by MaxCausalEnt-IRL. Figure 5.9 shows the goal positions for the first dimension in task space as well as the associated rewards for time steps that are close to the critical ones.

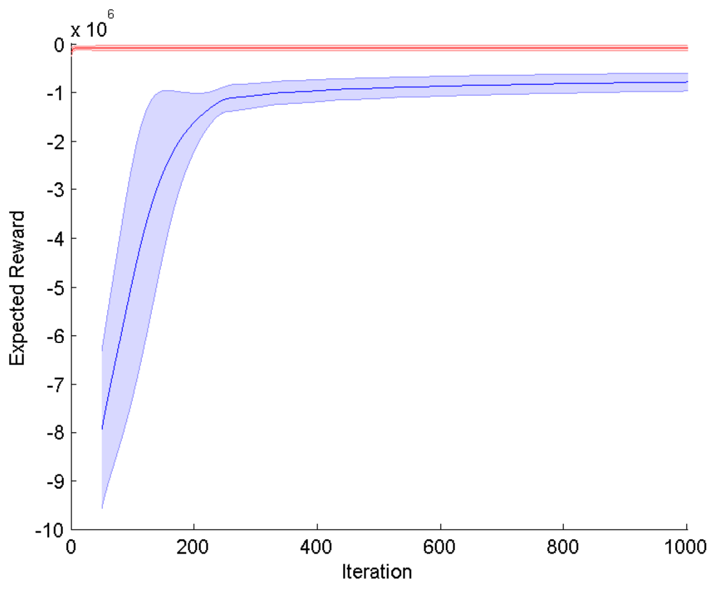

Figure 5.8.: MaxCausalEnt-IRL (blue curve) is compared to MSD-ACL (red curve) on the same sets of 7 samples. MSD-ACL converges significantly faster.

5.3 Discussion

A simple toy task was used to evaluate both variants of IRL-MSD and to compare them with MaxCausalEnt-IRL. Both, MSD-ACL and MSD-REG converged significantly faster than MaxCausalEnt-IRL. This is not surprising, given that IRL-MSD does not rely on gradient descent but directly computes the reward parameters that are optimal with respect to the current estimations.
When properly initialized, both variants of IRL-MSD seem to converge to similar solutions. MSD-REG converges almost instantly. The expected reward based on the initial estimates of the parameters was always close to the best one found over all iterations. However, the initialization has a huge impact on the performance (Figure 5.2 left) because it corresponds to a regularization. MSD-ACL updates the target policy after each iteration and, thus, the initial parameter is less influential (Figure 5.2 right).
The parameters of the reward function have been inspected in order to assess how well the true goal positions of the expert have been matched. MSD-REG was able to approximately recover the true goal positions at the important time steps even for the sample-based estimate of the target state distribution.

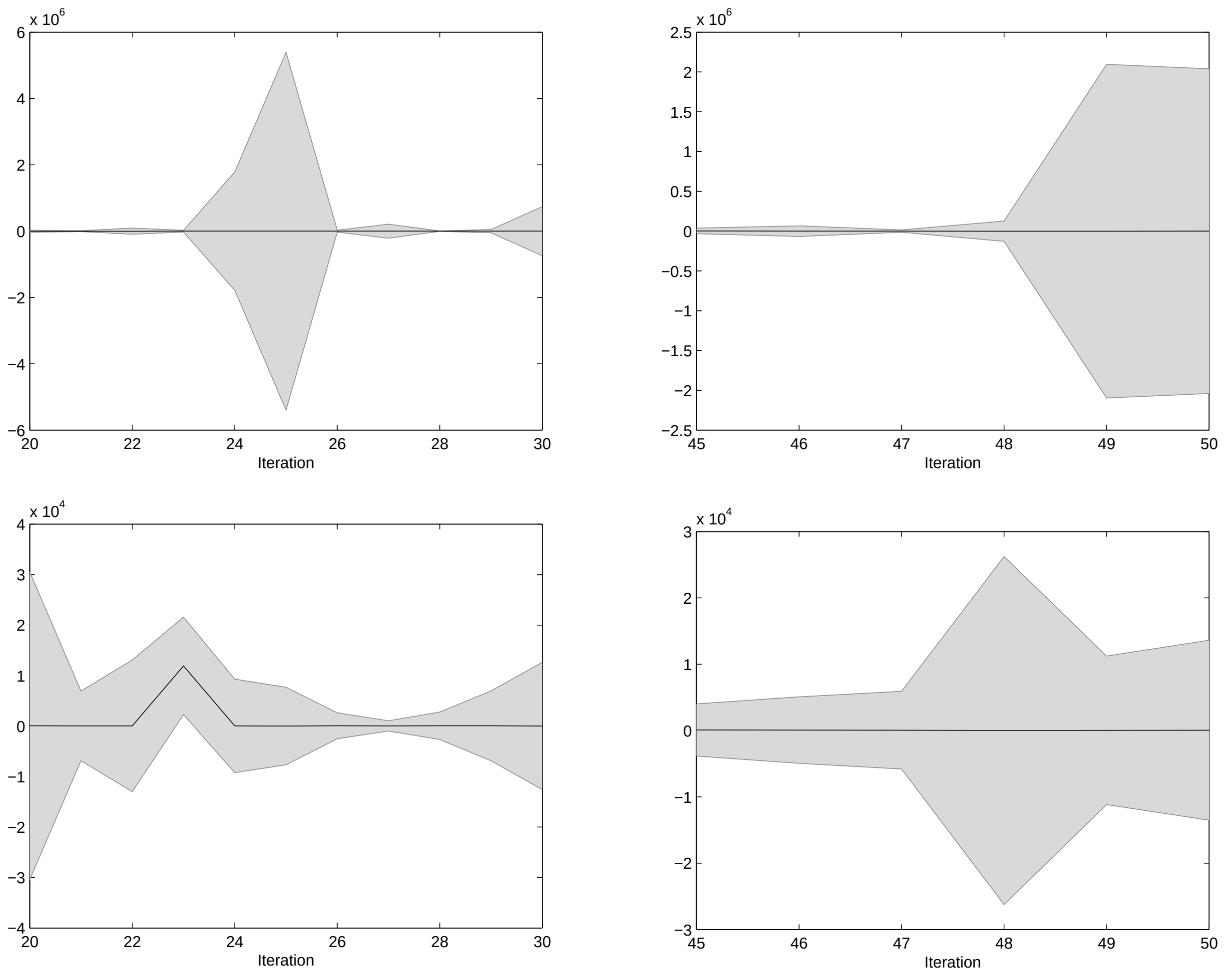

Figure 5.9.: The goal positions of the first dimension in task space and corresponding reward coefficient are shown for time steps close to the critical ones. The upper corresponds to MSD-REG and the bottom row corresponds to MaxCausalEnt-IRL. MSD-REG captures the sparseness of the true reward function better that MaxCausalEnt-IRL. The goal positions are shown in Table 5.3.

6 Future Work

Due to the good results in the experiments, IRL-MSD suggests itself for future research. This chapter discusses some open problems and how they might be addressed.

6.1 Learning Context Dependent Reward Functions

The goal state for a given time step t may depend on a context that is not modeled via the system dynamics. For example, the control of the racket for the task of playing table tennis may depend on the last observation of the ball. Therefore, the learned reward function should also be a function of that observation. IRL-MSD can be applied straightforwardly to learn such reward functions by adjusting the corresponding KL of the optimization goal (4.1a). For example, a conditional distribution $q_t(\mathbf{s}|\mathbf{o})$ could serve as a target state distribution that depends on the last observation $\mathbf{o}$. The parameters of the mean $\mu_{q,t} = \mathbf{K}_{q,t}\mathbf{o} + \mathbf{k}_{q,t}$ could be learned from the demonstrations, for example by linear regression.
By modifying the optimization goal similarly as shown in chapter 4.2.4, but using conditional distributions instead of marginals, the linear terms of the reward functions would become context dependent. More generally, it might also be useful to choose parametrized distributions.

6.2 Dropping the LQG Assumption

Only LQG systems have been considered within this thesis. As these assumptions rarely hold for real robot applications, ways have to be found to deal with the resulting implications. If the involved distributions are no longer Gaussian, both, the backward pass as well as the forward pass could no longer be solved in closed form. However, this problem is not specific to IRL-MSD but concerns other IRL methods as well. Methods for coping with non-LQG systems often include approximations. Local IRL methods (Levine and Koltun, 2012) apply Laplace approximations to approximate the policy by a Gaussian distribution along the demonstrated path. If the changes of the policy are sufficiently small, linear approximations of the system dynamics may provide satisfying results.
The reward function given in Equation 4.3 does not require the distributions to be Gaussian and, thus, may provide some insights.

6.3 Bounding the Policy-KL

The formulation of the optimization problem given in Equation 4.1a minimizes the KL between the policy and the target policy as part of the objective function. However, similarly to REPS, an inequality constraint could be introduced to bound the information loss between iterations. This could be especially useful when using linear approximations of the dynamics.

6.4 Learning State Independent Action Costs

MSD-REG does not learn any action costs at all whereas MSD-ACL learns quadratic action costs, but also state-dependent and linear ones (corresponding to non-zero action goal positions). It would be interesting to only learn the quadratic costs while keeping the linear and state depend costs equal to zero. It seems like this could be achieved by using the variance of the last policy as target policy, while keeping the target mean equal to zero. This corresponds to a hybrid of MSD-REG and MSD-ACL. However, quick tests did not produce meaningful learning.

7 Conclusion

An observed behavior can not always be explained by underlying a non-stationary policy. Even if it can be explained, the non-stationary policy may lead to increased complexity. This thesis investigated how to efficiently learn a time-dependent reward function from demonstrations. Furthermore, it was analyzed how well the actual goals could be recovered.

Two different approaches have been employed, namely MaxCausalEnt-IRL and IRL-MSD. Striving for an efficient implementation, the dual function corresponding to the MaxCausalEnt-IRL optimization problem was derived. The evaluation of the dual function can be used to adapt the step size during gradient descent. By exploring the difference between the actual Value function and the softmax Value function, it was shown that the dual function can be evaluated with negligible overhead.

IRL-MSD is a novel approach that has been developed as a part of this thesis. Inverse Reinforcement Learning by Matching State Distributions aims to minimize the relative entropy between the state distribution that results from the learned policy and the observed state distribution. This formulation does not require a constraint for matching feature counts. Therefore, in contrast to MaxCausalEnt-IRL, the corresponding optimization problem does not possess a Lagrangian multiplier $\boldsymbol{\theta}$ and no gradient-based optimization is necessary. Instead, the reward parameters are expressed directly in terms of the learned state distribution.

Two different variants of IRL-MSD have been presented, MSD-ACL and MSD-REG. MSD-ACL learns state-dependent action costs by updating the target policy for each iteration. Similar to REPS, the target policy is set to the policy of the previous iteration. The state-dependent action costs make it harder to interpret the learned reward function, but enable the algorithm to converge to a better solution than MSD-REG while being less sensitive to the initialization. MSD-REG treats the target policy as regularization and thereby learns only a state-dependent reward function.

Both variants have been compared to MaxCausalEnt-IRL on a toy task. Even though the task was relatively simple, it served well in making the advantage of dropping the necessity of gradient descent evident. Both variants of IRL-MSD converged significantly faster than MaxCausalEnt-IRL during all conducted experiments. MSD-REG converges almost instantly and changes only slightly after the first iteration. Interestingly, with good initialization, it was able to perform similar to MSD-ACL. Furthermore, by inspecting the learned parameters of the reward function, it could be shown that MSD-REG is able to infer the true goals of the expert for the important time steps.

As these results are very promising, the method should be further investigated. Slight reformulations of the optimization problem can be used to (1) bound the relative entropy of the policies similar to REPS, (2) learn context dependent reward functions or (3) learn state independent action costs. The arguably most interesting next step is to evaluate IRL-MSD on a real robot. Within this thesis it was only applied to a simple toy task which was accomplished convincingly.

Can it master the challenges of a real robot application?

References

Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

Pieter Abbeel, Adam Coates, and Andrew Y. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *Int. J. Rob. Res.*, 29(13):1608–1639, November 2010. ISSN 0278-3649.

Aris Alissandrakis, Chrystopher L Nehaniv, and Kerstin Dautenhahn. Imitation with alice: Learning to imitate corresponding actions across dissimilar embodiments. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 32(4):482–496, 2002.

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

C. G. Atkeson and S. Schaal. Robot learning from demonstration. In *Machine Learning: Proceedings of the Fourteenth International Conference (ICML '97)*, pages 12–20. morgan kaufmann, 1997. URL `http://www-clmc.usc.edu/publications/A/atkeson-ICML1997.pdf`.

Michael Bain and Claude Sammut. A framework for behavioural cloning. *Machine Intelligence*, 5, 1995.

Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957. URL `http://books.google.com/books?id=fyVtp3EMxasC&pg=PR5&dq=dynamic+programming+richard+e+bellman&client=firefox-a#v=onepage&q=dynamic%20programming%20richard%20e%20bellman&f=false`.

Abdeslam Boularias, Jens Kober, and Jan R Peters. Relative entropy inverse reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 182–189, 2011.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2009.

Allen Cypher and Daniel Conrad Halbert. *Watch what I do: programming by demonstration*. MIT press, 1993.

Walter R Gilks. *Markov chain monte carlo*. Wiley Online Library, 2005.

Peter D Grünwald and A Philip Dawid. Game theory, maximum entropy, minimum discrepancy and robust bayesian decision theory. *Annals of Statistics*, pages 1367–1433, 2004.

Florent Guenter, Micha Hersch, Sylvain Calinon, and Aude Billard. Reinforcement learning for imitating constrained reaching movements. *Advanced Robotics*, 21(13):1521–1544, 2007.

Daniel Conrad Halbert. *Programming by example*. PhD thesis, University of California, Berkeley, 1984.

A. Ijspeert, J. Nakanishi, and S. Schaal. Learning attractor landscapes for learning motor primitives. In *Advances in Neural Information Processing Systems 15*, pages 1547–1554. cambridge, ma: mit press, 2003. URL `http://www-clmc.usc.edu/publications/I/ijspeert-NIPS2002.pdf`.

J. A. Ijspeert, J. Nakanishi, and S. Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *International Conference on Robotics and Automation (ICRA2002)*, 2002. URL `http://www-clmc.usc.edu/publications/I/ijspeert-ICRA2002.pdf`.

Edwin T Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.

Seungsu Kim, Chang Hwan Kim, Bumjae You, and Sangrok Oh. Stable whole-body motion generation for humanoid robots to imitate human motions. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 2518–2524. IEEE, 2009.

Jens Kober and Jan R. Peters. Policy search for motor primitives in robotics. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 849–856. Curran Associates, Inc., 2009. URL `http://papers.nips.cc/paper/3545-policy-search-for-motor-primitives-in-robotics.pdf`.

Nate Kohl and Peter Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 3, pages 2619–2624. IEEE, 2004.

Petar Kormushev, Sylvain Calinon, and Darwin G Caldwell. Robot motor skill coordination with em-based reinforcement learning. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3232–3237. IEEE, 2010.

Dana Kulić, Christian Ott, Dongheui Lee, Junichi Ishikawa, and Yoshihiko Nakamura. Incremental learning of full body motion primitives and their sequencing through human motion observation. *The International Journal of Robotics Research*, page 0278364911426178, 2011.

Yasuo Kuniyoshi, Masayuki Inaba, and Hirochika Inoue. Learning by watching: Extracting reusable task knowledge from visual observation of human performance. *IEEE Transactions on Robotics and Automation*, 10:799–822, 1994.

Sergey Levine and Vladlen Koltun. Continuous inverse optimal control with locally optimal examples. *arXiv preprint arXiv:1206.4617*, 2012.

Sergey Levine, Zoran Popovic, and Vladlen Koltun. Nonlinear inverse reinforcement learning with gaussian processes. In *Advances in Neural Information Processing Systems*, pages 19–27, 2011.

Tomas Lozano-Perez. Robot programming. *Proceedings of the IEEE*, 71(7):821–841, 1983.

G.J. Maeda, M. Ewerton, R. Lioutikov, H.B. Amor, J. Peters, and G. Neumann. Learning interaction for collaborative tasks with probabilistic movement primitives. In *Proceedings of the International Conference on Humanoid Robots (HUMANOIDS)*, pages 527–534, 2014. URL `http://www.ausy.tu-darmstadt.de/uploads/Team/PubGJMaeda/maeda2014InteractionProMP_HUMANOIDS.pdf`.

D Michie, M Bain, and J Hayes-Miches. Cognitive models from subcognitive skills. *IEE control engineering series*, 44:71–99, 1990.

Hiroyuki Miyamoto, Stefan Schaal, Francesca Gandolfo, Hiroaki Gomi, Yasuharu Koike, Rieko Osu, Eri Nakano, Yasuhiro Wada, and Mitsuo Kawato. A kendama learning robot based on bi-directional theory. *Neural networks*, 9(8):1281–1302, 1996.

Chrystopher L Nehaniv and Kerstin Dautenhahn. *Imitation in animals and artifacts*. MIT Press, 2002.

Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, pages 663–670, 2000.

A. Paraschos, C. Daniel, J. Peters, and G Neumann. Probabilistic movement primitives. In *Advances in Neural Information Processing Systems (NIPS), Cambridge, MA: MIT Press.*, 2013. URL `http://www.ias.tu-darmstadt.de/uploads/Publications/Paraschos_NIPS_2013a.pdf`.

Jan Peters, Katharina Mülling, and Yasemin Altun. Relative entropy policy search. In *AAAI*, 2010.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. Technical report, DTIC Document, 1989.

Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. *Urbana*, 51:61801, 2007.

Claude Sammut, Scott Hurst, Dana Kedzier, and Donald Michie. Learning to fly. *Imitation in animals and artifacts*, page 171, 2002.

S. Schaal. Learning from demonstration. In *Advances in Neural Information Processing Systems 9*, pages 1040–1046. mit press, 1997. URL `http://www-clmc.usc.edu/publications/S/schaal-NIPS1997.pdf`.

S. Schaal, S. Kotosaka, and D. Sternad. Nonlinear dynamical systems as movement primitives. In *Humanoids2000, First IEEE-RAS International Conference on Humanoid Robots*. cd-proceedings, 2000. URL `http://www-slab.usc.edu/publications/S/schaal-ICHR2000.pdf`.

S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Control, planning, learning, and imitation with dynamic movement primitives. In *Workshop on Bilateral Paradigms on Humans and Humanoids, IEEE International Conference on Intelligent Robots and Systems (IROS 2003)*, 2003. URL `http://www-clmc.usc.edu/publications/S/schaal-IROS2003.pdf`.

S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Learning movement primitives. In *International Symposium on Robotics Research (ISRR2003)*. springer, 2004. URL `http://www-clmc.usc.edu/publications/S/schaal-ISRR2003.pdf`.

Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6): 233–242, 1999.

Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 1998.

Michael Tomasello, Ann Cale Kruger, and Hilary Horn Ratner. Cultural learning. *Behavioral and brain sciences*, 16(03):495–511, 1993.

Aleš Ude, Christopher G Atkeson, and Marcia Riley. Programming full-body movements for humanoid robots by observation. *Robotics and autonomous systems*, 47(2):93–108, 2004.

Max A Woodbury. Inverting modified matrices. *Memorandum report*, 42:106, 1950.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, pages 1433–1438, 2008.

Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. Modeling interaction via the principle of maximum causal entropy. *In ICML 2010, Haifa, Israel*, 2010.

A Derivations for Maximum Causal Entropy Inverse Reinforcement Learning

$$\begin{aligned}
\underset{\pi_t(\mathbf{a}|\mathbf{s})}{\text{maximize}} \sum_{t=1}^{T-1} \int_{\mathbf{s},\mathbf{a}} & -p_t(\mathbf{s})\pi_t(\mathbf{a}|\mathbf{s}) \log \pi_t(\mathbf{a}|\mathbf{s})\, d\mathbf{s} d\mathbf{a} \\
\text{subject to} \sum_{t=1}^{T-1} \int_{\mathbf{s},\mathbf{a}} & p_t(\mathbf{s})\pi_t(\mathbf{a}|\mathbf{s})\boldsymbol{\phi}_t(\mathbf{s},\mathbf{a}) d\mathbf{s} d\mathbf{a} + \int_{\mathbf{s}} p_T(\mathbf{s})\boldsymbol{\phi}_T(\mathbf{s},\mathbf{0})\, d\mathbf{s} = \hat{\boldsymbol{\phi}}, \\
& \forall_{t>1} \forall_{\mathbf{s}'} \int_{\mathbf{a}} p_t(\mathbf{s}')\pi_t(\mathbf{a}|\mathbf{s}')\, d\mathbf{a} = \int_{\mathbf{s},\mathbf{a}} p_{t-1}(\mathbf{s})\pi_{t-1}(\mathbf{a}|\mathbf{s}) p(\mathbf{s}'|\mathbf{s},\mathbf{a})\, d\mathbf{s} d\mathbf{a}, \\
& \forall_{\mathbf{s}}\, p_1(\mathbf{s}) = \mu_1(\mathbf{s}), \\
& \forall_{t<T} \forall_{\mathbf{s}} \int_{a} \pi_t(\mathbf{a}|\mathbf{s}) d\mathbf{a} = 1,
\end{aligned}$$

Lagrangian:

$$\begin{aligned}
\mathcal{L}(p_t(\mathbf{s}), \pi_t(\mathbf{a}|\mathbf{s}), \boldsymbol{\theta}, \lambda, V_t(\mathbf{s})) = & -\sum_{t=1}^{T-1} \int_{\mathbf{s},\mathbf{a}} p_t(\mathbf{s})\pi_t(\mathbf{a}|\mathbf{s}) \log \pi_t(\mathbf{a}|\mathbf{s})\, d\mathbf{s}\, d\mathbf{a} \\
& + \boldsymbol{\theta}^\top \left(\hat{\boldsymbol{\phi}} - \sum_{t=1}^{T-1} \int_{\mathbf{s},\mathbf{a}} p_t(\mathbf{s})\pi_t(\mathbf{a}|\mathbf{s})\boldsymbol{\phi}_t(\mathbf{s},\mathbf{a})\, d\mathbf{s}\, d\mathbf{a} - \int_{\mathbf{s}} p_T(\mathbf{s})\boldsymbol{\phi}_T(\mathbf{s},0)\, d\mathbf{s} \right) \\
& + \sum_{t=1}^{T-1} \int_{\mathbf{s}} \lambda_t(\mathbf{s}) \left(1 - \int_{\mathbf{a}} \pi_t(\mathbf{a}|\mathbf{s})\, d\mathbf{a} \right) d\mathbf{s} \\
& + \sum_{t=2}^{T} \int_{\mathbf{s}'} V_t(\mathbf{s}') \left(\int_{\mathbf{s},\mathbf{a}} p_{t-1}(\mathbf{s})\pi_{t-1}(\mathbf{a}|\mathbf{s}) p(\mathbf{s}'|\mathbf{s},\mathbf{a})\, d\mathbf{a}\, d\mathbf{s} - \int_{a} p_t(\mathbf{s}')\pi_t(\mathbf{a}|\mathbf{s}')\, d\mathbf{a} \right) d\mathbf{s}' \\
& + \int_{\mathbf{s}} V_1(\mathbf{s}) (\mu_1(\mathbf{s}) - p_1(\mathbf{s}))\, d\mathbf{s}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \mathcal{L}(p_t(\mathbf{s}), \pi_t(\mathbf{a}|\mathbf{s}), \boldsymbol{\theta}, \lambda, V_t(\mathbf{s})}{\partial \pi_t(\mathbf{a}|\mathbf{s})} = & -p_t(\mathbf{s}) \log \pi_t(\mathbf{a}|\mathbf{s}) - p_t(\mathbf{s}) - \boldsymbol{\theta}^\top p_t(\mathbf{s})\boldsymbol{\phi}_t(\mathbf{s},\mathbf{a}) - \lambda_t(\mathbf{s}) - \mathbb{1}_{t>1} V_t(\mathbf{s}) p_t(\mathbf{s}) \\
& + \int_{\mathbf{s}'} V_{t+1}(\mathbf{s}') p_t(\mathbf{s}) p(\mathbf{s}'|\mathbf{s},\mathbf{a})\, d\mathbf{s}' \overset{!}{=} 0 \\
\Rightarrow \pi_t(\mathbf{a}|\mathbf{s}) = & \exp\left(-1 - \boldsymbol{\theta}^\top \boldsymbol{\phi}_t(\mathbf{s},\mathbf{a}) - \frac{\lambda_t(\mathbf{s})}{p_t(\mathbf{s})} - \mathbb{1}_{t>1} V_t(\mathbf{s}) + \int_{\mathbf{s}'} V_{t+1}(\mathbf{s}') p(\mathbf{s}'|\mathbf{s},\mathbf{a})\, d\mathbf{s}' \right) \\
\propto & \exp\left(-\boldsymbol{\theta}^\top \boldsymbol{\phi}_t(\mathbf{s},\mathbf{a}) + \int_{\mathbf{s}'} V_{t+1}(\mathbf{s}') p(\mathbf{s}'|\mathbf{s},\mathbf{a})\, d\mathbf{s}' \right)
\end{aligned}$$

Dual Problem:

$$\begin{aligned}\mathcal{G}(p_t(\mathbf{s}),\boldsymbol{\theta},\lambda,V_t(\mathbf{s})) = &-\sum_{t=1}^{T-1}\int_{\mathbf{s},\mathbf{a}} p_t(\mathbf{s})\pi_t(\mathbf{a}|\mathbf{s})\left(-1-\boldsymbol{\theta}^\top\boldsymbol{\phi}_t(\mathbf{s},\mathbf{a})-\frac{\lambda_t(\mathbf{s})}{p_t(\mathbf{s})}-\mathbb{1}_{t>1}V_t(\mathbf{s})+\int_{\mathbf{s}'}V_{t+1}(\mathbf{s}')p(\mathbf{s}'|\mathbf{s},\mathbf{a})\,d\mathbf{s}'\right)d\mathbf{s}\,d\mathbf{a}\\ &+\boldsymbol{\theta}^\top\left(\hat{\boldsymbol{\phi}}-\sum_{t=1}^{T-1}\int_{\mathbf{s},\mathbf{a}}p_t(\mathbf{s})\pi_t(\mathbf{a}|\mathbf{s})\boldsymbol{\phi}_t(\mathbf{s},\mathbf{a})\,d\mathbf{s}\,d\mathbf{a}-\int_{\mathbf{s}}p_T(\mathbf{s})\boldsymbol{\phi}_T(\mathbf{s},0)\,d\mathbf{s}\right) \qquad \text{(A.2)}\\ &+\sum_{t=1}^{T-1}\int_{\mathbf{s}}\lambda_t(\mathbf{s})\left(1-\int_{\mathbf{a}}\pi_t(\mathbf{a}|\mathbf{s})\,d\mathbf{a}\right)d\mathbf{s}\\ &+\sum_{t=2}^{T}\int_{\mathbf{s}'}V_t(\mathbf{s}')\left(\int_{\mathbf{s},\mathbf{a}}p_{t-1}(\mathbf{s})\pi_{t-1}(\mathbf{a}|\mathbf{s})p(\mathbf{s}'|\mathbf{s},\mathbf{a})\,d\mathbf{s}\,d\mathbf{a}-\int_{\mathbf{a}}p_t(\mathbf{s}')\pi_t(\mathbf{a}|\mathbf{s}')\,d\mathbf{a}\right)d\mathbf{s}'\\ &+\int_{\mathbf{s}}V_1(\mathbf{s})\,(\mu_1(\mathbf{s})-p_1(\mathbf{s}))\,d\mathbf{s}\\ =&\ T-1+\boldsymbol{\theta}^\top\hat{\boldsymbol{\phi}}-\boldsymbol{\theta}^\top\int_{\mathbf{s}}p_T(\mathbf{s})\boldsymbol{\phi}_T(\mathbf{s},0)\,d\mathbf{s}+\sum_{t=1}^{T-1}\int_{\mathbf{s}}\lambda_t(\mathbf{s})\,d\mathbf{s}+\int_{\mathbf{s}}V_1(\mathbf{s})\,(\mu_1(\mathbf{s})-p_1(\mathbf{s}))\,d\mathbf{s}-\int_{\mathbf{s}}p_T(\mathbf{s})V_T(\mathbf{s})\,d\mathbf{s} \qquad \text{(A.3)}\end{aligned}$$

Removing $\lambda_t(\mathbf{s})$:

$$\int_{\mathbf{a}}\exp\left(-1-\boldsymbol{\theta}^\top\boldsymbol{\phi}_t(\mathbf{s},\mathbf{a})-\frac{\lambda_t(\mathbf{s})}{p_t(\mathbf{s})}-\mathbb{1}_{t>1}V_t(\mathbf{s})+\int_{\mathbf{s}'}V_{t+1}(\mathbf{s}')p(\mathbf{s}'|\mathbf{s},\mathbf{a})\,d\mathbf{s}'\right)d\mathbf{a}=1$$

$$\exp\left(\frac{\lambda_t(\mathbf{s})}{p_t(\mathbf{s})}+1+\mathbb{1}_{t>1}V_t(\mathbf{s})\right)=\int_{\mathbf{a}}\exp\left(-\boldsymbol{\theta}^\top\boldsymbol{\phi}_t(\mathbf{s},\mathbf{a})+\int_{\mathbf{s}'}V_{t+1}(\mathbf{s}')p(\mathbf{s}'|\mathbf{s},\mathbf{a})\,d\mathbf{s}'\right)d\mathbf{a}$$

$$\Rightarrow \lambda_t(\mathbf{s})=p_t(s)\left(-1-\mathbb{1}_{t>1}V_t(\mathbf{s})+\log\int_{\mathbf{a}}\exp\left(-\boldsymbol{\theta}^\top\boldsymbol{\phi}_t(\mathbf{s},\mathbf{a})+\int_{\mathbf{s}'}V_{t+1}(\mathbf{s}')p(\mathbf{s}'|\mathbf{s},\mathbf{a})\,d\mathbf{s}'\right)d\mathbf{a}\right) \qquad \text{(A.4)}$$

Substituting (A.4) in (A.3):

$$\begin{aligned}\mathcal{G}(p_t(\mathbf{s}),\boldsymbol{\theta},V_t(\mathbf{s}))=&\ T-1+\boldsymbol{\theta}^\top\hat{\boldsymbol{\phi}}-\boldsymbol{\theta}^\top\int_{\mathbf{s}}p_T(\mathbf{s})\boldsymbol{\phi}_T(\mathbf{s},0)\,d\mathbf{s}+\int_{\mathbf{s}}V_1(\mathbf{s})\,(\mu_1(\mathbf{s})-p_1(\mathbf{s}))\,d\mathbf{s}-\int_{\mathbf{s}}p_T(\mathbf{s})V_T(\mathbf{s})\,d\mathbf{s}\\ &+\sum_{t=1}^{T-1}\int_{\mathbf{s}}p_t(s)\left(-1-\mathbb{1}_{t>1}V_t(\mathbf{s})+\log\int_{\mathbf{a}}\exp\left(-\boldsymbol{\theta}^\top\boldsymbol{\phi}_t(\mathbf{s},\mathbf{a})+\int_{\mathbf{s}'}V_{t+1}(\mathbf{s}')p(\mathbf{s}'|\mathbf{s},\mathbf{a})\,d\mathbf{s}'\right)d\mathbf{a}\right)d\mathbf{s}\\ =&\ \boldsymbol{\theta}^\top\hat{\boldsymbol{\phi}}-\boldsymbol{\theta}^\top\int_{\mathbf{s}}p_T(\mathbf{s})\boldsymbol{\phi}_T(\mathbf{s},0)\,d\mathbf{s}+\int_{\mathbf{s}}V_1(\mathbf{s})\mu_1(\mathbf{s})\,d\mathbf{s}-\int_{\mathbf{s}}p_T(\mathbf{s})V_T(\mathbf{s})\,d\mathbf{s}\\ &+\sum_{t=1}^{T-1}\int_{\mathbf{s}}p_t(s)\left(-V_t(\mathbf{s})+\log\int_{\mathbf{a}}\exp\left(-\boldsymbol{\theta}^\top\boldsymbol{\phi}_t(\mathbf{s},\mathbf{a})+\int_{\mathbf{s}'}V_{t+1}(\mathbf{s}')p(\mathbf{s}'|\mathbf{s},\mathbf{a})\,d\mathbf{s}'\right)d\mathbf{a}\right)d\mathbf{s}\end{aligned}$$

Partial Derivatives:

$$\frac{\partial \mathscr{G}(p_t(\mathbf{s}), \boldsymbol{\theta}, V_t(\mathbf{s}))}{\partial \boldsymbol{\theta}} = \hat{\boldsymbol{\phi}} - \sum_{t=1}^{T-1} \int\limits_{\mathbf{s},\mathbf{a}} p_t(\mathbf{s}) \frac{\exp\left(-\boldsymbol{\theta}^\top \boldsymbol{\phi}_t(\mathbf{s},\mathbf{a}) + \int\limits_{\mathbf{s}'} V_{t+1}(\mathbf{s}')p(\mathbf{s}'|\mathbf{s},\mathbf{a})\, d\mathbf{s}'\right)}{\int\limits_{\mathbf{a}} \exp\left(-\boldsymbol{\theta}^\top \boldsymbol{\phi}_t(\mathbf{s},\mathbf{a}) + \int\limits_{\mathbf{s}'} V_{t+1}(\mathbf{s}')p(\mathbf{s}'|\mathbf{s},\mathbf{a})\, d\mathbf{s}'\right) d\mathbf{a}} \boldsymbol{\phi}(\mathbf{s},\mathbf{a})\, d\mathbf{s}\, d\mathbf{a} - \int\limits_{\mathbf{s}} p_T(\mathbf{s}) \boldsymbol{\phi}_T(\mathbf{s}, 0)\, d\mathbf{s}$$

$$= \hat{\boldsymbol{\phi}} - \sum_{t=1}^{T-1} \mathbb{E}_{p_t(\mathbf{s},\mathbf{a})}[\boldsymbol{\phi}(\mathbf{s},\mathbf{a})] - \mathbb{E}_{p_T(\mathbf{s})}[\boldsymbol{\phi}(\mathbf{s},\mathbf{0})] = \hat{\boldsymbol{\phi}} - \tilde{\boldsymbol{\phi}}$$

$$\frac{\partial \mathscr{G}(p_t(\mathbf{s}), \boldsymbol{\theta}, V_t(\mathbf{s}))}{\partial V_t(\mathbf{s})} = \begin{cases} -p_t(\mathbf{s}) + \mu_1 & \text{, if } t = 1 \\ -p_t(\mathbf{s}) + \int\limits_{\tilde{\mathbf{s}},\mathbf{a}} p_{t-1}(\tilde{\mathbf{s}}) \frac{\exp\left(-\boldsymbol{\theta}^\top \boldsymbol{\phi}(\tilde{\mathbf{s}},\mathbf{a}) + \int\limits_{\mathbf{s}'} V_t(\mathbf{s}')p(\mathbf{s}'|\tilde{\mathbf{s}},\mathbf{a})\, d\mathbf{s}'\right)}{\int\limits_{\mathbf{a}} \exp\left(-\boldsymbol{\theta}^\top \boldsymbol{\phi}(\tilde{\mathbf{s}},\mathbf{a}) + \int\limits_{\mathbf{s}'} V_t(\mathbf{s}')p(\mathbf{s}'|\tilde{\mathbf{s}},\mathbf{a})\, d\mathbf{s}'\right) d\mathbf{a}} p(\mathbf{s}|\tilde{\mathbf{s}},\mathbf{a})\, d\tilde{\mathbf{s}} & \text{, if } t > T \end{cases}$$

$$= \begin{cases} -p_1(\mathbf{s}) + \mu_1(\mathbf{s}) & \text{, if } t = 1 \\ -p_t(\mathbf{s}) + \int_{\mathbf{s},\mathbf{a}} \pi_{t-1}(\mathbf{a}|\mathbf{s}) p_{t-1}(\mathbf{s}) p(\mathbf{s}'|\mathbf{s},\mathbf{a}) & \text{, if } t > 1 \end{cases}$$

$$\frac{\partial \mathscr{G}(p_t(\mathbf{s}), \boldsymbol{\theta}, V_t(\mathbf{s}))}{\partial p_t(\mathbf{s})} = \begin{cases} -V_t(\mathbf{s}) + \log \int\limits_{\mathbf{a}} \exp\left(-\boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s},\mathbf{a}) + \int\limits_{\mathbf{s}'} V_{t+1}(\mathbf{s}')p(\mathbf{s}'|\mathbf{s},\mathbf{a})\, d\mathbf{s}'\right) d\mathbf{a} & \text{, if } t < T \\ -V_T(\mathbf{s}) - \boldsymbol{\theta}^\top \boldsymbol{\phi}_T(s, \mathbf{0}) & \text{, if } t = T \end{cases}$$

B Derivations for Inverse Reinforcement Learning by Matching State Distributions

$$\begin{aligned}
\underset{\pi_t(\mathbf{a}|\mathbf{s})}{\text{minimize}} \quad & \sum_{t=1}^{T} \int_{\mathbf{s}} p_t(\mathbf{s}) \log \frac{p_t(\mathbf{s})}{q_t(\mathbf{s})} d\mathbf{s} + \alpha \sum_{t=1}^{T-1} \int_{\mathbf{s}} p_t(\mathbf{s}) \int_{a} \pi_t(\mathbf{a}|\mathbf{s}) \log \frac{\pi_t(\mathbf{a}|\mathbf{s})}{q_{\text{reg},t}(\mathbf{a}|\mathbf{s})}\, d\mathbf{s}\, d\mathbf{a} \\
\text{subject to} \quad & \forall_{t<T} \forall_{\mathbf{s}} \int_{a} \pi_t(\mathbf{a}|\mathbf{s}) d\mathbf{a} = 1, \\
& \forall_{t>1} \forall_{\mathbf{s}'} \int_{a} p_t(\mathbf{s}') \pi_t(\mathbf{a}|\mathbf{s}') d\mathbf{a} = \int_{\mathbf{s},\mathbf{a}} p_{t-1}(\mathbf{s}) \pi_{t-1}(\mathbf{a}|\mathbf{s}) p(\mathbf{s}'|\mathbf{s},\mathbf{a}) d\mathbf{s} d\mathbf{a}, \\
& \forall_{\mathbf{s}} p_1(\mathbf{s}) = \mu_1(\mathbf{s}).
\end{aligned}$$

Lagrangian:

$$\begin{aligned}
\mathscr{L}(p_t(\mathbf{s}), \pi_t(\mathbf{a}|\mathbf{s}), \lambda_t(\mathbf{s}), V_t(\mathbf{s})) = & \sum_{t=1}^{T} \int_{\mathbf{s}} p_t(\mathbf{s}) \log \frac{p_t(\mathbf{s})}{q_t(\mathbf{s})} d\mathbf{s} + \alpha \sum_{t=1}^{T-1} \int_{\mathbf{s}} p_t(\mathbf{s}) \int_{a} \pi_t(\mathbf{a}|\mathbf{s}) \log \frac{\pi_t(\mathbf{a}|\mathbf{s})}{q_{\text{reg},t}(\mathbf{a}|\mathbf{s})} d\mathbf{s} \\
& + \sum_{t=1}^{T-1} \int_{\mathbf{s}} \lambda_t(\mathbf{s}) \left(1 - \int_{\mathbf{a}} \pi_t(\mathbf{a}|\mathbf{s}) \right) \\
& + \sum_{t=2}^{T} \int_{\mathbf{s}'} V_t(\mathbf{s}') \left(\left(\int_{a} p_t(\mathbf{s}') \pi_t(\mathbf{a}|\mathbf{s}') - \int_{\mathbf{s},\mathbf{a}} p_{t-1}(\mathbf{s}) \pi_{t-1}(\mathbf{a}|\mathbf{s}) p(\mathbf{s}'|\mathbf{s},\mathbf{a}) \right) \right. \\
& + \int_{\mathbf{s}} V_1(\mathbf{s}) (p_1(\mathbf{s}) - \mu_1(\mathbf{s}))
\end{aligned}$$

Gradient of Lagrangian: $(t < T)$

$$\begin{aligned}
\frac{\partial \mathscr{L}(p_t(\mathbf{s}), \pi_t(\mathbf{a}|\mathbf{s}), \lambda_t(\mathbf{s}), V_t(\mathbf{s}))}{\partial \pi_t(\mathbf{a}|\mathbf{s})} &= \alpha p_t(\mathbf{s}) \log \frac{\pi_t(\mathbf{a}|\mathbf{s})}{q_{\text{reg},t}(\mathbf{a}|\mathbf{s})} + \alpha p_t(\mathbf{s}) - \lambda_t(\mathbf{s}) + \mathbb{1}_{t>1} p_t(\mathbf{s}) V_t(\mathbf{s}) - \int_{\mathbf{s}'} V_{t+1}(\mathbf{s}') p_t(\mathbf{s}) p(\mathbf{s}'|\mathbf{s},\mathbf{a}) \overset{!}{=} 0 \\
\Rightarrow \pi_t^\star(\mathbf{a}|\mathbf{s}) &= q_{\text{reg},t}(\mathbf{a}|\mathbf{s}) \exp\left(\frac{1}{\alpha} \left(-\alpha + \frac{\lambda_t(\mathbf{s})}{p_t(\mathbf{s})} - \mathbb{1}_{t>1} V_t(\mathbf{s}) + \int_{\mathbf{s}'} V_{t+1}(\mathbf{s}') p(\mathbf{s}'|\mathbf{s},\mathbf{a}) \right) \right) \qquad \text{(B.1)}
\end{aligned}$$

Elimination of λ:

$$\begin{aligned}
\int_{\mathbf{a}} \pi_t^\star(\mathbf{a}|\mathbf{s}) = \int_{\mathbf{a}} \exp\left(\frac{1}{\alpha} \left(\alpha \log q_{\text{reg},t}(\mathbf{a}|\mathbf{s}) - \alpha + \frac{\lambda_t(\mathbf{s})}{p_t(\mathbf{s})} - \mathbb{1}_{t>1} V_t(\mathbf{s}) + \int_{\mathbf{s}'} V_{t+1}(\mathbf{s}') p(\mathbf{s}'|\mathbf{s},\mathbf{a}) \right) \right) = 1 \\
\Rightarrow \exp\left(1 - \frac{\lambda_t(\mathbf{s})}{\alpha p_t(\mathbf{s})} + \mathbb{1}_{t>1} \frac{V_t(\mathbf{s})}{\alpha} \right) = \int_{\mathbf{a}} q_{\text{reg},t}(\mathbf{a}|\mathbf{s}) \exp\left(\frac{\int_{\mathbf{s}'} V_{t+1}(\mathbf{s}') p(\mathbf{s}'|\mathbf{s},\mathbf{a}) d\mathbf{s}'}{\alpha} \right) \\
\lambda_t(\mathbf{s}) = p_t(\mathbf{s}) \left(\alpha + \mathbb{1}_{t>1} V_t(\mathbf{s}) - \alpha \log \int_{a} q_{\text{reg},t}(\mathbf{a}|\mathbf{s}) \exp\left(\frac{\int_{\mathbf{s}'} V_{t+1}(\mathbf{s}') p(\mathbf{s}'|\mathbf{s},\mathbf{a}) d\mathbf{s}'}{\alpha} \right) \right) \qquad \text{(B.2)}
\end{aligned}$$

Dual (using (B.1) and (B.2)):

$$\begin{aligned}
\mathscr{G}(p_t(\mathbf{s}), V_t(\mathbf{s})) = &\sum_{t=1}^{T} \int_{\mathbf{s}} p_t(\mathbf{s}) \log \frac{p_t(\mathbf{s})}{q_t(\mathbf{s})} ds + \alpha \sum_{t=1}^{T-1} \int_{\mathbf{s}} p_t(\mathbf{s}) \int_{\mathbf{a}} \pi_t(\mathbf{a}|\mathbf{s}) \left(\frac{1}{\alpha} \left(-\alpha + \frac{\lambda_t(\mathbf{s})}{p_t(\mathbf{s})} - \mathbb{1}_{t>1} V_t(\mathbf{s}) + \int_{\mathbf{s}'} V_{t+1}(\mathbf{s}') p(\mathbf{s}'|\mathbf{s},\mathbf{a}) \right) \right) \\
&+ \sum_{t=1}^{T-1} \int_{\mathbf{s}} \lambda_t(\mathbf{s}) \left(1 - \int_{\mathbf{a}} \pi_t(\mathbf{a}|\mathbf{s}) \right) \\
&+ \sum_{t=2}^{T} \int_{\mathbf{s}'} V_t(\mathbf{s}') \left(\int_{a} p_t(\mathbf{s}') \pi_t(\mathbf{a}|\mathbf{s}') - \int_{\mathbf{s},\mathbf{a}} p_{t-1}(\mathbf{s}) \pi_{t-1}(\mathbf{a}|\mathbf{s}) p(\mathbf{s}'|\mathbf{s},\mathbf{a}) \right) \\
&+ \int_{\mathbf{s}} V_1(\mathbf{s}) (p_1(\mathbf{s}) - \mu_1(\mathbf{s})) \\
= &\sum_{t=1}^{T} \int_{\mathbf{s}} p_t(\mathbf{s}) \log \frac{p_t(\mathbf{s})}{q_t(\mathbf{s})} ds \\
&+ \sum_{t=1}^{T-1} \int_{\mathbf{s}} p_t(\mathbf{s}) \left(V_t(\mathbf{s}) - \alpha \log \int_{a} q_{\mathrm{reg},t}(\mathbf{a}|\mathbf{s}) \exp \left(\frac{\int_{\mathbf{s}'} V_{t+1}(\mathbf{s}') p(\mathbf{s}'|\mathbf{s},\mathbf{a}) d\mathbf{s}'}{\alpha} \right) \right) \\
&+ \int_{\mathbf{s}'} V_T(\mathbf{s}') \int_{a} p_T(\mathbf{s}') \pi_T(\mathbf{a}|\mathbf{s}') \\
&- \int_{\mathbf{s}} V_1(\mathbf{s}) \mu_1(\mathbf{s})
\end{aligned}$$

Gradients of Dual:

$$\begin{aligned}
\frac{\partial \mathscr{G}(p_t(\mathbf{s}), V_t(\mathbf{s}))}{p_t(\mathbf{s})} &= \begin{cases} V_T(\mathbf{s}) + \log \frac{p_t(\mathbf{s})}{q_t(\mathbf{s})} + 1 & \text{, if } t = T \\ V_t(\mathbf{s}) + \log \frac{p_t(\mathbf{s})}{q_t(\mathbf{s})} + 1 - \alpha \log \int_{\mathbf{a}} q_{\mathrm{reg},t}(\mathbf{a}|\mathbf{s}) \exp \left(\frac{1}{\alpha} \int_{\mathbf{s}'} V_{t+1}(\mathbf{s}') p(\mathbf{s}'|\mathbf{s},\mathbf{a}) d\mathbf{s}' \right) & \text{, if } t < T \end{cases} \\
&= \begin{cases} V_T(\mathbf{s}) + \log \frac{p_t(\mathbf{s})}{q_t(\mathbf{s})} + 1 & \text{, if } t = T \\ V_t(\mathbf{s}) - \alpha \log \int_{\mathbf{a}} \exp \left(\frac{1}{\alpha} \left(\alpha \log(q_{\mathrm{reg},t}(\mathbf{a}|\mathbf{s})) + \frac{\log(q_t(\mathbf{s}))}{\log(p_t(\mathbf{s}))} - 1 + \int_{\mathbf{s}'} V_{t+1}(\mathbf{s}') p(\mathbf{s}'|\mathbf{s},\mathbf{a}) d\mathbf{s}' \right) \right) & \text{, if } t < T \end{cases} \\
\frac{\partial \mathscr{G}(p_t(\mathbf{s}), V_t(\mathbf{s}))}{V_t(\mathbf{s})} &= \begin{cases} p_1(\mathbf{s}) - \mu_1(\mathbf{s}) & \text{, if } t = 1 \\ p_t(\mathbf{s}) - \int_{\mathbf{s},\mathbf{a}} \pi_{t-1}(\mathbf{a}|\mathbf{s}) p_{t-1}(\mathbf{s}) p(\mathbf{s}'|\mathbf{s},\mathbf{a}) & \text{, if } t > 1 \end{cases}
\end{aligned}$$

Let $q_{\mathrm{reg},t}(\mathbf{a}|\mathbf{s}) = \mathcal{N}\left(\mathbf{a}|\mu_{q_{\mathrm{reg},t}}(\mathbf{s}), \mathbf{\Sigma}_{q_{\mathrm{reg},t}}\right)$ and $\mu_{q_{\mathrm{reg},t}}(\mathbf{s}) = \mathbf{K}_{reg,t}\mathbf{s} + \mathbf{k}_{reg,t}$. The reward function can now be defined as

$$\begin{aligned}
r_t(\mathbf{s},\mathbf{a}) &= \alpha \log(q_{\mathrm{reg},t}(\mathbf{a}|\mathbf{s})) + \log(q_t(\mathbf{s})) - \log(p_t(\mathbf{s})) - 1 \\
&= -\frac{\alpha}{2} \left(\log |2\pi \mathbf{\Sigma}_{q_{\mathrm{reg},t}}| + (\mathbf{a} - \mu_{q_{\mathrm{reg},t}}(\mathbf{s}))^\top \mathbf{\Sigma}_{q_{\mathrm{reg},t}}^{-1} (\mathbf{a} - \mu_{q_{\mathrm{reg},t}}(\mathbf{s})) \right) \\
&\quad - \frac{1}{2} \left(\log |2\pi \mathbf{\Sigma}_{q,t}| + (\mathbf{s} - \mu_{q,t})^\top \mathbf{\Sigma}_{q,t}^{-1} (\mathbf{s} - \mu_{q,t}) \right) \\
&\quad + \frac{1}{2} \left(\log |2\pi \mathbf{\Sigma}_{s,t}| + (\mathbf{s} - \mu_{s,t})^\top \mathbf{\Sigma}_{s,t}^{-1} (\mathbf{s} - \mu_{s,t}) \right) - 1 \\
&= -\frac{\alpha}{2} \left(\log |2\pi \mathbf{\Sigma}_{q_{\mathrm{reg},t}}| + (\mathbf{a} - \mathbf{K}_{\mathrm{reg},t}\mathbf{s} - \mathbf{k}_{\mathrm{reg},t})^\top \mathbf{\Sigma}_{q_{\mathrm{reg},t}}^{-1} (\mathbf{a} - \mathbf{K}_{\mathrm{reg},t}\mathbf{s} - \mathbf{k}_{\mathrm{reg},t}) \right) \\
&\quad - \frac{1}{2} \left(\log |2\pi \mathbf{\Sigma}_{q,t}| + (\mathbf{s} - \mu_{q,t})^\top \mathbf{\Sigma}_{q,t}^{-1} (\mathbf{s} - \mu_{q,t}) \right) \\
&\quad + \frac{1}{2} \left(\log |2\pi \mathbf{\Sigma}_{s,t}| + (\mathbf{s} - \mu_{s,t})^\top \mathbf{\Sigma}_{s,t}^{-1} (\mathbf{s} - \mu_{s,t}) \right) - 1 \\
&= -\mathbf{s}^\top \mathbf{R}_t \mathbf{s} - \mathbf{s}^\top \mathbf{F_t} \mathbf{a} - \mathbf{a}^\top \mathbf{F_t}^\top \mathbf{s} - \mathbf{a}^\top \mathbf{H_t} \mathbf{a} + \mathbf{s}^\top \mathbf{r}_t + \mathbf{a}^\top \mathbf{h_t} + \text{ const},
\end{aligned}$$

with $\mathbf{R}_t = \frac{1}{2}\left(\mathbf{\Sigma}_{q,t}^{-1} + \alpha \mathbf{K}_{\mathrm{reg},\mathbf{t}}^\top \mathbf{\Sigma}_{q_{\mathrm{reg},t}}^{-1} \mathbf{K}_{\mathrm{reg},\mathbf{t}} - \mathbf{\Sigma}_{s,t}^{-1} \right)$, $\mathbf{r}_t = \mathbf{\Sigma}_{q,t}^{-1}\mu_{q,t} - \alpha \mathbf{K}_{\mathrm{reg}}^\top \mathbf{\Sigma}_{q_{\mathrm{reg},t}}^{-1} \mathbf{k}_{\mathrm{reg},\mathbf{t}} - \mathbf{\Sigma}_{s,t}^{-1}\mu_{s,t}$, $\mathbf{F}_t = -\frac{\alpha}{2}\mathbf{K}_{\mathrm{reg},\mathbf{t}}^\top \mathbf{\Sigma}_{q_{\mathrm{reg},t}}^{-1}$, $\mathbf{H}_t = -\frac{\alpha}{2}\mathbf{\Sigma}_{q_{\mathrm{reg},t}}^{-1}$ and $\mathbf{h}_t = \alpha \mathbf{\Sigma}_{q_{\mathrm{reg},t}}^{-1} \mathbf{k}_{\mathrm{reg},\mathbf{t}}$.

C The Difference between $\tilde{V}_t^\pi(\mathbf{s})$ and $V_t^\pi(\mathbf{s})$

Let $r_t(\mathbf{s},\mathbf{a}) = -\mathbf{s}^\top \mathbf{R}_t \mathbf{s} - \mathbf{s}^\top \mathbf{F}_t \mathbf{a} - \mathbf{a}^\top \mathbf{F}_t^\top \mathbf{s} - \mathbf{a}^\top \mathbf{H}_t \mathbf{a} + \mathbf{s}^\top \mathbf{r}_t + \mathbf{a}^\top \mathbf{h}_t$ denote the reward function for time step t. Further assume linear system dynamics $\mathbf{s}' \sim \mathcal{N}(\mathbf{s}'|\mathbf{A}_t\mathbf{s} + \mathbf{B}_t\mathbf{a} + \mathbf{b}_t, \Sigma_{\mathbf{s}',t})$. The state-action Value function can then be computed in closed form,

$$
\begin{aligned}
\mathbf{Q}_t(\mathbf{s},\mathbf{a}) &= r_t(\mathbf{s},\mathbf{a}) + \mathbb{E}_{\mathbf{s}'}\left[\mathbf{V}_{t+1}(\mathbf{s}')\right] \\
&= -\mathbf{s}^\top \mathbf{R}_t \mathbf{s} - \mathbf{s}^\top \mathbf{F}_t \mathbf{a} - \mathbf{a}^\top \mathbf{F}_t^\top \mathbf{s} - \mathbf{a}^\top \mathbf{H}_t \mathbf{a} + \mathbf{s}^\top \mathbf{r}_t + \mathbf{a}^\top \mathbf{h}_t + \mathbb{E}_{\mathbf{s}'}\left[\mathbf{V}_{t+1}(\mathbf{s})\right] \\
&= -\mathbf{s}^\top \mathbf{R}_t \mathbf{s} - \mathbf{s}^\top \mathbf{F}_t \mathbf{a} - \mathbf{a}^\top \mathbf{F}_t^\top \mathbf{s} - \mathbf{a}^\top \mathbf{H}_t \mathbf{a} + \mathbf{s}^\top \mathbf{r}_t + \mathbf{a}^\top \mathbf{h}_t \\
&\quad - (\mathbf{A}_t\mathbf{s} + \mathbf{B}_t\mathbf{a} + \mathbf{b}_t)^\top \mathbf{V}_{t+1} (\mathbf{A}_t\mathbf{s} + \mathbf{B}_t\mathbf{a} + \mathbf{b}_t) - \mathrm{Tr}(\mathbf{V}_{t+1}\Sigma_{s',t}) + \mathbf{v}_{t+1}^\top (\mathbf{A}_t\mathbf{s} + \mathbf{B}_t\mathbf{a} + \mathbf{b}_t) - v_{t+1} \\
&= -\mathbf{s}^\top \mathbf{R}_t \mathbf{s} - \mathbf{s}^\top \mathbf{F}_t \mathbf{a} - \mathbf{a}^\top \mathbf{F}_t^\top \mathbf{s} - \mathbf{a}^\top \mathbf{H}_t \mathbf{a} + \mathbf{s}^\top \mathbf{r}_t + \mathbf{a}^\top \mathbf{h}_t \\
&\quad - \mathbf{s}^\top \mathbf{A}_t^\top \mathbf{V}_{t+1} \mathbf{A}_t \mathbf{s} - \mathbf{s}^\top \mathbf{A}_t^\top \mathbf{V}_{t+1} \mathbf{B}_t \mathbf{a} - \mathbf{s}^\top \mathbf{A}_t^\top \mathbf{V}_{t+1} \mathbf{b}_t - \mathbf{a}^\top \mathbf{B}_t^\top \mathbf{V}_{t+1} \mathbf{A}_t \mathbf{s} \\
&\quad - \mathbf{a}^\top \mathbf{B}_t^\top \mathbf{V}_{t+1} \mathbf{B}_t \mathbf{a} - \mathbf{a}^\top \mathbf{B}_t^\top \mathbf{V}_{t+1} \mathbf{b}_t - \mathbf{b}_t^\top \mathbf{V}_{t+1} \mathbf{A}_t \mathbf{s} - \mathbf{b}_t^\top \mathbf{V}_{t+1} \mathbf{B}_t \mathbf{a} - \mathbf{b}_t^\top \mathbf{V}_{t+1} \mathbf{b}_t - \mathrm{Tr}(\mathbf{V}_{t+1}\Sigma_{s',t}) \\
&\quad + \mathbf{v}_{t+1}^\top \mathbf{A}_t \mathbf{s} + \mathbf{v}_{t+1}^\top \mathbf{B}_t \mathbf{a} + \mathbf{v}_{t+1}^\top \mathbf{b}_t - v_{t+1} \\
&= -\frac{1}{2}\mathbf{s}^\top \mathbf{I}_t \mathbf{s} - \frac{1}{2}\mathbf{s}^\top \mathbf{J}_t \mathbf{a} - \frac{1}{2}\mathbf{a}^\top \mathbf{J}_t^\top \mathbf{s} - \frac{1}{2}\mathbf{a}^\top \mathbf{K}_t \mathbf{a} + \mathbf{s}^\top \mathbf{l}_t + \mathbf{a}^\top \mathbf{m}_t + c_t,
\end{aligned}
$$

where $\mathbf{I}_t = 2\left(\mathbf{R}_t + \mathbf{A}_t^\top \mathbf{V}_{t+1} \mathbf{A}_t\right)$, $\mathbf{J}_t = 2\left(\mathbf{A}_t^\top \mathbf{V}_{t+1} \mathbf{B}_t + \mathbf{F}_t\right)$, $\mathbf{K}_t = 2\left(\mathbf{H}_t + \mathbf{B}_t^\top \mathbf{V}_{t+1} \mathbf{B}_t\right)$, $\mathbf{l}_t = \left(\mathbf{r}_t - 2\mathbf{A}_t^\top \mathbf{V}_{t+1} \mathbf{b}_t + \mathbf{A}_t^\top \mathbf{v}_{t+1}\right)$, $\mathbf{m}_t = \left(\mathbf{h}_t - 2\mathbf{B}_t^\top \mathbf{V}_{t+1} \mathbf{b}_t + \mathbf{B}_t^\top \mathbf{v}_{t+1}\right)$ and $c_t = -\mathbf{b}_t^\top \mathbf{V}_{t+1} \mathbf{b}_t - \mathrm{Tr}(\mathbf{V}_{t+1}\Sigma_{s',t}) + \mathbf{v}_{t+1}^\top \mathbf{b}_t - v_{t+1}$.

Using $\mathbf{z} = \begin{pmatrix} \mathbf{s} \\ \mathbf{a} \end{pmatrix}$, $\mathbf{C}_t = \begin{pmatrix} \mathbf{I}_t & \mathbf{J}_t \\ \mathbf{J}_t^\top & \mathbf{K}_t \end{pmatrix}$ and $\mathbf{c}_t = \begin{pmatrix} \mathbf{l}_t \\ \mathbf{m}_t \end{pmatrix}$ the state-action value function can be written

$$
Q_t(\mathbf{s},\mathbf{a}) = -\frac{1}{2}\mathbf{z}^\top \mathbf{C}_t \mathbf{z} + \mathbf{z}^\top \mathbf{c}_t + c_t.
$$

This induction is used for computing the Value function of the maximum entropy policy $V^\pi(\mathbf{s},\mathbf{a}) = \mathbb{E}_\pi\left[\mathbf{Q}^\pi{}_{t+1}(\mathbf{s},\mathbf{a})\right]$ as well as for computing the Lagrangian multiplier $\tilde{V}_t^\pi(\mathbf{s},\mathbf{a}) = \operatorname{softmax}_{\mathbf{a}} \tilde{Q}_t^\pi(\mathbf{s},\mathbf{a})$ for MaxCausalEnt-IRL or IRL-MSD.

C.1 Derivation of $V_t^\pi(\mathbf{s})$

The softmax-policy $\pi_t(\mathbf{a}|\mathbf{s}) \propto Q_t(\mathbf{s},\mathbf{a})$ can be computed as conditional of the joint distribution $\mathcal{N}[\mathbf{z}|\mathbf{c},\mathbf{C}] \propto Q_t(\mathbf{s},\mathbf{a})$,

$$
\pi_t(\mathbf{a}|\mathbf{s}) = \mathcal{N}[\mathbf{a}|\mathbf{m}_t - \mathbf{J}_t^\top \mathbf{s}, \mathbf{K}_t] = \mathcal{N}(\mathbf{a}|\mathbf{K}_t^{-1}\mathbf{m}_t - \mathbf{K}_t^{-1}\mathbf{J}_t^\top \mathbf{s}, \mathbf{K}_t^{-1}) = \mathcal{N}(\mathbf{a}|\mu_{a,t}, \Sigma_{a,t}).
$$

The policy can then be employed to compute the corresponding Value function,

$$
\begin{aligned}
V_t^{\pi}(\mathbf{s}) &= \mathbb{E}_{\pi(\mathbf{a}|\mathbf{s})}[Q_t(\mathbf{s},\mathbf{a})] \\
&= -\mathbf{s}^\top \frac{1}{2}\mathbf{I}_t\mathbf{s} - \mathbf{s}^\top \frac{1}{2}\mathbf{J}_t\mu_{a,t} - \mu_{a,t}^\top \frac{1}{2}\mathbf{J}_t^\top \mathbf{s} - \mu_{a,t}^\top \frac{1}{2}\mathbf{K}_t\mu_{a,t} + \mathbf{s}^\top \mathbf{l}_t + \mu_{a,t}^\top \mathbf{m}_t + c - \mathrm{Tr}\,\frac{1}{2}(\mathbf{K}_t\mathbf{K}_t^{-1}) \\
&= -\mathbf{s}^\top \frac{1}{2}\mathbf{I}_t\mathbf{s} - \mathbf{s}^\top \frac{1}{2}\mathbf{J}_t\mathbf{K}^{-1}\mathbf{m}_t + \mathbf{s}^\top \frac{1}{2}\mathbf{J}_t\mathbf{K}_t^{-1}\mathbf{J}_t^\top \mathbf{s} - \mathbf{m}_t^\top \mathbf{K}_t^{-1\top} \frac{1}{2}\mathbf{J}_t^\top \mathbf{s} + \mathbf{s}^\top \mathbf{J}_t\mathbf{K}_t^{-1\top} \frac{1}{2}\mathbf{J}_t^\top \mathbf{s} \\
&\quad - \mathbf{m}_t^\top \mathbf{K}_t^{-1\top} \frac{1}{2}\mathbf{K}_t\mathbf{K}_t^{-1}\mathbf{m}_t + \mathbf{m}_t^\top \mathbf{K}_t^{-1\top} \frac{1}{2}\mathbf{K}_t\mathbf{K}_t^{-1}\mathbf{J}_t^\top \mathbf{s} \\
&\quad + \mathbf{s}^\top \mathbf{J}_t\mathbf{K}_t^{-1\top} \frac{1}{2}\mathbf{K}_t\mathbf{K}_t^{-1}\mathbf{m}_t - \mathbf{s}^\top \mathbf{J}_t\mathbf{K}_t^{-1\top} \frac{1}{2}\mathbf{K}_t\mathbf{K}_t^{-1}\mathbf{J}_t^\top \mathbf{s} \\
&\quad + \mathbf{s}^\top \mathbf{l}_t + \mathbf{m}_t^\top \mathbf{K}_t^{-1\top} \mathbf{m}_t - \mathbf{s}^\top \mathbf{J}_t\mathbf{K}_t^{-1\top} \mathbf{m}_t + c_t - \frac{1}{2}N_a \\
&= -\mathbf{s}^\top \mathbf{V}_t\mathbf{s} + \mathbf{v}_t^\top \mathbf{s} - \upsilon_t,
\end{aligned}
$$

with $\mathbf{V}_t = \frac{1}{2}(\mathbf{I}_t - \mathbf{J}_t\mathbf{K}_t^{-1}\mathbf{J}_t^\top)$, $\mathbf{v}_t = \mathbf{l}_t - \mathbf{J_t}\mathbf{K_t}^{-1}\mathbf{m}_t$ and $\upsilon_t = -\frac{1}{2}\mathbf{m}_t^\top \mathbf{K}_t^{-1}\mathbf{m}_t - c_t + \frac{1}{2}N_a$.

C.2 Derivation of $\tilde{V}_t^{\pi}(\mathbf{s})$

The exponential of the state-action Value function can be regarded as unnormalized Gaussian joint distribution

$$
\begin{aligned}
\exp\left(\tilde{Q}_t^{\pi}(\mathbf{s},\mathbf{a})\right) &= \exp\left(-\frac{1}{2}\mathbf{z}^\top \mathbf{C}_t\mathbf{z} + \mathbf{z}^\top \mathbf{c}_t + c_t\right) \\
&= \frac{|2\pi\mathbf{C}_t^{-1}|^{\frac{1}{2}}}{\exp\left(-c_t - \frac{1}{2}\mathbf{c_t}^\top \mathbf{C_t}^{-1}\mathbf{c_t}\right)}\mathcal{N}[\mathbf{z}|\mathbf{c_t},\mathbf{C_t}].
\end{aligned}
$$

The Lagrangian multiplier $\tilde{V}_t(\mathbf{s})$ can now be computed,

$$
\begin{aligned}
\tilde{V}_t^{\pi}(\mathbf{s}) &= \log\int_{\mathbf{a}} \exp\left(\tilde{Q}_t^{\pi}(\mathbf{s},\mathbf{a})\right) \\
&= \log\int_{\mathbf{a}} \frac{|2\pi\mathbf{C}_t^{-1}|^{\frac{1}{2}}}{\exp\left(-c_t - \frac{1}{2}\mathbf{c_t}^\top \mathbf{C_t}^{-1}\mathbf{c}\right)}\mathcal{N}[\mathbf{z}|\mathbf{c},\mathbf{C}] \\
&= \log\frac{|2\pi\mathbf{C}^{-1}|^{\frac{1}{2}}}{\exp\left(-c_t - \frac{1}{2}\mathbf{c_t}^\top \mathbf{C_t}^{-1}\mathbf{c_t}\right)} + \log\mathcal{N}[\mathbf{s}|\mathbf{l_t} - \mathbf{J_t}\mathbf{K_t}^{-1}\mathbf{m_t}, \mathbf{I_t} - \mathbf{J_t}\mathbf{K_t}^{-1}\mathbf{J_t}^\top] \\
&= \log\frac{|2\pi\mathbf{C}_t^{-1}|^{\frac{1}{2}}}{\exp\left(-c_t - \frac{1}{2}\mathbf{c_t}^\top \mathbf{C_t}^{-1}\mathbf{c_t}\right)} \\
&\quad -\frac{1}{2}\Big[\log|2\pi(\mathbf{I_t} - \mathbf{J_t}\mathbf{K_t}^{-1}\mathbf{J_t}^\top)^{-1}| + (\mathbf{l_t} - \mathbf{J_t}\mathbf{K_t}^{-1}\mathbf{m_t})^\top(\mathbf{I_t} - \mathbf{J_t}\mathbf{K_t}^{-1}\mathbf{J_t}^\top)^{-1}(\mathbf{l_t} - \mathbf{J_t}\mathbf{K_t}^{-1}\mathbf{m_t}) \\
&\quad +\mathbf{s}^\top(\mathbf{I_t} - \mathbf{J_t}\mathbf{K_t}^{-1}\mathbf{J_t}^\top)\mathbf{s} - 2\mathbf{s}^\top(\mathbf{l_t} - \mathbf{J_t}\mathbf{K_t}^{-1}\mathbf{m_t})\Big] \\
&= -\mathbf{s}^\top \mathbf{V_t}\mathbf{s} + \mathbf{v_t}^\top \mathbf{s} - \upsilon_t,
\end{aligned}
$$

with $V_t(s) = \frac{1}{2}(\mathbf{I_t} - \mathbf{J_t}\mathbf{K_t^{-1}}\mathbf{J_t^\top})$, $\mathbf{v}_t(\mathbf{s}) = (\mathbf{l_t} - \mathbf{J_t}\mathbf{K_t^{-1}}\mathbf{m_t})$ and $\upsilon_t = -c_t - \frac{1}{2}\mathbf{c}_t^\top \mathbf{C}_t^{-1}\mathbf{c}_t - \frac{1}{2}N_a \log(2\pi) - \frac{1}{2}\log\frac{|V_t^{-1}|}{|C_t^{-1}|} + \frac{1}{4}\mathbf{v}_t^\top \mathbf{V}_t^{-1}\mathbf{v}_t$. The state-independent term can be further simplified,

$$
\begin{aligned}
\upsilon_t &= -c_t - \frac{1}{2}\mathbf{c}_t^\top \mathbf{C}_t^{-1}\mathbf{c}_t - \frac{1}{2}N_a \log(2\pi) - \frac{1}{2}\log\frac{|V_t^{-1}|}{|C_t^{-1}|} + \frac{1}{4}\mathbf{v}_t^\top \mathbf{V}_t^{-1}\mathbf{v}_t \\
&= -c_t - \frac{1}{2}\begin{pmatrix}\mathbf{l}_t \\ \mathbf{m}_t\end{pmatrix}^\top \begin{pmatrix} (\mathbf{I}_t - \mathbf{J}_t\mathbf{K}_t^{-1}\mathbf{J}_t^\top)^{-1} & -(\mathbf{I}_t - \mathbf{J}_t\mathbf{K}_t^{-1}\mathbf{J}^\top)^{-1}\mathbf{J}_t\mathbf{K}_t^{-1} \\ -\mathbf{K}_t^{-1}\mathbf{J}_t^\top(\mathbf{I}_t - \mathbf{J}_t\mathbf{K}_t^{-1}\mathbf{J}_t)^{-1} & (\mathbf{K}_t - \mathbf{J}_t^\top\mathbf{I}_t^{-1}\mathbf{J}_t)^{-1}\end{pmatrix}\begin{pmatrix}\mathbf{l}_t \\ \mathbf{m}_t\end{pmatrix} - \frac{1}{2}N_a\log(2\pi) - \frac{1}{2}\log|\mathbf{K}_t^{-1}| + \frac{1}{4}\mathbf{v}_t^\top\mathbf{V}_t^{-1}\mathbf{v}_t \\
&= -c_t - \frac{1}{2}\begin{pmatrix}\mathbf{l}_t \\ \mathbf{m}_t\end{pmatrix}^\top \begin{pmatrix} \frac{1}{2}\mathbf{V}_t^{-1} & -\frac{1}{2}\mathbf{V}_t^{-1}\mathbf{J}_t\mathbf{K}_t^{-1} \\ \frac{1}{2}\mathbf{K}_t^{-1}\mathbf{J}_t\mathbf{V}_t^{-1} & \mathbf{K}_t^{-1} - \mathbf{K}_t^{-1}\mathbf{J}_t^\top(-\mathbf{I}_t + \mathbf{J}_t\mathbf{K}_t^{-1}\mathbf{J}_t^\top)\mathbf{J}_t\mathbf{K}_t^{-1}\end{pmatrix}\begin{pmatrix}\mathbf{l}_t \\ \mathbf{m}_t\end{pmatrix} - \frac{1}{2}\log|2\pi\mathbf{K}_t^{-1}| + \frac{1}{4}\mathbf{v}_t^\top\mathbf{V}_t^{-1}\mathbf{v}_t \qquad \text{(C.1a)} \\
&= -c_t - \frac{1}{2}\begin{pmatrix}\mathbf{l}_t \\ \mathbf{m}_t\end{pmatrix}^\top \begin{pmatrix} \frac{1}{2}\mathbf{V}_t^{-1} & -\frac{1}{2}\mathbf{V}_t^{-1}\mathbf{J}_t\mathbf{K}_t^{-1} \\ \frac{1}{2}\mathbf{K}_t^{-1}\mathbf{J}_t\mathbf{V}_t^{-1} & \mathbf{K}_t^{-1} + (\mathbf{J}_t\mathbf{K}_t^{-1})^\top\mathbf{V}_t^{-1}(\mathbf{J}_t\mathbf{K}_t^{-1})\end{pmatrix}\begin{pmatrix}\mathbf{l}_t \\ \mathbf{m}_t\end{pmatrix} - \frac{1}{2}\log|2\pi\mathbf{K}_t^{-1}| + \frac{1}{4}\mathbf{v}_t^\top\mathbf{V}_t^{-1}\mathbf{v}_t \\
&= -c_t - \frac{1}{2}\mathbf{m}_t^\top\mathbf{K}_t^{-1}\mathbf{m}_t - \frac{1}{4}\mathbf{v}_t^\top\mathbf{V}_t^{-1}\mathbf{v}_t - \frac{1}{2}\log|2\pi\mathbf{K}_t^{-1}| + \frac{1}{4}\mathbf{v}_t^\top\mathbf{V}_t^{-1}\mathbf{v}_t \\
&= -c_t - \frac{1}{2}\mathbf{m}_t^\top\mathbf{K}_t^{-1}\mathbf{m}_t - \frac{1}{2}\log|2\pi\mathbf{K}_t^{-1}|.
\end{aligned}
$$

Equation C.1a can be derived by using the Woodbury matrix identity (Woodbury, 1950).

C.3 Comparison of $V_t^\pi(\mathbf{s})$ and $\tilde{V}_t^\pi(\mathbf{s})$

Comparing $V_t^\pi(\mathbf{s})$ and $\tilde{V}_t^\pi(\mathbf{s})$ reveals, that both functions have the same state dependent parts, whereas the computed state-independent parts differ at each time step by $\frac{1}{2}(N_a + \log|2\pi\mathbf{K}_t^{-1}|)$. This difference is inherited by the state-action Value function and thereby passed to $V_{t-1}^\pi(\mathbf{s})$. Hence the difference at time step t between the Value function of the maximum entropy policy, $V_t^\pi(\mathbf{s})$, and the Lagrangian multiplier $\tilde{V}_t^\pi(\mathbf{s})$ is given by

$$V_t^\pi(\mathbf{s}) - \tilde{V}_t^\pi(\mathbf{s}) = \sum_{i=t}^{T-1}\frac{1}{2}(N_a + \log|2\pi\mathbf{K}_i^{-1}|),$$

which corresponds to the sum of the log partition functions of the current policy and all following ones plus a constant offset.