Sample-Efficient I-Projections for Robot Learning

Arenz, Julian Oleg

(2021)

Sample-Efficient I-Projections for Robot Learning

Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)
Genehmigte Dissertation von Oleg Arenz aus Wiesbaden
Tag der Einreichung: 12.08.2020, Tag der Prüfung: 23.09.2020

1. Gutachten: Prof. Jan Peters, Ph.D.
2. Gutachten: Prof. Dr. techn. Gerhard Neumann
Darmstadt – D 17

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Computer Science
Department
Intelligente Autonome
Systeme

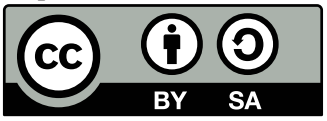
Sample-Efficient I-Projections for Robot Learning

Accepted doctoral thesis by Oleg Arenz

1. Review: Prof. Jan Peters, Ph.D.
2. Review: Prof. Dr. techn. Gerhard Neumann

Date of submission: 12.08.2020
Date of thesis defense: 23.09.2020

Darmstadt, Technische Universität Darmstadt



Für Lea & Luise

Erklärungen laut Promotionsordnung

§8 Abs. 1 lit. c PromO

Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

§8 Abs. 1 lit. d PromO

Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. In diesem Fall sind nähere Angaben über Zeitpunkt, Hochschule, Dissertationsthema und Ergebnis dieses Versuchs mitzuteilen.

§9 Abs. 1 PromO

Ich versichere hiermit, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde.

§9 Abs. 2 PromO

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 12.08.2020

O. Arenz

Abstract

Robots had a great impact on the manufacturing industry ever since the early seventies when companies such as KUKA and ABB started deploying their first industrial robots. These robots merely performed very specific tasks in specific ways within well-defined environments. Still, they proved to be very useful as they could exceed human performance at these tasks.[1] However, in order to enable robots to enter our daily life, they need to become more versatile and need to operate in much less structured environments. This thesis is partly devoted to stretching these limitations by means of learning, namely imitation learning (IL) and inverse reinforcement learning (IRL).

Reinforcement learning (RL) is a powerful approach to enable robots to solve a task in an unknown environment. The practitioner describes a desired behavior by specifying a reward function and the robot autonomously interacts with the environment in order to find a control policy that generates high accumulated reward. However, RL is not suitable for teaching new tasks by non-experts because specifying appropriate reward functions can be difficult. Demonstrating the desired behavior is often easier for non-experts. Imitation learning can be used in order to enable the robot to reproduce the demonstrations. However, without explicitly inferring and modeling the intentions of the demonstrations, it can become difficult to solve the task for unseen situations. Inverse reinforcement learning (IRL) therefore aims to infer a reward function from the demonstrations, such that optimizing this reward function yields the desired behavior even for different situations.

This thesis introduces a unifying approach to solve the inverse reinforcement learning problem in the same way as the reinforcement learning problem. This is achieved by framing both problems as information projection problems, i.e., we strive to minimize the relative entropy between a probabilistic model of the robot behavior and a given desired distribution. Furthermore, a trust region on the robot behavior is used to stabilize the optimization. For inverse reinforcement learning, the desired distribution is implicitly given by the expert demonstrations. The resulting optimization can be efficiently solved using state-of-the-art reinforcement learning methods. For reinforcement learning, the log-likelihood of the desired distribution is given by the reward function. The resulting

[1] Interestingly, the same could be said about current machine learning methods.

optimization problem corresponds to a standard reinforcement learning formulation, except for an additional objective of maximizing the entropy of the robot behavior. This entropy objective adds little overhead to the optimization, but can lead to better exploration and more diversified policies.

Trust-region I-projections are not only useful for training robots, but can also be applied to other machine learning problems. I-projections are typically used for variational inference, in order to approximate an intractable distribution by a simpler model. However, the resulting optimization problems are usually optimized based on stochastic gradient descent which often suffers from high variance in the gradient estimates. As trust-region I-projections where shown to be effective for reinforcement learning and inverse reinforcement learning, this thesis also explores their use for variational inference. More specifically, trust-region I-projections are investigated for the problem of approximating an intractable distribution by a Gaussian mixture model (GMM) with an adaptive number of components. GMMs are highly desirable for variational inference because they can yield arbitrary accurate approximations while inference from GMMs is still relatively cheap. In order to make learning the GMM feasible, we derive a lower bound that enables us to decompose the objective function. The optimization can then be performed by iteratively updating individual components using a technique from reinforcement learning. The resulting method is capable of learning approximations of significantly higher quality than existing variational inference methods.

Due to the similarity of the underlying optimization problems, the insights gained from our variational inference method are also useful for IL and IRL. Namely, a similar lower bound can be applied also for the I-projection formulation of imitation learning. However, whereas for variational inference the lower bound serves to decompose the objective function, for imitation learning it allows us to provide a reward signal to the robot that does not depend on its behavior. Compared to reward functions that are relative to the current behavior of the robot—which are typical for popular adversarial methods—behavior-independent reward functions have the advantages that we can show convergence even for greedy optimization. Furthermore, behavior-independent reward functions solve the inverse reinforcement learning problem, thereby closing the gap between imitation learning and IRL. However, algorithms derived from our non-adversarial formulation are actually very similar to existing AIL methods, and we can even show that adversarial inverse reinforcement learning (AIRL) is indeed an instance of our formulation. AIRL was derived from an adversarial formulation, and we point out several problems of that derivation. In contrast, we show that AIRL can be straightforwardly derived from out non-adversarial formulation. Furthermore, we demonstrate that the non-adversarial formulation can be also used to derive novel algorithms by presenting a non-adversarial method for offline imitation learning.

Zusammenfassung

Roboter haben schon seit den frühen siebziger Jahren einen großen Einfluss auf die Fertigungsindustrie, als Unternehmen wie KUKA und ABB ihre ersten Industrieroboter auslieferten. Diese Roboter führten zwar in der Regel nur eng definierte Aufgaben auf ganz bestimmte Weise und in genau definierten Umgebungen aus, doch erwiesen sie sich schon damals als sehr nützlich, da sie bei diesen Aufgaben den Menschen überlegen waren.[1] Um es Robotern jedoch zu ermöglichen, auch in unserem täglichen Leben von Nutzen zu sein, müssen sie vielseitiger werden und in viel weniger strukturierten Umgebungen arbeiten können.

Verstärkendes Lernen ist ein vielversprechender Ansatz, um es Robotern zu ermöglichen, eine Aufgabe in einer unbekannten Umgebung zu lösen. Der Robotik-Experte beschreibt ein gewünschtes Verhalten, indem er eine Belohnungsfunktion angibt, die das Verhalten des Roboters kontinuierlich bewertet und diese skalaren Bewertungen an den Roboter weitergibt. Der Roboter interagiert autonom mit der Umgebung, und verändert sein Verhalten mit dem Ziel, auf lange Sicht eine hohe Belohnung zu erhalten. Verstärkendes Lernen eignet sich jedoch nicht für das Lehren neuer Aufgaben durch Nicht-Experten, da es sehr schwierig ist solche mathematischen Belohnungsfunktionen so zu definieren, dass sie zum gewünschten Verhalten führen. Für Nicht-Experten ist es häufig einfacher das gewünschte Verhalten vorzumachen. Lernen durch Imitation kann verwendet werden, um den Roboter in die Lage zu versetzen, solche Demonstrationen zu reproduzieren. Ohne die Absichten der Demonstrationen explizit abzuleiten und zu modellieren, kann es jedoch schwierig werden, die Aufgabe unter veränderten Bedingungen zu lösen. Inverses Verstärkendes Lernen zielt daher darauf ab, aus den Demonstrationen eine Belohnungsfunktion abzuleiten, sodass die Optimierung dieser Belohnungsfunktion auch für neue Situationen zum gewünschten Verhalten führt.

Diese Arbeit stellt einen vereinheitlichenden Ansatz vor, um Inverses Verstärkendes Lernen auf die gleiche Weise zu lösen wie Verstärkendes Lernen. Dies wird erreicht, indem beide Probleme als Informations-Projektion (I-Projection) formuliert werden, das heißt es wird versucht, die relative Entropie zwischen einem probabilistischen Modell des

[1] Interessanterweise könnte man dasselbe über die aktuellen Methoden des maschinellen Lernens sagen.

Roboterverhaltens und einer gegebenen gewünschten Wahrscheinlichkeitsverteilung zu minimieren. Um die Stabilität beim Lösen dieses Optimierungsproblems zu erhöhen wird ein sogenanntes Trust-Region-Verfahren angewendet. Das Trust-Region-Verfahren ist ein iteratives Verfahren, bei dem sich das Verhalten des Roboters zwischen jeder Iteration nur leicht verändern darf. Die resultierende Trust-Region I-Projection kann sowohl für Verstärkendes Lernen als auch für Inverses Verstärkendes Lernen angewendet werden. Beim inversen Verstärkenden Lernen ist die gewünschte Verteilung durch die Demonstrationen implizit gegeben. Die daraus resultierende Optimierung kann mit modernsten Methoden des Verstärkenden Lernens effizient gelöst werden. Beim Verstärkenden Lernen ist die (unnormalisierte) Wahrscheinlichkeitverteilung durch die exponentierte Belohnungsfunktion gegeben. Das resultierende Optimierungsproblem entspricht einer Standardformulierung des Verstärkenden Lernens, doch wird zusätzlich noch versucht die Entropie des Roboterverhaltens zu maximieren. Dieses Entropie-Kriterium verleitet den Roboter dazu, die Auswirkungen seines Verhaltens besser zu erkunden und führt zudem zu vielseitigem Verhalten.

Trust-Region I-Projections sind nicht nur für das Trainieren von Robotern nützlich, sondern können auch auf andere Problemstellungen des Maschinellen Lernens angewandt werden. So werden I-Projections häufig dazu verwendet komplexe Wahrscheinlichkeitsverteilungen durch ein einfacheres Modell zu approximieren. Das entsprechende Optimierungsproblem nennt sich Variational Inference und wird häufig mit einem Verfahren namens Stochastic Gradient Descent gelöst. Da sich die Trust-Region I-Projection allerdings sowohl für Verstärkendes Lernen als auch für Inverses Verstärkendes Lernen als effektiv erwiesen hat, untersuchen wir diesen Ansatz auch für Variational Inference. Genauer gesagt wird die Trust-Region I-Projection auf das Problem der Approximation einer komplexen Verteilung durch ein Gaußsches Mischmodell (GMM) untersucht. GMMs eignen sich zur Variational Inference, da sie jede Verteilung beliebig genau approximieren können, dabei allerdings relativ einfach zu handhaben sind. Um das Lernen des GMMs zu vereinfachen, leiten wir eine untere Schranke ab, die es uns ermöglicht, das Optimierungskriterium zu zerlegen. Die Optimierung kann dann durch iteratives Lernen einzelner Komponenten mithilfe einer Technik aus dem Verstärkendem Lernen durchgeführt werden. Die resultierende Methode ist in der Lage, Approximationen von deutlich höherer Qualität zu lernen als bestehenden Methoden der Variational Inference.

Aufgrund der Ähnlichkeit der zugrundeliegenden Optimierungsprobleme sind die mit unserer Variational-Inference-Methode gewonnenen Erkenntnisse auch für das Lernen durch Imitation und für Inverses Verstärkende Lernen relevant. Eine ähnliche untere Schranke kann nämlich auch für die I-Projection-Formulierung des Lernens durch Imitation angewandt werden. Während jedoch bei der Variational Inference die untere Schranke dazu dient, die Zielfunktion zu zerlegen, erlaubt sie uns beim Lernen durch

Imitation, eine Belohnungsfunktion zu lernen, die nicht vom momentanen Verhalten abhängt. Verglichen mit Belohnungsfunktionen, die relativ zum aktuellen Verhalten des Roboters sind – was typisch für die momentan beliebten adversarialen Methoden ist –, haben verhaltensunabhängige Belohnungsfunktionen den Vorteil, dass wir selbst bei vollständiger Optimierung in jeder Iteration, Konvergenz zeigen können. Darüber hinaus lösen verhaltensunabhängige Belohnungsfunktionen das Problem des Inversen Verstärkenden Lernens und schließen damit die Lücke zwischen dem Lernen durch Imitation und Inversem Verstärkenden Lernen. Allerdings sind die Algorithmen, die aus unserer nicht-adversarialen Formulierung abgeleitet wurden, den bestehenden adversarialen Methoden sehr ähnlich, und wir können sogar zeigen, dass die Methode namens Adversarial Inverse Reinforcement Learning (AIRL) tatsächlich ein Beispiel für unsere Formulierung ist. AIRL wurde allerdings aus einer adversarialen Formulierung abgeleitet, was zu mehreren Problemen führte, die wir in dieser Arbeit aufzeigen. Im Gegensatz dazu zeigen wir, dass AIRL direkt aus unserer nicht-adversarialen Formulierung abgeleitet werden kann. Darüber hinaus zeigen wir, dass die nicht-adversariale Formulierung auch zur Ableitung neuer Algorithmen verwendet werden kann, indem wir eine nicht-adversariale Methode für das interaktionslose Lernen durch Imitation vorstellen.

Acknowledgments

First of all, I would like to express my deepest gratitude to my adviser Gerhard Neumann, who—apart from me—had the most direct impact on this thesis. Already during my master's studies, you invested countless hours teaching me how to derive, debug and improve an algorithm, and during my PhD, you were always up for a scientific discussion and very helpful whenever I had questions. I am well-aware of the privilege I had by getting mentored by a great researcher who was still at the beginning of his career.

I was also privileged for being part of a group full of smart people and well-maintained robots and, thus, would like to extend my deepest gratitude also to Jan Peters who gave me a home at the intelligent autonomous systems group. You were very understanding and provided me with the time and freedom (and also the money!) that allowed me to become father of two children during my PhD and still finish this thesis.

No matter how much you love what you are doing, you can not be happy at work without colleagues you enjoy working with. Some of you left already years ago, some of you where just here for a visit, some of you just joined the group and some of you are from other labs. I would like to express my appreciation to Joni, Rudi, Diego, Marco, Doro, Takayuki, Riad, Boris, Hany, Samuele, Julen, Firas, Philipp, Max and Onur for a variety of different reasons. Whether we had nice collaborations, whether I could always ask you some questions, whether you are constantly challenging my views or whether I just enjoyed hanging around with you—thank you all for a great time. Special thanks to Veronika, Marion, Nanette and Sabine; not only for taking care of any administrative hurdles, but also for your kindness and warmth. You are invaluable.

I gratefully acknowledge the effort of Prof. Kersting and Prof. Rothkopf by participating in the defense.

I would also like to thank my family—my parents for their unconditional love and support, and my wife for providing me with the strength for finishing this thesis. I dedicate this work to my daughters, Lea and Luise, because they somehow manage to give back even more energy than they take. Despite your young age, you are a massive tower of strength.

Contents

1. Introduction

Optimization problems are mathematical formulations that are fundamental for robot learning and machine learning in general. Optimization problems are typically written in the form [Boyd and Vandenberghe, 2004]

$$\begin{aligned} \underset{\boldsymbol{\theta}}{\text{maximize}} \quad & J(\boldsymbol{\theta}) \\ \text{subject to} \quad & f_i(\boldsymbol{\theta}) \leq b_i, \quad i = 1, \ldots, N_i \\ & g_j(\boldsymbol{\theta}) = c_j, \quad j = 1, \ldots, N_j, \end{aligned}$$

where $\boldsymbol{\theta}$ is a vector of parameters, J is the objective function, f_i and g_j are inequality and equality constraint functions and b_i and c_j are the corresponding bounds. Formulating an appropriate optimization problem and deriving an approximate solution is often the main step when developing new robot learning methods. Accordingly, when we say that a robot is learning, it is often just solving such optimization problems.

Some optimization problems can be solved analytically, e.g., ordinary least-squares OLS [Russell and Norvig, 2016]. However, such closed-form solutions rarely exist– especially for robot learning applications, where the objective function often depends on interactions with the real world. For example, let $\boldsymbol{\theta}$ be the parameters of the robot controller that chooses actions depending on perceptions of the environment and let $J(\boldsymbol{\theta})$ be a performance function rating the resulting behavior of the robot. As the interactions between the robot and its environment can usually not be modeled sufficiently accurate, it is often not even possible to evaluate the objective function. Instead, the performance of the robot for certain parameters has to be approximated by applying them and observing the resulting behavior. Such *samples* can be used to either implicitly or explicitly model the relation between parameters and the corresponding performance. These models, however, are typically only valid in the vicinity of the samples they are based on.

Trust region methods [Yuan, 2015] employ an iterative procedure to converge to locally optimal solutions for such intractable objective functions. At each iteration i, trust region methods build a model to approximate the objective function around the current iterate $\boldsymbol{\theta}^{(i)}$. Subsequently, a new subproblem is formed by using the model as surrogate for the intractable objective function and by adding a constraint that ensures that the

parameters remain in a region for which the model is sufficiently accurate. The solution of this subproblem defines the next iterate. Traditionally, the trust region constraints were defined by upper-bounding the euclidean distance to the current iterate [Sorensen, 1982]. However, in the field of robot learning, information-geometric trust regions are becoming increasingly popular [Abdolmaleki et al., 2015; Levine and Abbeel, 2014; Peters et al., 2010; Schulman et al., 2015]. For this subclass of trust region methods, the parameters $\boldsymbol{\theta}$ define a probability distribution and the trust regions are bounded based on information-geometric divergences between the respective distributions.

In this thesis we will explore the use of information-geometric trust regions to optimize a specific type of information-geometric objective functions. The resulting high-level optimization procedure will be denoted as trust region I-projections. We will show that trust region I-projections can be adopted to derive novel state-of-the-art algorithms for various different applications. Namely, we show that the problems of variational inference (VI), reinforcement learning (RL) and inverse reinforcement learning (IRL) can be framed as trust-region I-projections and that the resulting optimization problems can be efficiently solved. Before introducing these fields and motivating the use of trust region I-projections, relevant concepts from information theory need to be discussed.

1.1. Background

The methods discussed in this thesis heavily build on information theoretic quantities like entropies and Kullback-Leibler (KL) divergences. In this section we will introduce these quantities and explore a specific type of optimization problems called I-projections. Based on these preliminaries, the concept of trust region I-projections is defined which will underlie the remainder of this thesis.

1.1.1. Information Theory

The study of information theory dates back to the groundbreaking work “A Mathematical Theory of Communication” by Shannon [1948]. In this article, Shannon investigated the minimum amount of bits needed to transmit a message via any communication system like telegraphs or nowadays the internet. When assuming noiseless communication, this quantity only depends on the probability distribution over the input messages.

Entropy

For a finite number N of possible messages, Shannon defined the entropy as

$$\mathrm{H}(p) = -\sum_{i=1}^{N} p_i \log p_i, \tag{1.1}$$

where p_i corresponds to the probability of generating the message i. Hence, the entropy is zero when always the same message is sent. When all messages have equal likelihood, the entropy obtains its maximum value at $\log(N)$. The entropy corresponds to the average amount of information contained in a message measured in *nats*. When the logarithm to base two is used, the information is measured in *bits*. Intuitively, the entropy defines the *randomness* or *unpredictability* of a probability distribution.

Differential Entropy

The differential entropy is an extension of the notion of entropy to continuous probability distributions $p(\boldsymbol{x})$ and is defined as

$$\mathrm{H}(p) = -\int_{x} p(\boldsymbol{x}) \log p(\boldsymbol{x}) d\boldsymbol{x}. \tag{1.2}$$

The differential entropy shares many properties with the discrete case and can also be regarded as a measure of randomness. However, because a change of the coordinate frame will in general affect the differential entropy, it measures the randomness relative to an assumed standard [Shannon, 1948] and not on an absolute scale as it is the case for the discrete entropy. For this reason, the differential entropy may become negative, whereas the discrete entropy is always non-negative. Throughout this thesis, we will use the term entropy both to refer to the discrete entropy and to the differential entropy; whether Equation 1.1 or Equation 1.2 needs to be applied can be decided based on the kind of probability distribution.

Relative Entropy

The relative entropy or Kullback-Leibler divergence relates to the similarity of two probability distributions. The KL divergence between two distributions $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ is defined as

$$D_{\mathrm{KL}}(p||q) = \int_{\boldsymbol{x}} p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} d\boldsymbol{x}.$$

For the discrete case, it can be analogously defined by replacing the integral by a summation. The KL divergence corresponds to the expected additional number of *nats* (or *bits* if $\log_2$ is used instead of $\log$) required to submit a message generated by $q(\boldsymbol{x})$, when using a code that was optimized for $p(\boldsymbol{x})$. It can also be understood as the average amount of information contained in a sample $\boldsymbol{x} \sim p$ for discriminating between $p(\boldsymbol{x})$ and $q(\boldsymbol{x})$ [Kullback and Leibler, 1951].

The KL divergence is always non-negative and is zero if and only if $p(\boldsymbol{x}) = q(\boldsymbol{x})$. Furthermore, the KL divergence is not symmetric, i.e., $D_{\mathrm{KL}}(p||q) \neq D_{\mathrm{KL}}(q||p)$. In machine learning, the KL divergence has an important application for comparing a learned model $p(\boldsymbol{x})$ to a target distribution $q(\boldsymbol{x})$. For such cases $D_{\mathrm{KL}}(q||p)$ is often denoted as *forward* KL and $D_{\mathrm{KL}}(p||q)$ as *reverse* KL.

1.1.2. Information-Geometric Optimization

The field of robot learning, and machine learning in general, heavily relies on probabilistic models. For example, consider the very general robot learning problem sketched earlier, where we want to find the parameters of a robot controller that maximizes some performance measure of the emerging behavior. As the performance usually also depends on interactions with the environment that we can not fully control (like interactions with humans), we usually need to model it probabilistically and optimize the *expected* performance. When tackling such optimization problems that involve probability distributions, information-theoretic quantities naturally lend themselves for formalizing objectives or constraints. For example, we can try to imitate a human by minimizing the relative entropy $D_{\mathrm{KL}}(p||q)$ between the robot's distribution of behaviors $p(\boldsymbol{x})$ and the (modeled) distribution $q(\boldsymbol{x})$ of the human behavior. Similarly, information-geometric trust regions can be defined by constraining the relative entropy between the updated behavior and the current behavior. In this section we will explore different ways of using entropies and relative entropies for framing objectives and constraints for optimization problems. The resulting insights will be used for formulating trust region I-projections, which will be used in the remainder of this thesis to derive novel algorithms for VI, RL and IRL.

M-Projection and I-Projection

In this section, we will examine two specific kinds of information-geometric optimization problems called moment projection and information projection.

Moment Projection Moment projection or M-projection refers to optimization problems of the form

$$\begin{aligned}\boldsymbol{\theta}^\star &= \arg\min_{\boldsymbol{\theta}} \quad \int_{\boldsymbol{x}} q(\boldsymbol{x}) \log \frac{q(\boldsymbol{x})}{p(\boldsymbol{x};\boldsymbol{\theta})} d\boldsymbol{x} \\ &= \arg\max_{\boldsymbol{\theta}} \quad \mathrm{H}(q) + \int_{\boldsymbol{x}} q(\boldsymbol{x}) \log p(\boldsymbol{x};\boldsymbol{\theta}) d\boldsymbol{x},\end{aligned}$$

that is, optimization problems that aim to minimize the forward KL $D_{\mathrm{KL}}(q||p)$ between a target distribution $q(\boldsymbol{x})$ and the optimized distribution $p(\boldsymbol{x};\boldsymbol{\theta})$. The entropy of the target distribution $\mathrm{H}(q)$ is independent of $\boldsymbol{\theta}$ and thus does not affect the optimal solution $\boldsymbol{\theta}^\star$ and can be ignored during the optimization. In general, the integral can not be solved in closed form, and thus a Monte Carlo estimate based on a finite number of N samples from the target distribution is often used to estimate its value, yielding the optimization problem

$$\boldsymbol{\theta}^\star = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log p(\boldsymbol{x}_i;\boldsymbol{\theta}),$$

which corresponds to maximum likelihood estimation (MLE).

Information Projection Information projection or I-projection refers to optimization problems of the form

$$\begin{aligned}\boldsymbol{\theta}^\star &= \arg\min_{\boldsymbol{\theta}} \quad \int_{\boldsymbol{x}} p(\boldsymbol{x};\boldsymbol{\theta}) \log \frac{p(\boldsymbol{x};\boldsymbol{\theta})}{q(\boldsymbol{x})} d\boldsymbol{x} \\ &= \arg\max_{\boldsymbol{\theta}} \quad \mathrm{H}(p) + \int_{\boldsymbol{x}} p(\boldsymbol{x};\boldsymbol{\theta}) \log q(\boldsymbol{x}) d\boldsymbol{x}\end{aligned}$$

that is, optimization problems that aim to minimize the reverse KL $D_{\mathrm{KL}}(p||q)$. Although, now both terms affect the solution $\boldsymbol{\theta}^\star$, it can be noted that the target distribution $q(\boldsymbol{x})$ does not need to be normalized as a rescaling of $q(\boldsymbol{x})$ only results in a constant offset in the objective function. Similar to the M-projection, either one or both terms of the objective function often need to be approximated from samples, which now need to be drawn from the model $p(\boldsymbol{x})$.

Comparison of M-Projections and I-Projections Interestingly, the M-projection and the I-projection do not merely define different measures of dissimilarity, but also require

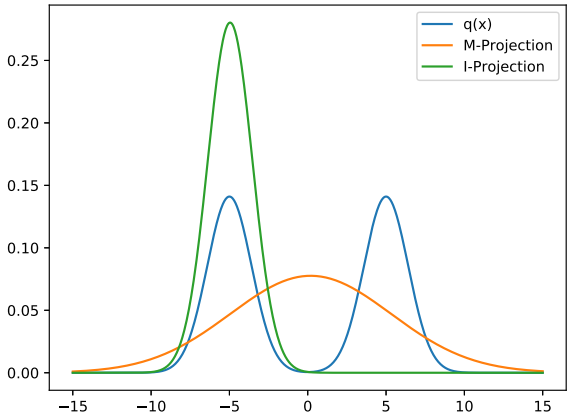


Figure 1.1.: The plot shows the M-projection (orange) and the I-projection (green) of a mixture of two Gaussian distributions (blue) onto the set of Gaussian distributions. Note that the green curve shows only one of two local optima of the I-projection.

different problem settings. The sample-based estimate of the M-projection requires samples from the target distribution but does not require evaluating $q(\boldsymbol{x})$. It is therefore suited for density estimation or regression problems. The sample-based estimate of the I-projection does not require samples from $q(\boldsymbol{x})$ but from the model $p(\boldsymbol{x})$ which are often easier to obtain. However, we need to be able to evaluate $\log q(\boldsymbol{x}) + c$ for an arbitrary constant c. It can thus be used to approximate intractable distributions for variational inference.

Yet, for some simple problems it is possible to solve both projections and it is interesting to compare the effect of the divergence on the learned solution. Figure 1.1 compares optimal solutions of the I-projection and M-projection for approximating a mixture of two Gaussian distributions with a single Gaussian distribution. The M-projection finds a distribution that covers both components of the mixture model. M-projections tend to match the moments of the target distribution and avoid putting low probability at regions where the target distribution has non-negligible mass. They are therefore also called *moment-matching* and *zero-avoiding*. I-projections on the other hand, avoid putting high probability at regions where the density of the target distribution is low and thereby focus on approximating individual modes of the target distribution. They are therefore also called *mode-seeking* and *zero-forcing* and have typically lower entropy compared to

M-projections. Intuitively, the different behavior can be explained based on the fact that the expectation is computed based on the model for I-projections and based on the target distribution for M-projections. Hence, regions where $p(\boldsymbol{x})$ is low will hardly affect the reverse KL of the I-projection, even if the approximation error in these regions is high. For M-projections, putting very little mass on some region R where the target distribution has non-negligible density would be heavily punished as $\int_R -q(\boldsymbol{x}) \log p(\boldsymbol{x}) dR$ would become very large.

In this thesis we will use the I-projection to derive novel algorithms for variational inference, reinforcement learning and inverse reinforcement learning because in these fields missing some solutions is usually much less severe than including bad solutions. Furthermore, the M-projection would be difficult to implement for variational inference and reinforcement learning, because samples from the target distribution are usually not available. However, for inverse reinforcement learning, the M-projection would be more straightforward, because here we typically have samples from an expert but do not know the generating distribution. Still, we will show that efficient algorithms for IRL can be derived based on I-projections.

Sample-Based Optimization of I-Projections Stochastic gradient descent is a standard approach to solve optimization problems where the objective function can only be approximated by samples. Stochastic gradient descent uses samples from the current model $p(\boldsymbol{x}; \boldsymbol{\theta}^{(\mathrm{i})})$ to approximate the gradient of the objective function which is then used to update the iterate. Such stochastic gradient-based optimization has been used for I-projections in variational inference [Gershman et al., 2012; Miller et al., 2017; Ranganath et al., 2014] as well as for similar problems in reinforcement learning [Kakade, 2001; Williams, 1992]. However, the Monte-Carlo estimates of the gradient often have high variance. Hence, the step size has to be chosen very small in order to ensure convergence resulting in a large number of iterations and thus demanding many samples. This problem has been identified in both communities and was addressed using various variance reduction techniques such as control variates (baselines) [Kimura and Kobayashi, 1997; Ranganath et al., 2014; Williams, 1992], Rao-Blackwellization [Ranganath et al., 2014] and the reparametrization trick [Heess et al., 2015; Kingma and Welling, 2013; Miller et al., 2017]. However, in spite of sophisticated variance reduction techniques, gradient-based optimization often still yields unstable and slow convergence for these optimization problems.

Due to the bad performance of vanilla gradient descent, information-geometric trust-region method have become popular in the field of reinforcement learning. Instead of updating the parameters $\boldsymbol{\theta}$ along its approximated gradient, information-geometric trust region methods aim to find the optimal parameters while changing the resulting

distribution $p(\boldsymbol{x};\boldsymbol{\theta})$ only slightly at each iteration. By ensuring that the current distribution $p(\boldsymbol{x};\boldsymbol{\theta})$–which is used to obtain samples to approximate the objective function–changes only slightly, the approximation is more likely to remain valid. Still, by learning the optimal parameters within this region of trust, relatively large progress with respect to the objective is made, leading to faster convergence and better sample efficiency in comparison to vanilla gradient descent. In reinforcement learning, the trust-region constraint is often implemented by bounding the reverse KL divergence between the learned controller and the last (stochastic) controller [Abdolmaleki et al., 2015; Levine and Abbeel, 2014; Peters et al., 2010; Schulman et al., 2015]. Technically, it would be more sound to bound the KL divergence between the resulting behavior $p(\boldsymbol{x})$ directly which, as shown by Abdulsamad et al. [2017], can indeed result in more efficient updates. However, accomplishing a trust-region on the resulting behavior is substantially more difficult than accomplishing a trust-region on the controller, and thus the latter constraint is often preferred. Albeit the great success of trust region optimization in the field of reinforcement learning, its application to variational inference has been little explored.

Trust Region I-Projections

Motivated by the success of trust regions for reinforcement learning, this thesis explores this technique in the context of I-projection problems. More specifically, the algorithms proposed within this thesis center on a scheme of optimization problems that have the form

$$\begin{aligned} \boldsymbol{\theta}^{(i+1)} = \quad & \underset{\boldsymbol{\theta}}{\arg\min} \quad \widetilde{J}(\boldsymbol{\theta}) \\ & \text{subject to} \quad D_{\mathrm{KL}}\left(p(\boldsymbol{x};\boldsymbol{\theta})||p(\boldsymbol{x};\boldsymbol{\theta}^{(i)})\right) \leq \epsilon, \end{aligned}$$

where ϵ specifies the size of the trust region and $\widetilde{J}(\boldsymbol{\theta})$ is a sample based approximation of the reverse KL $D_{\mathrm{KL}}\left(p(\boldsymbol{x};\boldsymbol{\theta})||q(\boldsymbol{x})\right)$.

The reverse KL is also used for defining the trust region, because it is in general safer by preventing updates that put probability mass at regions where the last distribution had low density and the sample-based model $\widetilde{J}(\boldsymbol{\theta})$ can not be trusted. A drawback of the reverse KL, however, is that it tends to decrease the entropy of $p(\boldsymbol{x};\boldsymbol{\theta})$ quickly, which can lead to bad local optima. We typically alleviate this problem, by adding an objective for maximizing the entropy, i.e.,

$$\begin{aligned} \boldsymbol{\theta}^{(i+1)} = \quad & \underset{\boldsymbol{\theta}}{\arg\min} \quad \widetilde{J}(\boldsymbol{\theta}) - \beta \mathrm{H}(p(\boldsymbol{x};\boldsymbol{\theta})) \\ & \text{subject to} \quad D_{\mathrm{KL}}\left(p(\boldsymbol{x};\boldsymbol{\theta})||p(\boldsymbol{x};\boldsymbol{\theta}^{(i)})\right) \leq \epsilon, \end{aligned}$$

where β is a coefficient for trading off the two terms in the objective and can be decreased towards zero during optimization.

1.2. Trust Region I-Projections for Robot Learning and Variational Inference

In this thesis we will discuss novel algorithms that have been derived by establishing information-geometric trust-region optimization to the field of variational inference and by unifying the robot learning problems of reinforcement learning (RL), imitation learning (IL) and inverse reinforcement learning (IRL) by framing them as a common I-projection problem.

We will now briefly introduce these learning problems and sketch how they can be framed as trust region I-projections. The specific problem formulations and the resulting algorithms are described in their respective chapters.

1.2.1. Trust Region I-Projections for Variational Inference

Variational inference addresses the fundamental problem in machine learning of performing inference on intractable distributions. For example, in Bayesian inference the posterior distribution is often intractable because its normalizer can not be computed in closed form. For a very general formulation of VI, the desired distribution $q(\boldsymbol{x})$ has the form

$$q(\boldsymbol{x}) = \tilde{q}(\boldsymbol{x})/Z,$$

where the normalizing constant $Z = \int_{\boldsymbol{x}} \tilde{q}(\boldsymbol{x}) d\boldsymbol{x}$ can not be evaluated and only unbiased estimates of $\tilde{q}(\boldsymbol{x})$ can be obtained for any given $\boldsymbol{x}$. Yet, it is often desirable to draw samples from $q(\boldsymbol{x})$, for example in order to estimate an expected value of interest. In order to accomplish this goal, variational inference learns a tractable approximation $p(\boldsymbol{x})$ of $q(\boldsymbol{x})$ and uses it to perform inference. The optimization problem is commonly framed as an I-projection,

$$p^\star(\boldsymbol{x}) = \arg\min_{p(\boldsymbol{x})} \; \mathbb{E}_p \left[\log \frac{p(\boldsymbol{x})}{\tilde{q}(\boldsymbol{x})} \right], \tag{1.3}$$

which exploits that Z enters the optimization problem as a constant offset and thus does not affect the solution. However, in order to make the optimization given by Equation 1.3 feasible, $p(\boldsymbol{x})$ is often restricted to belong to a simple family of models, for example multivariate normal distributions [Regier et al., 2017] or is assumed to have non-correlating

degrees of freedom [Blei et al., 2017; Peterson and Hartman, 1989], which is known as the mean field approximation.

In order to allow for robust learning of more challenging, multimodal approximations, variational inference can be framed as trust region I-projection by adding a trust-region constraint to the optimization problem (Eq. 1.3), e.g.,

$$\begin{aligned} p^\star(\boldsymbol{x}) = \quad & \underset{p(\boldsymbol{x})}{\arg\min} \quad \mathbb{E}_p \left[\log \frac{p(\boldsymbol{x})}{\tilde{q}(\boldsymbol{x})} \right] \\ & \text{subject to} \quad D_{\text{KL}}(p||p_{\text{last}}) \leq \epsilon, \end{aligned} \tag{1.4}$$

where $p_{\text{last}}(\boldsymbol{x})$ is the approximation that has been learned during the last iteration and ϵ is an appropriately chosen upper bound.

1.2.2. Trust Region I-Projections for Reinforcement Learning

Manually programming robots to perform complex tasks can be very cumbersome or even infeasible. Reinforcement learning aims to reduce the burden on the human by letting the robot learn the desired behavior by trial and error. In RL, the human only needs to specify a reward function $r_t(\boldsymbol{s}, \boldsymbol{a})$ for each time step t that computes a (scalar) score based on the current state of the robot, $\boldsymbol{s}$, and its chosen action $\boldsymbol{a}$. If the reward function is chosen appropriately, the robot can learn to perform the desired task by iteratively adapting its policy $\pi_t(\boldsymbol{a}|\boldsymbol{s})$ in order to increase the expected reward.

The reinforcement learning problem with finite time horizon T can be formulated as

$$\underset{\pi}{\text{maximize}} \quad \sum_{t=0}^{T} \mathbb{E}_{p_t^\pi} \left[r_t(\boldsymbol{s}, \boldsymbol{a}) \right], \tag{1.5}$$

where the expectation is computed with respect to the distribution $p_t^\pi(\boldsymbol{s}, \boldsymbol{a})$ that results from applying the policy π.

As the objective function typically can not be computed in closed-form, the policy optimization has to be performed based on samples that are obtained by executing a policy. In order to prevent overfitting to the samples, trust-region constraints have been successfully applied both, for model-free [Abdolmaleki et al., 2015; Peters et al., 2010; Schulman et al., 2015] and for model-based [Abdulsamad et al., 2017; Levine and Abbeel, 2014] RL methods.

Furthermore, the optimization can be prone to premature convergence, because reducing the variance of the policy is often effective for preventing bad behavior and thus increasing the expected reward. In order to address the problem of premature collapsing of the

policy, Williams and Peng [1991] added an objective that rewards high entropy. This technique is recently becoming increasingly popular [Abdolmaleki et al., 2015; Akrour et al., 2016; Mnih et al., 2016; O'Donoghue et al., 2016]. While the aforementioned work only considered the entropy of the policy, it is interesting to investigate the effect of rewarding the entropy of the robot behavior. Let $\boldsymbol{x}$ be a vector that concatenates the states and actions for all time steps and therefore has size $T(D_s + D_a)$, where D_s and D_a refer to the dimensionality of the states and actions respectively. We can rewrite the objective given by Equation 1.5 in terms of the behavior $\boldsymbol{x}$ and add a trust-region constraint and entropy objective, i.e.,

$$\begin{aligned} \pi^\star(\boldsymbol{a}|\boldsymbol{s}) = \quad & \underset{\pi(\boldsymbol{a}|\boldsymbol{s})}{\arg\max} \quad \beta \mathrm{E}_{p^\pi(\boldsymbol{x})}\left[r(\boldsymbol{x})\right] + \mathrm{H}\left(p^\pi(\boldsymbol{x})\right) \\ & \text{subject to} \quad D_{\mathrm{KL}}(p^\pi(\boldsymbol{x})||p_{\text{last}}(\boldsymbol{x})) \leq \epsilon, \end{aligned} \tag{1.6}$$

where the reward function with respect to the behavior, $r(\boldsymbol{x})$, can be chosen to coincide with the sum over the time-dependent rewards $r_t(\boldsymbol{s}, \boldsymbol{a})$, and

$$\mathrm{H}\left(p^\pi(\boldsymbol{x})\right) = -\int_{\boldsymbol{x}} p^\pi(\boldsymbol{x}) \log p^\pi(\boldsymbol{x}) d\boldsymbol{x}$$

refers to the entropy of the distribution over robot behavior. Note that the coefficient β was introduced in order to trade off the two objectives. Maximizing the entropy of the resulting behavior rather than the entropy of the controller outputs can lead to policies that exhibit more versatile behavior. Furthermore, the optimization problem given by Equation 1.6 may lead to more favorable exploration, because different solution strategies are actively explored.

When rewriting the reward optimization problem as KL divergence minimization and defining $\tilde{q}(\boldsymbol{x}) = \exp\left(\beta r(\boldsymbol{x})\right)$, Optimization Problem 1.6 is very similar to a trust region I-projection, namely

$$\begin{aligned} \pi^\star(\boldsymbol{a}|\boldsymbol{s}) = \quad & \underset{\pi(\boldsymbol{a}|\boldsymbol{s})}{\arg\min} \quad \mathrm{E}_{p^\pi(\boldsymbol{x})}\left[\log \frac{p^\pi(\boldsymbol{x})}{\tilde{q}(\boldsymbol{x})}\right] \\ & \text{subject to} \quad D_{\mathrm{KL}}(p^\pi(\boldsymbol{x})||p_{\text{last}}(\boldsymbol{x})) \leq \epsilon. \end{aligned} \tag{1.7}$$

However, an additional challenge of the reinforcement learning problem stems from the fact that we can not directly specify the distribution of the emerging behavior $p^\pi(\boldsymbol{x})$. Instead, we can only control it indirectly by choosing the policy $\pi^\star(\boldsymbol{a}|\boldsymbol{s})$. The reformulation (Eq. 1.7) reveals, that optimizing Equation 1.6 aims to match the desired distribution $q(\boldsymbol{x}) = \tilde{q}(\boldsymbol{x})/Z_q$ as closely as possible with respect to the KL divergence. Consequently, the

human who is designing the reward function can better predict the resulting behavior. This is an important benefit of maximizing the entropy with respect to the distribution over behavior because reward function engineering can be very cumbersome for the human. For example, consider the task of putting a peg into a hole in a wall. An obvious reward function for this task produces a constant positive reward, if the behavior results in placing the peg in the hole and zero reward otherwise. However, such sparse reward function is very difficult to optimize by trial-and-error because it does not produce any guidance unless the robot manages by chance to correctly place the peg. A more sensible reward functions would penalize the euclidean distance between the peg and the hole, for example by using a concave quadratic function. However, it can still become difficult to appropriately set the coefficients of the reward function, especially when additional objectives need to be considered, for example, energy efficiency. If the coefficients are too low, it might not be worthwhile to correctly solve the task. On the other hand, other objectives might be neglected if the coefficients are too high. Reinforcement learning, therefore usually also involves trial-and-error by the human, who has to repeatedly adapt the reward function and observe the resulting behavior. While not fully resolving this labor, specifying a desired distribution over behaviors can ease the task of reward engineering. For example, the human could specify a reward distribution that is Gaussian with respect to the position of the peg. By setting the mean position to the center of the hole and the covariance matrix appropriately with respect to the size of the hole, iteratively optimizing Equation 1.6 will result in the desired behavior if possible or otherwise trade off conflicting objectives in an information-geometrically optimal way.

1.2.3. Trust Region I-Projections for Imitation Learning

Imitation learning (IL) [Argall et al., 2009; Hussein et al., 2017; Osa et al., 2018; Schaal, 1999] is a different approach to facilitate the task of programming robots. Instead of specifying a reward function, the human can demonstrate the desired behavior, either by letting the robot passively observe the human demonstrator [Koert et al., 2016; Kulić et al., 2012; Kuniyoshi et al., 1994; Maeda et al., 2016], or by physically moving the robot directly using kinesthetic teaching [Ewerton et al., 2016; Kormushev et al., 2011] or indirectly using teleoperation [Abbeel et al., 2010].

The trust region I-projection formulation (Eq. 1.7) for RL can also be applied to imitation learning, however the desired distribution $\tilde{q}(\boldsymbol{x})$ is only represented via samples of the expert. A naive solution would be to use either a non-parametric model or to fit a parametric model via maximum likelihood estimation (MLE) (and hence an M-projection) and subsequently perform the I-projection based on this model. However, MLE is prone to overfitting and also non-parametric approaches like kernel density estimation (KDE) [Parzen,

1962; Rosenblatt, 1956] have difficulties in extrapolating the data—especially for high-dimensional spaces. Estimating the distribution over expert behaviors is, however, not necessary, since the objective function only depends on the density ratio $p^{\pi}(\boldsymbol{x})/\tilde{q}(\boldsymbol{x})$. Density ratio estimation based on samples from the robot behavior and the expert behavior is a simpler problem, because the density ratio can be computed from the densities, but the densities can not be recovered from the density ratio [Sugiyama et al., 2012].

1.2.4. Trust Region I-Projections for Inverse Reinforcement Learning

Similar to imitation learning, inverse reinforcement learning uses expert demonstrations for specifying the task. However, the goal of IRL is not primarily to learn a policy $\pi(\boldsymbol{a}|\boldsymbol{s})$, but rather to learn a reward function $r(\boldsymbol{s}, \boldsymbol{a})$ that, when optimized using reinforcement learning, leads to the desired policy. Intuitively, IL aims to learn *how* to exhibit the desired behavior without explicitly modeling its intentions, whereas IRL aims to infer *why* the demonstrators acted in the way they did. By explicitly modeling the intentions, reward functions may remain valid across different environments that necessitate different solution strategies. Furthermore, reward functions can be useful on its own, i.e., also when not used in conjunction with RL. They enable the robot to detect when its performance decreases or can be used for modeling and predicting the behavior of other agents [Shimosaka et al., 2015; Ziebart et al., 2009]. However, reward functions do not capture intentions per se, but may also reward specific solution strategies. Learning such shaped reward functions has indeed little benefit over learning policies and ensuring generalizable reward functions is arguable the biggest open problem in the field of IRL. Recently, Fu et al. [2018] proposed to learn reward functions that disregard the robot actions in order to increase the likelihood of making them generalizable. Still, independence of the applied actions–while often necessary–is not sufficient to capture the expert intentions.

Early approaches for IRL [Ng and Russell, 2000; Ratliff et al., 2006; Ziebart et al., 2008] required to solve the (forward) RL problem iteratively, and it appeared that the inverse reinforcement learning problem was therefore significantly harder than RL. As noted by Ho and Ermon [2016], such approaches correspond to using dual ascent for imitation learning, where iteratively the policy (primal) is optimized for a given dual value (cost) and the cost is updated based on the resulting violations of the constraint of matching the expert's behavior. However, state-of-the-art IRL methods apply more efficient optimization techniques, that interleave policy and reward optimization [Finn et al., 2016b; Fu et al., 2018]. By performing a single RL optimization with a varying reward function, these methods can perform IRL as efficient as IL and comparably efficient to RL. This optimization is reminiscent to the way generative adversarial nets (GANs) [Goodfellow et al., 2014] are optimized [Finn et al., 2016a].

The strong relationship between imitation learning and inverse reinforcement learning can also be observed for the trust region I-projection formulation of imitation learning, by examining its Lagrangian

$$\mathcal{L}(\pi) = -\mathbb{E}_{p^\pi(\boldsymbol{x})}\left[\log \tilde{q}(\boldsymbol{x}) - (1+\eta)\log p^\pi(\boldsymbol{x}) + \eta \log p_{\text{last}}(\boldsymbol{x})\right] + \eta\epsilon,$$

where $\eta \geq 0$ is the Lagrangian multiplier corresponding to the trust region constraint. Trust region I-projection for imitation learning–and thereby also trust region I-projection for RL–thus corresponds to (standard) reinforcement learning with a varying reward function

$$r^\pi(\boldsymbol{x}) = \log \tilde{q}(\boldsymbol{x}) - (1+\eta)\log p^\pi(\boldsymbol{x}) + \eta \log p_{\text{last}}(\boldsymbol{x}).$$

For imitation learning, this reward function converges towards a reward function that matches the expert behavior and, hence, solves the IRL problem.

1.2.5. Challenges of the Applications

The aforementioned applications of VI, RL and IL/IRL have several challenges related to multimodality, discontinuities, stability, local optima and time series. Figure 1.2 contains a list of typical challenges for each application.

Multimodality When the target distributions has multiple modes that all need to be found, the approximation needs to be appropriately complex, which typically makes the optimization more difficult. Furthermore, careful exploration is required in order to discover all relevant modes.

Local Optima When the optimization problem has several local optima, it can become difficult to find good solutions. Hence, exploration in the parameter space can also become challenging.

Discontinuities Discontinuities can lead to high modeling errors even in the vicinity of the samples and thereby makes stable optimization more challenging. In order to approximate such discontinuities, the approximation needs to be highly nonlinear.

Stability Especially, when varying samples are required to estimate the objective function, it can become difficult to ensure stable converge to a solution.

	Multi-Modality	Local Optima	Discontinuities	Stability	Time Series
variational inference	✓	✓	✓	✓	
reinforcement learning		✓	✓	✓	✓
inverse reinforcement learning	✓	✓	✓	✓	✓

Figure 1.2.: The table shows typical challenges of variational inference, reinforcement learning and inverse reinforcement learning. Note that for reinforcement learning we often settle with a single good solution. However, when we want to learn versatile behavior, multimodality can also become a major challenge for RL.

Time Series For reinforcement learning and inverse reinforcement learning, the objective function depends on trajectories over time. The relation between the parameters of the robot controller and the resulting time series can only be approximately modeled and such models are typically highly nonlinear.

1.3. Summary of Contributions

This thesis presents new methods and theoretical insights related to information-geometric trust region optimization of I-projection problems.

Namely, we will derive a method for trust region I-projections that unifies the problems of IRL and RL and a trust-region based method for variational inference that is suitable for fitting GMM approximations. We will further examine how a KL-proximal I-projection that is similar to an information-geometric trust region can be used to derive non-adversarial methods for imitation learning that are very similar to modern adversarial imitation learning (AIL) methods.

1.3.1. Model-based Trust Region I-Projections for Unifying RL and IRL

We unify RL and IRL by framing both problems as a common I-projection problem. For IRL, the target distribution is computed by using the M-projection to fit a Gaussian distribution over feature trajectories from expert demonstrations. For RL we assume a reward function that is quadratic in features. By using iterative linearizations of the features and the system dynamics, the trust-region I-projections can be efficiently solved for learning time-dependent linear control policies and quadratic reward functions. We show that the I-projection formulation is several orders of magnitude more efficient than solving the same problem with the very common maximum causal entropy IRL approach (MaxCausalEnt-IRL). We also show that this method can be used to intuitively specify tasks by treating reward functions as desired trajectory distributions.

1.3.2. Trust Region I-Projections for Variational Inference

Many methods for approximating intractable distributions for variational inference are restricted to simple model families, e.g., by using the mean-field approximation, in order to make the optimization easier. Such methods are therefore not suited for approximating complex, multimodal distributions. Instead, we will derive a method for learning GMM approximations with an adaptive number of components. Such GMM models can be used to approximate the target distribution arbitrary well. In order to make the optimization feasible, a method similar to expectation-maximization (EM) is derived for decomposing the trust-region I-projection into tractable subproblems that are optimized by treating them as reinforcement learning problems. The resulting method is capable of learning approximations of higher quality than any previous VI method and the quality of samples from the approximation are on par with samples created by MCMC methods that have been devoted two to three orders of magnitude more computational time.

1.3.3. Non-Adversarial Imitation Learning by KL-Proximal I-Projections

Although our model-based approach to unify RL and IRL has several appealing properties, it is limited to learning linear policies and only learns quadratic reward functions. In contrast, many modern methods for imitation learning scale to neural network policies and reward functions. These methods are often based on an adversarial formulation that was inspired by generative adversarial nets (GANs, Goodfellow et al. 2014). We show that the I-projection formulation of our model-based approach can be efficiently solved also for nonlinear function approximators when lower-bounding the Lagrangian multiplier of the trust-region by one. The resulting algorithms, however,—albeit non-adversarial

in nature—are very similar to existing adversarial approaches. While it is difficult to claim algorithmic advantages compared to existing methods due to the strong similarities, our non-adversarial formulation enjoys stronger theoretical guarantees and yields new insights. For example, we show that adversarial inverse reinforcement learning (AIRL, Fu et al. 2018), which previously was not theoretically well-understood, can indeed be derived from our non-adversarial formulation. However, we also demonstrate that the non-adversarial formulation can be used to derive novel algorithms by presenting offline non-adversarial imitation learning (O-NAIL). O-NAIL considers the offline setting, that is, we do not assume that the agent can interact with the environment for learning a policy. O-NAIL was inspired by the recent (adversarial) offline imitation learning method ValueDice [Kostrikov et al., 2020] and differs by using an actor-critic [Konda and Tsitsiklis, 2000] optimization instead of solving a saddle point problem, which results in stronger convergence guarantees as we can show convergence also for large policy improvements.

1.4. Thesis Outline

Each of the aforementioned contributions is devoted an individual chapter that can be read independently.

Chapter 2 describes our model-based unification of reinforcement learning and inverse reinforcement learning. The chapter was published at the international conference on intelligent robots and systems [Arenz et al., 2016][1].

In Chapter 3, we present our trust region based method for variational inference. Except for Section 3.6, this chapter is based on an article that was published by the journal of machine learning research [Arenz et al., 2020]. Preliminary work for this chapter was published at the international conference on machine learning [Arenz et al., 2018]. Section 3.6 briefly discusses follow-ups by Ewerton, Arenz, and Peters [2020] and Becker, Arenz, and Neumann [2020]. Although the research for these works was primarily conducted by the corresponding first authors, these works are very relevant to the scope of this thesis by presenting a robotic application of our variational inference method [Ewerton et al., 2020], and by reducing the gap between variational inference and imitation learning by applying ideas presented in Chapter 3 to density estimation [Becker et al., 2020].

Chapter 4 discusses our non-adversarial formulation for imitation learning and its connection to adversarial methods. This chapter is based on a preprint [Arenz and Neumann, 2020] that is currently under review by the journal of machine learning research.

[1] ©2016 IEEE. Reprinted, with permission, from Oleg Arenz, Hany Abdulsamad and Gerhard Neumann, Optimal Control and Inverse Optimal Control by Distribution Matching, IROS, 2016.

Text and images of the next chapter were taken from prior work [Arenz et al., 2016].©2016 IEEE. Reprinted, with permission, from Oleg Arenz, Hany Abdulsamad and Gerhard Neumann, Optimal Control and Inverse Optimal Control by Distribution Matching, IROS, 2016.

2. Inverse Reinforcement Learning by Matching Distributions

Optimal control [Kappen, 2007; Stengel, 1986] aims at finding optimal behavior given a cost function and the dynamics of the system. Typically, the cost function consists of several objectives that need to be traded off by the experimenter. Finding the correct trade-off often requires fine-tuning until the optimal behavior matches the desired behavior of the experimenter. Conversely, inverse reinforcement learning (IRL) algorithms [Ng and Russell, 2000] aim at finding a reward function for a Markov decision problem (MDP) that is consistent with observed expert demonstrations. It can be used for inferring the expert's goals as well as for apprenticeship learning. By learning a reward function from demonstrations rather than learning a policy directly, a succinct and feature based task representation is learned, that generalizes to different situations and is unaffected by changes in the dynamics.

In this paper, we approach the problem of optimal control and inverse optimal control in the same framework by matching the induced state distribution of the policy with the desired state distribution or—for inverse optimal control—the observed state distribution of the experimenter. We will focus on the stochastic trajectory optimization case where the desired distributions are given by Gaussian distributions over trajectories. Instead of defining a quadratic cost function as it is the case for optimal control, we specify a desired mean state and a desired accuracy for reaching the mean state. In addition, we can specify the importance of matching the desired distribution for each time step, i.e., we can specify desired mean state and accuracy only for a subset of the time steps. For optimal control, the desired mean state and accuracy is chosen by the experimenter for a subset of the time steps, while in the inverse RL case, the desired distribution is typically estimated from the experimenter. Our approach implicitly estimates a time-dependent quadratic reward function such that the optimal control policy that is obtained from maximizing this reward function matches the desired trajectory distribution.

In our approach, the trajectory distribution can be given either in joint or in operational space. Our objective is now to estimate a controller that matches the desired trajectory distribution. In order to realize this objective, we minimize the relative entropy or Kullback-

Leibler (KL) divergence between the distribution induced by the policy and the desired distribution. This KL minimization procedure has a strong resemblance to existing inverse reinforcement learning algorithms such as maximum causal entropy IRL [Ziebart et al., 2010] if we match the first and second order moments as features of the trajectory for each time step, i.e., when matching mean and variance. Matching the variance in addition to the mean is often sensible when inferring the expert's goal, given that it is an important indicator for the importance of accurate control. However, using existing IRL algorithms is computationally very expensive as the number of parameters grow linearly with the number of time steps and even quadratically with the dimensionality of the system. Using our KL minimization objective, we develop a new update rule for obtaining the parameters of the reward function and show that this update rule can be several orders of magnitudes more efficient than the standard gradient descent update rule. Moreover, many IRL algorithms [Levine and Koltun, 2012; Levine et al., 2011] need to observe the actions in the demonstrations. This assumption is often unrealistic, for example when we get the demonstrations by observing a human expert.

Similar to most trajectory optimization algorithms, we formulate our algorithm with linear system dynamics to make the estimation of the policy feasible in closed form. For non-linear systems, we introduce an incremental procedure based on linearizations similar to the well-known incremental LQG algorithm [Li and Todorov, 2004]. In order to ensure stability of the policy update, we follow the trajectory optimization approach from Levine and Abbeel [2014] used for guided policy search. In addition to minimizing the KL to the desired distribution over task space positions, we also limit the KL to the old policy, which has been used for obtaining the linearization of the system. This KL-bound ensures stability of the iterative optimization.

We evaluate our method on optimal control and inverse optimal control problems and compare our algorithm to competing IRL approaches. We show that our novel update direction outperforms the standard update direction from MaxCausalEnt-IRL. Moreover, we compare our approach to an IRL approach by Yin et al. [2014] on a handwritten letter trajectory data set and to an approach by Englert et al. [2013] on a pendulum swing-up. Finally, we show the applicability of our approach on a non-linear four-link pendulum that needs to achieve a peg-in-the-hole task with a given accuracy.

2.1. Related Work

The first IRL approaches [Ng and Russell, 2000] formulated the problem of obtaining the reward function as linear optimization problem, where the reward, that is assumed to be linear in some features, should be maximal for the demonstrated trajectories. As pointed

out by Ng and Russell [2000], the inverse RL problem is ill-defined and many reward functions exists that satisfy this criterion. Ziebart et al. [2008] introduced a max-entropy formulation for inverse reinforcement learning that resolves the ill-posedness based on the principle of maximum entropy for estimating distributions [Jaynes, 1957]. We will discuss the MaxCausalEnt-IRL algorithm in more details in the preliminaries. While the standard MaxCausalEnt-IRL algorithms all require a model of the system dynamics to perform dynamic programming, Boularias et al. [2011] proposed a model-free variant that uses reinforcement learning to obtain the optimal policies that are induced by a given reward function.

Another IRL algorithm that is based on maximum entropy IRL and on local trajectory optimization has been introduced by Levine et al. [2011]. The algorithm computes a reward function that renders the demonstration locally optimal by using second order Taylor approximations of the learned reward function and linearizations of the system dynamics. Yet, this algorithm does not take stochastic system dynamics into account and can not be directly used to estimate a time-dependent reward function for the trajectories. Furthermore, the method has to observe the actions in the demonstrations, which is not the case for our approach. Another IRL approach that tries to estimate a similar, time-dependent reward function was presented by Yin et al. [2014]. The authors use stochastic optimization to obtain the parameters of the reward function by optimizing the max-entropy IRL objective. Subsequently, an LQR solution is used to obtain the optimal controller. However, as the maximum entropy objective does not consider the stochasticity of the system, the resulting controller does not match the desired features. Consequently, the estimated reward function also does not fully explain the expert's behavior. We will compare our approach to the one by Yin et al. [2014] in the experiment section.

There are many stochastic trajectory optimization techniques that rely on linearizations, including the incremental LQG algorithm [Li and Todorov, 2004], AICO [Toussaint, 2009], Robust Policy Updates [Rueckert et al., 2014] and the algorithm used for guided policy search [Levine and Abbeel, 2014]. To our experience, the approach presented by Levine and Abbeel [2014] is the most stable one as it uses a KL bound to the old policy to stabilize the policy update.

Englert et al. [2013] presented a model-based approach to imitation learning that shares a similar objective to our approach. They modified the model-based policy search algorithm PILCO [Deisenroth and Rasmussen, 2011] such that it minimizes the KL to the distribution of the demonstrator instead of maximizing the reward. While our objective is similar, we obtain a closed form solution for linear feedback controllers with our approach while Englert et al. [2013] obtain a highly non-linear policy by performing a computationally heavy, non-convex optimization.

2.2. Preliminaries

This paper focuses on finite-horizon Markov decision processes (MDPs). A finite-horizon MDP is a 5-Tuple $\left(\boldsymbol{s}, \boldsymbol{a}, p_t(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}), r_t(\boldsymbol{s}, \boldsymbol{a}), T\right)$ where $\boldsymbol{s}$ denotes a vector of states, $\boldsymbol{a}$ denotes a vector of actions and T denotes the time horizon. The reward function at time step t is denoted by $r_t(\boldsymbol{s}, \boldsymbol{a})$ and the system dynamics by $p_t(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$.

We define $\boldsymbol{y}_t = f(\boldsymbol{s}_t, \boldsymbol{a}_t)$ as the task space position at time t and f as the task space transformation. We want to match the observed task space distribution of the demonstrator with the task space distribution that is induced by the policy. The task space can for example be defined by the forward kinematics of the end-effector. If we want to match the observed states and actions, the task-space transformation is defined by the identity function.

We define the feature vector $\boldsymbol{\psi}(\boldsymbol{y})$ as the linear and quadratic expansion of the task space vector $\boldsymbol{y}$. This expansion is needed to match the first and second order moments of the distribution $p(\boldsymbol{y})$ over the task space. For time-dependent vectors or functions, a subscript (usually t) is used to refer to a given time step while dropping the subscript refers to every time step.

Our method is based on the Kullback-Leibler divergence, or relative entropy, between two distributions $p_t(\boldsymbol{y})$ and $q_t(\boldsymbol{y})$, given by

$$D_{\mathrm{KL}}(p_t(\boldsymbol{y})||q_t(\boldsymbol{y})) = \int_{\boldsymbol{y}} p_t(\boldsymbol{y}) \log \frac{p_t(\boldsymbol{y})}{q_t(\boldsymbol{y})} \, d\boldsymbol{y}$$

and the conditional, differential entropy of a policy $\pi_t(\boldsymbol{a}|\boldsymbol{s})$, given by

$$H(\pi_t(\boldsymbol{a}|\boldsymbol{s})) = -\int_{\boldsymbol{s}} p_t(\boldsymbol{s}) \int_{\boldsymbol{a}} \pi_t(\boldsymbol{a}|\boldsymbol{s}) \log \pi_t(\boldsymbol{a}|\boldsymbol{s}) \, d\boldsymbol{a} \, d\boldsymbol{s},$$

where $p_t(\boldsymbol{s})$ denotes the distribution over states at time step t.

2.2.1. Maximum Causal Entropy IRL

Maximum causal entropy IRL (MaxCausalEnt-IRL) [Ziebart et al., 2010] aims at finding a reward function such that the resulting policy produces trajectories that are close to the expert's demonstrations. Following Abbeel and Ng [2004], closeness is measured by comparing the expected feature counts when following the policy with the empirical feature counts of the expert, given by

$$\hat{\boldsymbol{\psi}}_t = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \boldsymbol{\psi}_t(\boldsymbol{y}_{i,t}),$$

where $|\mathcal{D}|$ denotes the number of demonstrated trajectories and $\boldsymbol{y}_{i,t}$ denotes the task position achieved by demonstration i at time step t. In the basic formulation, the feature counts are matched exactly, however, a small error based on the ℓ_1 or ℓ_2 norm is often allowed when regularization is needed.

Since the expert's demonstrations are usually not fully optimal, and we have limited data to estimate the average feature counts, an optimal policy for the given MDP is not the right basis when comparing the feature counts. Indeed, the soundness of such approaches depends on how good the expert is modeled when computing a policy for a given reward function. MaxCausalEnt-IRL models the expert using the policy that has maximum entropy among all policies that are able to match the feature expectations and thereby does not make any ungrounded assumptions on the expert's policy.

The corresponding optimization problem can be written as

$$\max_{\pi_t(\boldsymbol{a}|\boldsymbol{s})} \sum_{t=1}^{T-1} H(\pi_t(\boldsymbol{a}|\boldsymbol{s})) \text{ s.t. } \forall_{t>1} : \int_{\boldsymbol{y}} p_t^{\pi}(\boldsymbol{y}) \psi_t(\boldsymbol{y}) d\boldsymbol{y} = \hat{\psi}_t,$$

where additional constraints specify $p_t^{\pi}(\boldsymbol{y})$ as the distribution over task space positions at time step t that results from applying the given state transformation $p_t(\boldsymbol{y}|\boldsymbol{s}, \boldsymbol{a})$. Furthermore, we need to ensure that the state distribution $p_t^{\pi}(\boldsymbol{s})$ is consistent with the previous policy $\pi_{t-1}(\boldsymbol{a}|\boldsymbol{s})$, system dynamics $p_{t-1}(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$ as well as the previous state distribution $p_{t-1}^{\pi}(\boldsymbol{s})$, i.e.,

$$p_t^{\pi}(\boldsymbol{s}') = \int_{\boldsymbol{s}} p_{t-1}(\boldsymbol{s}) \int_{\boldsymbol{a}} \pi_{t-1}(\boldsymbol{a}|\boldsymbol{s}) p_{t-1}^{\pi}(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) d\boldsymbol{a} d\boldsymbol{s}.$$

This optimization problem is solved by minimizing the Lagrangian dual problem $\mathcal{G}$, where we refer to the thesis of Ziebart [2010] for further details on the derivation of the dual problem. The maximum entropy policy is then found to be

$$\pi_t(\mathbf{a}|\mathbf{s}) = \exp\left(Q^{\text{soft}}(\mathbf{s}, \mathbf{a}) - V^{\text{soft}}(\mathbf{s})\right). \tag{2.1}$$

where $V_t^{\text{soft}}(\mathbf{s})$ is the Lagrangian multiplier of the dynamics constraint and relates to the value function of π. The optimality condition for $V_t^{\text{soft}}(\mathbf{s})$ is found by setting the partial derivative $\frac{\partial \mathcal{G}}{\partial p_t^{\pi}(\mathbf{s})}$ to zero, yielding

$$V_t^{\text{soft}}(\mathbf{s}) = \log \int_{\mathbf{a}} \exp\left(Q_t^{\text{soft}}(\mathbf{s}, \mathbf{a})\right) d\mathbf{a}, \tag{2.2}$$

which corresponds to the softmax of the softened state-action value function

$$Q_t^{\text{soft}}(\mathbf{s}, \mathbf{a}) = r_t(\mathbf{s}, \mathbf{a}) + \int_{\mathbf{s}'} p_t(\mathbf{s}'|\mathbf{s}, \mathbf{a}) V_{t+1}^{\text{soft}}(\mathbf{s}') d\mathbf{s}'. \tag{2.3}$$

The learned reward function is linear in the features, i.e.

$$r_t(\mathbf{s}, \mathbf{a}) = \theta^\top \int_{\boldsymbol{y}} p_t(\boldsymbol{y}|\mathbf{s}, \mathbf{a}) \psi(\boldsymbol{y}) d\boldsymbol{y}, \tag{2.4}$$

where the weights $\boldsymbol{\theta}_t$ are the Lagrangian multipliers of the feature matching constraint and learned by minimizing the dual function [Ziebart, 2010]

$$\mathcal{G} = \mathrm{E}_{p_1(\mathbf{s})} \left[V_1^{\text{soft}}(\mathbf{s}) \right] - \sum_t \boldsymbol{\theta}_t^\top \hat{\boldsymbol{\psi}}_t$$

using gradient based optimization, where the gradient is given by the difference between the empirical feature counts of the expert $\hat{\boldsymbol{\psi}}_t$ and the expected feature counts $\tilde{\boldsymbol{\psi}}_t$ of the policy $\pi(\boldsymbol{a}|\boldsymbol{s})$ given by (2.1), i.e.

$$\frac{\partial \mathcal{G}}{\partial \boldsymbol{\theta}_t} = \tilde{\boldsymbol{\psi}}_t - \hat{\boldsymbol{\psi}}_t. \tag{2.5}$$

MaxCausalEnt-IRL can be applied for matching expert distributions by matching their moments. For example, a Gaussian distribution over the task space $\boldsymbol{y}$ can be matched by matching its first and second moments. Hence, MaxCausalEnt-IRL can be applied for matching the expert's distribution by matching a vector $\hat{\boldsymbol{\psi}}(\boldsymbol{y}_t)$, that includes the task space positions y_i and all second-degree monomials $y_i y_j$, with $1 \leq i \leq j \leq N_{\boldsymbol{y}}$, where $N_{\boldsymbol{y}}$ denotes the number of task space variables.

However, treating first-degree monomials and second-degree monomials as independent features impairs regularization as well as optimization. For example, punishing high weights on a given first-degree monomial using ℓ_1 or ℓ_2 regularization does not take the variance of the respective feature into account and may hence introduce large regularization errors on the mean even for crucial low-variance time steps. Similarly, the optimization does not take into account that changing the expected feature count of a first-degree monomial also affects the expected feature counts of its corresponding second-degree monomials.

2.3. (I)OC by Matching Distributions

In order to address the issues of MaxCausalEnt-IRL when matching first and second order moments, we propose a novel application of the principle of maximum entropy for inverse reinforcement learning that aims at minimizing the relative entropy to the distribution

$q_t(\boldsymbol{y})$ estimated from the expert rather than matching the expert's feature counts. The corresponding constrained optimization problem is given by

$$\max_{\pi_t(\mathbf{a}|\mathbf{s})} \sum_{t=1}^{T-1} H(\pi_t(\mathbf{a}|\mathbf{s})) - \sum_{t=2}^{T} \beta_t D_{\mathrm{KL}}(p_t^\pi(\boldsymbol{y})||q_t(\boldsymbol{y})), \tag{2.6}$$

where the same constraints are used for modeling the relation between $p_t^\pi(\mathbf{s})$, $\pi(\mathbf{a}|\mathbf{s})$ and $p_t^\pi(\boldsymbol{y})$ as in MaxCausalEnt-IRL. Regularization is controlled based on the coefficients $\beta_t > 0$, where high values emphasize the objective of matching the expert's distribution, and thus yield low regularization.

The optimization problem sketched by (2.6) is solved by minimizing the Lagrangian dual problem

$$\begin{aligned}\mathcal{G} =& \mathrm{E}_{p_1(\mathbf{s})}\left[V_1^{\mathrm{soft}}(\mathbf{s})\right] \\ &+ \sum_t \beta_t \log \int_{\boldsymbol{y}} \exp\left(\log q_t(\boldsymbol{y}) - \frac{1}{\beta_t}\eta_t(\boldsymbol{y})\right) d\boldsymbol{y},\end{aligned} \tag{2.7}$$

where the policies $\pi(\mathbf{a}|\mathbf{s})$ and the softened state and state-action value functions $V^{\mathrm{soft}}(\mathbf{s})$ and $Q^{\mathrm{soft}}(\mathbf{s},\mathbf{a})$ are the same as for MaxCausalEnt-IRL, i.e., they are given by (2.1), (2.2) and (2.3). The reward functions, however, are directly given by the Lagrangian multipliers $\eta_t(\boldsymbol{y}_t)$ corresponding to the transformation constraints of the task space variable, i.e.,

$$r_t(\mathbf{s},\mathbf{a}) = \int_{\boldsymbol{y}} p_t^\pi(\boldsymbol{y}|\mathbf{s},\mathbf{a})\eta_t(\boldsymbol{y})d\boldsymbol{y}.$$

Setting the partial derivative $\frac{\partial \mathcal{G}}{\partial p_t^\pi(\boldsymbol{y})}$ to zero yields an optimality condition between $p_t^\pi(\boldsymbol{y})$ and $\eta_t(\boldsymbol{y})$, given by

$$\eta_t(\boldsymbol{y}) = \beta_t \left(\log q_t(\boldsymbol{y}) - \log p_t^\pi(\boldsymbol{y})\right) + \text{const.} \tag{2.8}$$

Note that (2.8) defines the reward function recursively since the task space distribution $p_t^\pi(\boldsymbol{y})$ depends on the policy which in turn depends on the task space reward function via (2.1), (2.2) and (2.3). Furthermore, (2.8) provides information about the structure of the reward function. For example, if $q_t(\boldsymbol{y})$ and $p_t^\pi(\boldsymbol{y})$ are normally distributed, the reward function is quadratic in $\boldsymbol{y}$.

Instead of using (2.8) for estimating the task space reward function $\eta_t(\boldsymbol{y})$ based on an estimate of $p_t^\pi(\boldsymbol{y})$, (2.8) can also be reformulated for estimating the desired distribution over task variables $\tilde{p}_t(\boldsymbol{y})$ based on the current estimate of $\eta_t(\boldsymbol{y})$, i.e.

$$\tilde{p}_t(\boldsymbol{y}) \propto \exp\left(\log q_t(\boldsymbol{y}) - \frac{1}{\beta_t}\eta_t(\boldsymbol{y})\right), \tag{2.9}$$

where a tilde is used to distinguish this estimate of the distribution over task space variables from $p_t^\pi(\boldsymbol{y})$, the distribution over task space variables that is produced by the policy $\pi(\boldsymbol{a}|\boldsymbol{s})$ which is computed according to (2.1).

The dual function (2.7) can be minimized using gradient-based optimization. For example, when assuming the reward function to be linear in a given feature vector $\boldsymbol{\psi}_t(\boldsymbol{y})$, i.e. $\eta_t(\boldsymbol{y}) = \boldsymbol{\theta}_t^\top \boldsymbol{\psi}_t(\boldsymbol{y})$, the partial derivative of the dual with respect to the weight vector $\boldsymbol{\theta}_t$ is given by

$$\frac{\partial \mathcal{G}}{\partial \boldsymbol{\theta}_t} = \mathrm{E}_{p_t^\pi(\boldsymbol{y})}\left[\boldsymbol{\psi}_t(\boldsymbol{y})\right] - \mathrm{E}_{\tilde{p}_t(\boldsymbol{y})}\left[\boldsymbol{\psi}_t(\boldsymbol{y})\right]. \tag{2.10}$$

However, while we use the gradient (2.10) for discussing the relative entropy based regularization as well as the relation between our approach and MaxCausalEnt-IRL in Section 2.3.1, we use a different procedure for optimizing the dual function (2.7) that is based on the recursive definition of the reward function (2.8) and discussed in Section 2.3.2.

2.3.1. Relative Entropy Based Regularization

The gradient of the new formulation (2.10) only differs from the gradient of MaxCausalEnt-IRL (2.5) in that the empirical feature average $\hat{\boldsymbol{\psi}}_t$ has been replaced by the expectation of $\boldsymbol{\psi}_t(\boldsymbol{y})$ under $\tilde{p}_t(\boldsymbol{y})$. As the empirical feature average corresponds to the expectations of $\boldsymbol{\psi}_t(\boldsymbol{y})$ under the empirical expert distribution $q_t(\boldsymbol{y})$, our gradient (2.10) corresponds to the gradient of MaxCausalEnt-IRL (2.5), but with the target distribution replaced by $\tilde{p}_t(\boldsymbol{y})$. However, this new target distribution, $\tilde{p}_t(\boldsymbol{y})$, is adapted during optimization as it depends on the current estimate of the task space reward function.

More specifically, as it can be seen from (2.9), it is a modification of the actual target distribution $q_t(\boldsymbol{y})$ where the log-likelihood of task space variables is increased if they are assigned high reward and decreased if they are assigned low reward. Assigning high rewards to a task space variable $\boldsymbol{y}_i$ at time step t, indicates that its log-likelihood would otherwise be too small, while assigning low rewards indicates that the log-likelihood would otherwise be too high. Hence, the modified target distribution is—according to the current estimate of the task space reward function—easier to match.

Although $\tilde{p}_t(\boldsymbol{y})$ might not be feasible in the beginning of the optimization, it will converge to the same distribution as $p_t^\pi(\boldsymbol{y})$ as the algorithm converges to the optimal reward function $\eta(\boldsymbol{y})$ and policy $\pi(\boldsymbol{a}|\boldsymbol{s})$. Note that $p_t^\pi(\boldsymbol{y})$ is always feasible as it is defined as the distribution over task space variables that is produced by the current policy.

If the target distribution $q_t(\boldsymbol{y})$ is feasible, the difference between the modified target distribution $\tilde{p}_t(\boldsymbol{y})$ and $q_t(\boldsymbol{y})$ is caused solely by regularization and converges to zero as β approaches infinity. Similar to ℓ_1 or ℓ_2 regularization in MaxCausalEnt-IRL, our regularization scheme aims at increasing the controller entropy at the cost of not matching

the expert demonstrations exactly. However, instead of measuring the distance to the expert demonstrations based on the absolute or squared distances between the empirical and the expected feature count, our approach employs the relative entropy between the resulting distribution and the empirical expert's distribution over task space positions.

2.3.2. Alternative Descent Direction

In contrast to MaxCausalEnt-IRL, the relative entropy formulation does not only provide the gradient with respect to the reward function but additionally enables us to estimate the optimal reward function based on (2.8). It is therefore appealing to exploit this additional information to achieve faster optimization. Starting from an estimate $\boldsymbol{\theta}_{\eta,t}^{(i)}$ of the parameters of the reward function, the resulting distribution over task space positions can be computed based on (2.1), (2.2) and (2.3), which can then be used for computing a new estimate $\boldsymbol{\theta}_{\eta,t}^{(i+1)}$ based on (2.8). However, such greedy jumps based on estimates $p_t^{\pi^{(i)}}(\boldsymbol{y})$ of the optimal, regularized distribution over task space positions do not guarantee convergence due to the recursive relation of the reward function and the task space distribution $p_t^{\pi}(\boldsymbol{y})$. Instead, we propose to interpolate the current estimate of the weights $\boldsymbol{\theta}_{\eta,t}^{(i)}$ with the estimate of the optimal weights $\tilde{\boldsymbol{\theta}}_{\eta,t}^{(i)}$ according to (2.8), i.e.

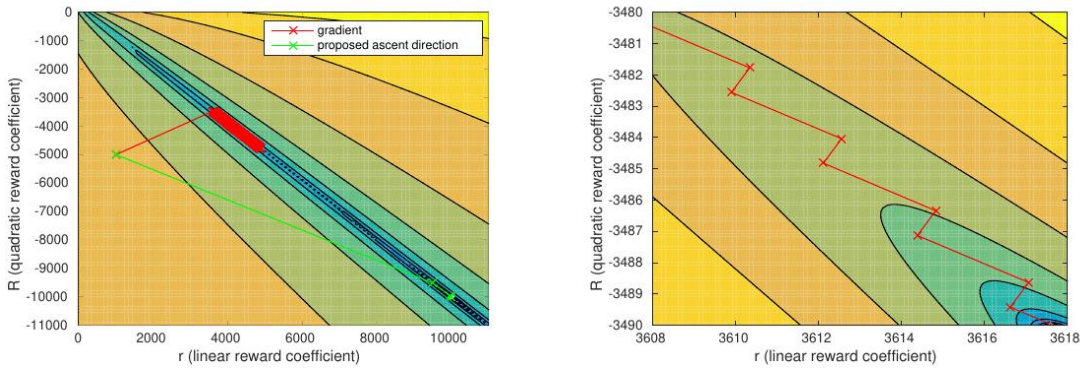
$$\begin{aligned}\boldsymbol{\theta}_{\eta,t}^{(i+1)} &= (1-\alpha)\boldsymbol{\theta}_{\eta,t}^{(i)} + \alpha\tilde{\boldsymbol{\theta}}_{\eta,t}^{(i)} \\ &= \boldsymbol{\theta}_{\eta,t}^{(i)} - \alpha\left(\boldsymbol{\theta}_{\eta,t}^{(i)} - \tilde{\boldsymbol{\theta}}_{\eta,t}^{(i)}\right) = \boldsymbol{\theta}_{\eta,t}^{(i)} - \alpha\boldsymbol{\delta}_{\boldsymbol{\theta}_{\eta,t}},\end{aligned}$$

with step size α.

The update direction $\boldsymbol{\delta}_{\boldsymbol{\theta}_{\eta,t}}$ is an ascent direction of the dual function (2.7) and the proposed update scheme, thus, converges for reasonable step sizes. A proof is given in the supplementary material, that also covers the special case of MaxCausalEnt-IRL, where $\tilde{p}_t(\boldsymbol{y}) = q_t(\boldsymbol{y})$. Hence, our update direction can be also applied for other methods that are based on MaxCausalEnt-IRL, assuming that the parameters of the expert distribution $\boldsymbol{\theta}_{q_t}$ can be estimated.

2.3.3. LQR Solutions

Computing the policy for a given reward function, as well as the resulting distribution over task space positions is in general hard and a major challenge when applying IRL to real-world applications. Linear-quadratic regulators (LQRs) are an important exception that allow to compute both, the policies and the resulting state distributions by means of



(a) (b)

Figure 2.1.: (a) Comparison of both update directions on a toy task with only one task variable and two features (linear and quadratic term). While gradient descent needs many updates as it neglects the dependencies of the linear and the quadratic parameters, our new update direction achieves fast convergence within few updates. (b) Zoomed updates of the gradient descent. ©2016 IEEE.

dynamic programming. An LQR is a control problem with linear Gaussian state dynamics

$$p_t(\mathbf{s}'|\mathbf{s},\mathbf{a}) = \mathcal{N}\left(\mathbf{s}'|\mathbf{A}_t\mathbf{s} + \mathbf{B}_t\mathbf{a} + \mathbf{c}_t, \mathbf{\Sigma}_t\right)$$

and concave quadratic state-action reward functions

$$r_t(\mathbf{s},\mathbf{a}) = \begin{pmatrix}\mathbf{s}\\ \mathbf{a}\end{pmatrix}^\top \mathbf{R}_t \begin{pmatrix}\mathbf{s}\\ \mathbf{a}\end{pmatrix} + \begin{pmatrix}\mathbf{s}\\ \mathbf{a}\end{pmatrix}^\top \mathbf{r}_t + r_t.$$

The resulting softened state-action value functions

$$Q_t^{\text{soft}}(\mathbf{s},\mathbf{a}) = \begin{pmatrix}\mathbf{s}\\ \mathbf{a}\end{pmatrix}^\top \mathbf{Q}_t \begin{pmatrix}\mathbf{s}\\ \mathbf{a}\end{pmatrix} + \begin{pmatrix}\mathbf{s}\\ \mathbf{a}\end{pmatrix}^\top \mathbf{q}_t + q_t$$

as well as the softened state value functions

$$V_t^{\text{soft}}(\mathbf{s}) = \mathbf{s}^\top \mathbf{V}_t \mathbf{s} + \mathbf{s}^\top \mathbf{v}_t + v_t$$

are then also concave quadratic functions. Furthermore, the policies given by (2.1) are given by stochastic linear controllers

$$\pi_t(\mathbf{a}|\mathbf{s}) = \mathcal{N}\left(\mathbf{a}|\mathbf{K}_t\mathbf{s} + \mathbf{k}_t, \mathbf{\Sigma}_{\pi,t}\right). \tag{2.11}$$

The parameters can be computed using dynamic programming starting with $V_T(\boldsymbol{s}) = r_T(\boldsymbol{s}, \mathbf{0})$. The resulting softened value function differs from the actual value function of the controller given in (2.1), which would be computed as $V_t(\mathbf{s}) = \int_{\mathbf{a}} \pi_t(\mathbf{a}|\mathbf{s}) Q_t(\mathbf{s}, \mathbf{a}) d\boldsymbol{a}$, only in the offset v_t which is increased by the amount of entropy of the controller, i.e.

$$V_t^{\text{soft}}(\mathbf{s}) = V_t(\mathbf{s}) + \sum_{i=t}^{T-1} H(\pi_i).$$

Given a Gaussian initial state distribution $p_1(\mathbf{s})$ and linear transformation of the task space

$$p_t(\boldsymbol{y}|\mathbf{s}, \mathbf{a}) = \mathcal{N}\left(\boldsymbol{y}|\mathbf{F}_t \begin{bmatrix}\mathbf{s} & \mathbf{a}\end{bmatrix}^\top + \mathbf{f}_t, \boldsymbol{\Sigma}_{F,t}\right),$$

the controller given by (2.11) produces Gaussian state and task space distributions. We will use linearizations for obtaining the task space, e.g., if the task space corresponds to the end-effector position, $\boldsymbol{F}$ is given by the Jacobian matrix. Further assuming Gaussian expert distributions, the reward functions computed according to (2.8) are quadratic in task space positions and the related rewards $r_t(\mathbf{s}, \mathbf{a}) = \int_{\boldsymbol{y}} p_t(\boldsymbol{y}|\mathbf{s}, \mathbf{a}) \eta_t(\boldsymbol{y}) d\boldsymbol{y}$ are again quadratic in states and actions.

Comparison of Descent Directions

The difference between the gradient and the ascent direction $\delta_{\theta_\eta, t}$ can be better understood by comparing them in the LQR setting. The partial derivatives with respect to the weights of the linear and quadratic terms are then given by

$$\begin{aligned}
\frac{\partial \mathcal{G}}{\partial \mathbf{r}_{\boldsymbol{y},t}} &= \boldsymbol{\mu}_{p^\pi,t} - \boldsymbol{\mu}_{\tilde{p}}, \\
\frac{\partial \mathcal{G}}{\partial \mathbf{R}_{\boldsymbol{y},t}} &= \boldsymbol{\mu}_{p^\pi,t}\boldsymbol{\mu}_{p^\pi,t}^\top + \boldsymbol{\Sigma}_{p^\pi,t} - \boldsymbol{\mu}_{\tilde{p},t}\boldsymbol{\mu}_{\tilde{p},t}^\top - \boldsymbol{\Sigma}_{\tilde{p},t},
\end{aligned}$$

whereas the proposed ascent directions update along

$$\begin{aligned}
\delta_{\mathbf{r}_{\boldsymbol{y},t}} &= \boldsymbol{\Sigma}_{p^\pi,t}^{-1}\boldsymbol{\mu}_{p^\pi,t} - \boldsymbol{\Sigma}_{\tilde{p},t}^{-1}\boldsymbol{\mu}_{\tilde{p},t}, \\
\delta_{\mathbf{R}_{\boldsymbol{y},t}} &= -\frac{1}{2}\boldsymbol{\Sigma}_{p^\pi,t}^{-1} + \frac{1}{2}\boldsymbol{\Sigma}_{\tilde{p},t}^{-1}.
\end{aligned} \tag{2.12}$$

The gradient does not take into account the correlations between the weights of the linear terms and the weights of the quadratic terms for a given time step t. Hence, the gradient would not change the linear term of the reward function, if both distributions

matched in mean but differed in their covariances. The drawbacks of neglecting these interdependencies are depicted in Fig. 2.1. For a better illustration, a simple IRL problem was chosen with only two time steps, a single task variable and true knowledge of the expert distribution. Fig. 2.1a shows three iterations of inverse reinforcement learning when following the proposed search direction and a thousand iterations when following the gradient. In both cases, the optimal step size was found using a line search. Following the gradient quickly leads to a good goal position (relating to the ratio of quadratic and linear coefficient), however, it converges very slowly to the correct quadratic coefficient. The resulting distributions are quickly matching the expert distribution in mean but fail in matching the variance accurately. Fig. 2.1b shows a zoomed in view on the gradient updates. By changing the linear coefficients too slowly when the means are closely matched, even small changes to the quadratic terms lead to wrong goal positions and thus increase the value of the dual function. In contrast, the true reward function could be recovered by following the proposed ascent direction after only three iterations.

Regularized Gaussian Distributions

The effect of regularization can also be better understood by an examination in the LQR setting. The resulting covariance matrices and mean vectors of our approach are then given by

$$\begin{aligned}\boldsymbol{\Sigma}_{\tilde{p},t} &= \left(\boldsymbol{\Sigma}_{q,t}^{-1} + 2\beta_t^{-1}\mathbf{R}_{\boldsymbol{y},t}\right)^{-1}, \\ \boldsymbol{\mu}_{\tilde{p},t} &= \boldsymbol{\Sigma}_{\tilde{p},t}\left(\boldsymbol{\Sigma}_{q,t}^{-1}\boldsymbol{\mu}_{q,t} - \beta_t^{-1}\boldsymbol{r}_{\boldsymbol{y},t}\right).\end{aligned}$$

Hence, the precision matrices of the expert distribution are interpolated with the reward matrices $\mathbf{R}_{\boldsymbol{y},t}$. When computing $\boldsymbol{\mu}_{\tilde{p},t}(\boldsymbol{y})$, the mean of the expert distribution is rescaled based on the precision matrix before interpolation with the linear reward coefficient, thereby putting more weight on the expert's mean for low-variance time steps.

2.3.4. Linearized Dynamics

For many real-world applications, the system dynamics are non-linear and the LQR derivations can not be applied straightforwardly. In order to make the computation still feasible, linearizations of the dynamics are commonly applied. Li and Todorov [2004] estimate a locally optimal controller for a given reward function by iteratively using linear approximations of the dynamics and quadratic approximations of the reward function based on the state-action trajectory of the last iteration. However, since these approximations are only valid in the proximity of the last trajectory distribution, optimization might

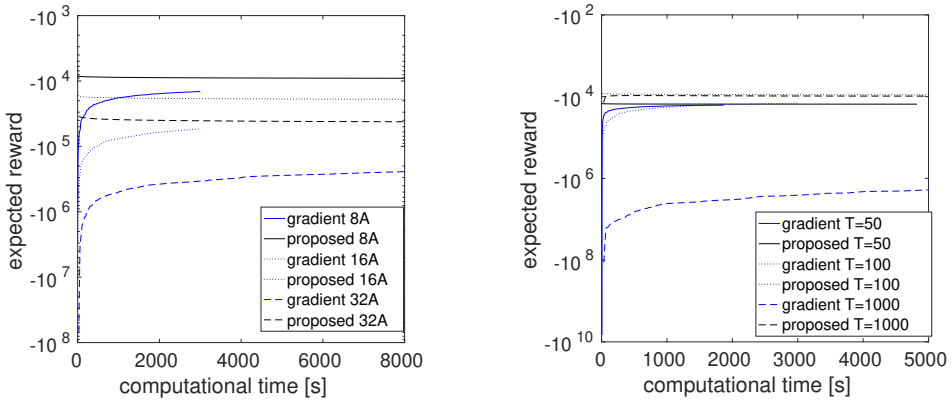


Figure 2.2.: (left) Expected reward for different number of actions and $T{=}100$. (right) Expected reward for different time horizons and eight actions. When using the proposed search direction, the algorithm converges significantly faster. ©2016 IEEE.

become unstable if the trajectory distributions change too much. We follow Levine and Abbeel [2014] by adding a constraint to our optimization problem that bounds the relative entropy between the learned controller and the last controller, $\pi_{\text{last}}(\boldsymbol{a}|\boldsymbol{s})$, that was used to obtain the linearization, i.e.

$$\forall_t : \quad D_{\text{KL}}\left(\pi_t(\mathbf{a}|\mathbf{s})||\pi_t^{\text{last}}(\mathbf{a}|\mathbf{s})\right) \leq \epsilon_t,$$

where ϵ_t is the desired bound. This constraint induces additional reward based on the likelihood of the action under the last controller, namely

$$r_t(\mathbf{s},\mathbf{a}) = \int_{\boldsymbol{y}} p_t(\boldsymbol{y}|\mathbf{s},\mathbf{a})\eta_t(\boldsymbol{y})d\boldsymbol{y} + \alpha_t \log \pi_t^{\text{last}}(\mathbf{a}|\mathbf{s}),$$

where α_t are the corresponding Lagrangian multipliers. Furthermore, the dual function is augmented by $\sum_t \alpha_t\epsilon_t$. The weights α_t can be learned based on the partial derivative

$$\frac{\partial \mathcal{G}}{\partial \alpha_t} = D_{\text{KL}}\left(\pi_t(\mathbf{a}|\mathbf{s})||\pi_t^{\text{last}}(\mathbf{a}|\mathbf{s})\right) - \epsilon_t.$$

2.4. Experiments

We start our evaluation by comparing our method with MaxCausalEnt-IRL in terms of convergence speed and quality of regularization on a simple linear system. In the second part of our experiments we compare our method with related work [Yin et al., 2014] where we want to learn a reward function for robotic handwriting and with the approach by Englert et al. [2013] where we want to learn a pendulum swing-up by matching a distribution over joint positions and velocities. In our main experiment, we demonstrate the applicability of our differential dynamic programming based (I)OC method on a simulated quad-link. For IOC, we learn a time-dependent reward function of a near-optimal swing up and for optimal control we learn to produce a dynamic peg-in-hole movement based on a specified target distribution over end-effector positions and orientations.

2.4.1. Linear System

We chose a stochastic linear system with one action and two states per dimension, such that the actions corresponds to accelerations and the states to corresponding velocities and positions. The single dimensions are not coupled. The underlying reward function is a quadratic, time-dependent function that assigns high rewards to four via points at time steps $T/4$, $T/3$, $T/2$ and T and very low rewards for the remaining time steps. Additionally, we use time-independent, uncorrelated quadratic action costs. The expert policy of this LQR is computed based on optimal control and the resulting distributions over positions are to be matched.

Speed of Convergence

The convergence speed is compared for different number of dimensions as well as for different time horizons T. When evaluating the gradient based MaxCausalEnt-IRL, we use L-BFGS [Boyd and Vandenberghe, 2004] for optimization. For the proposed search direction such gradient based optimizers are not applicable. Therefore, we chose a simple step size adaption scheme that increases the step size by 1.2 if the dual function decreased after the last step and decreases the step size by 0.5 if the dual function increased. Furthermore, steps that led to an increase of the dual function are undone. Fig. 2.2 shows the expected reward of the learned policy under the true reward function for different horizons and action dimensions. Albeit the simplicity of the system, MaxCausalEnt-IRL often failed to match the target distribution sufficiently well even after several hours of optimization.

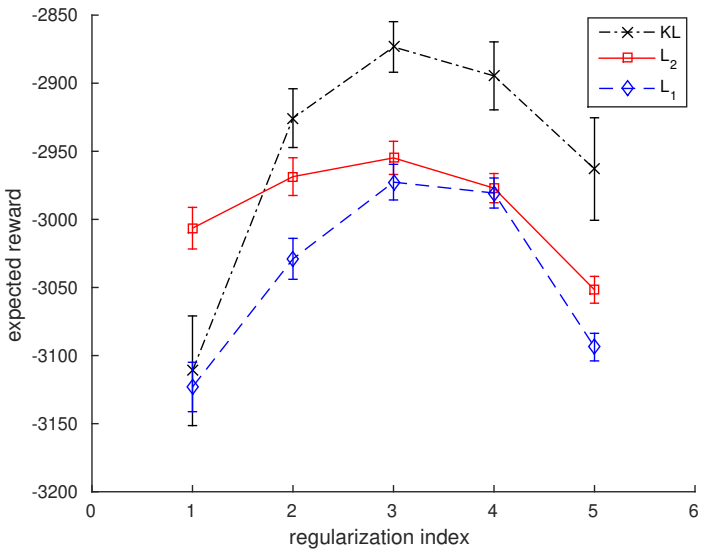


Figure 2.3.: Mean of the expected reward when presented with three demonstrations. Five different coefficients have been tested for each type of regularization. KL-based regularization can achieve more expected reward than regularization based on the ℓ_1 or ℓ_2 norm. ©2016 IEEE.

Regularization

Due to the difficulty of matching higher order moments based on the gradient, we were restricted when choosing a system for comparing the KL based regularization with regularization based on L_1 or L_2. Therefore, we had to opt for a one-dimensional system with $T{=}50$. The distribution over positions was estimated based on three sample trajectories of the optimal controller. Fig. 2.3 shows the estimated mean of the expected reward with 2σ-confidence for five different coefficients per regularization type. Mean and standard error have been computed based on 96 trials. The results of our experiment indicate that, for reasonably chosen coefficients, regularization based on the relative entropy performs significantly better than regularization based on the ℓ_1 or ℓ_2 norm.

2.4.2. Robotic Handwriting

The problem of inferring a reward function for matching a target distribution was also tackled by Yin et al. [2014] based on a variant [Ziebart et al., 2008] of MaxCausalEnt-IRL

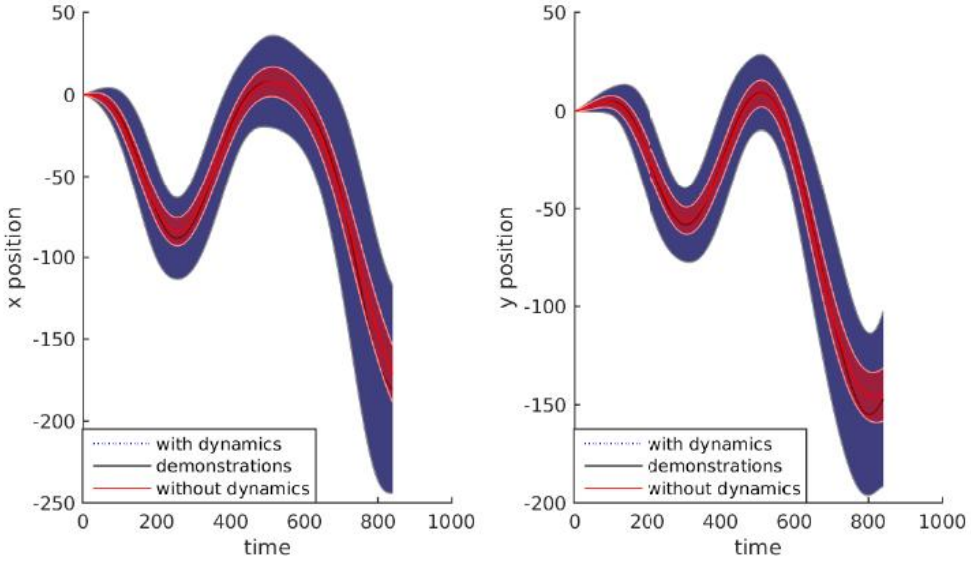


Figure 2.4.: Only when taking the system dynamics into account while learning the reward function, the resulting distribution (blue) matches the target distribution (black) accurately. ©2016 IEEE.

that does not take causality into account. When learning the reward function for robotic handwriting, they neglect the effect of the dynamics on the resulting distribution over pen tip trajectories. However, when solving the optimal control problem for such reward functions the resulting trajectory distribution of the optimal controller would no longer match the expert distribution. Instead, we apply our method to match a distribution over pen tip trajectories [Lichman, 2013; Williams et al., 2006] while taking into account the system dynamics. We demonstrate the difference between these approaches based on the linear model discussed in the last section, yielding two actions for accelerations in x and y direction and four states for the corresponding positions and velocities. The demonstrated trajectories have been aligned by curve-fitting and sub-sampling to a fixed horizon $T{=}840$. The resulting distributions over task space positions after optimizing the different reward functions are shown in Fig 2.4. By taking the system dynamics into account, we are still able to produce the target distribution. In this case, neglecting the system dynamics led to a reward function that assigns too much reward for staying close to the mean trajectory and produces a stiff controller.

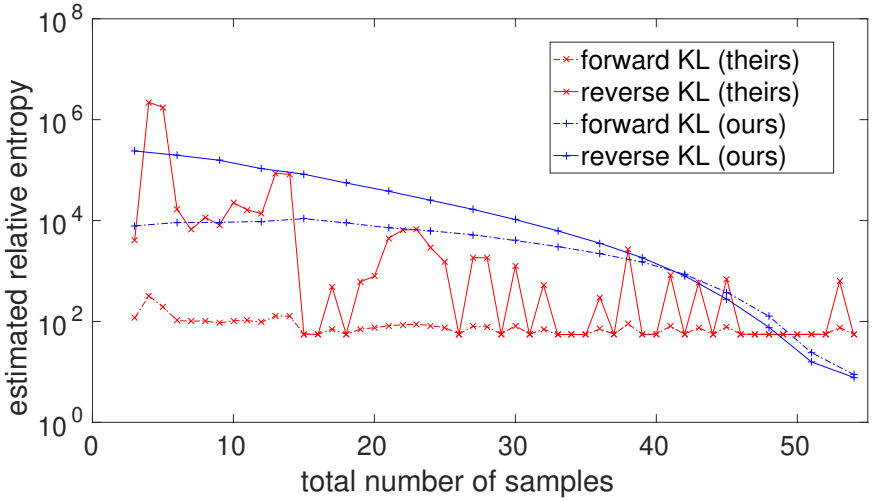


Figure 2.5.: Forward KL and reverse KL of our approach (blue) compared with the approach by Englert et al. [2013] (red). The relative entropies have been estimated based on 1000 samples on the actual system. While our approach is less sample efficient, it converges to a better solution. ©2016 IEEE.

2.4.3. Pendulum Swing-Up

We compared our work to the approach by Englert et al. [2013] on a simulated pendulum with a length of 0.6 meter that weighs 500 gram. A target distribution over joint positions and joint velocities for a swing-up movement was estimated from samples of their controller and presented to both algorithms. The movement took 2.5 seconds and was discretized into 25 time steps, yielding intervals of 100 ms.

For our approach, we iteratively used linear approximations of the dynamics as discussed in Section 2.3.4. We learned the linear approximations using ridge regression based on five sampled trajectories. However, we only generated three samples for each iteration and reused two samples of the previous rollout. We did not address sample efficiency in this work, but want to point out that Levine and Abbeel [2014] reuses samples from previous iterations and different time steps by learning a Gaussian mixture model as prior. Such modifications could be straightforwardly applied to our method as well.

The approach presented by Englert et al. [2013] is very sample efficient by learning a Gaussian Process for approximating the system dynamics and, thus, uses only one sample per rollout.

Fig. 2.5 shows the relative entropy for both approaches plotted over the total number of samples. Since Englert et al. [2013] minimize the forward KL, $D_{\mathrm{KL}}(q_t(\boldsymbol{y})||p_t^{\pi}(\boldsymbol{y}))$, whereas our approach minimizes the reverse KL, $D_{\mathrm{KL}}(p_t^{\pi}(\boldsymbol{y})||q_t(\boldsymbol{y}))$, we show the results of both objectives. Although the approach by Englert et al. [2013] provides good results already after few executions on the actual system, our approach converges to a better solution.

2.4.4. Frictionless Quad-Link

While we were restricted to low-dimensional systems and small number of time steps for our comparisons to related work, we also tested the applicability of our approach on a more challenging simulation of a frictionless, planar kinematic chain of four links. Each link has a length of 1 meter and weighs 1 kilogram. The simulation takes into account gravity as well as coriolis and centrifugal forces. Time is discretized into intervals of 10 milliseconds.

Peg in Hole

For the optimal control task, the target distribution is specified directly in order to define via points in task space. We use three task space variables for specifying the end-effector position in x and y position as well the end-effector angle relative to the y axis. We use independent coefficients $\beta_{f,t}$ to define the importance of meeting the objective for task space f at time step t. Setting the corresponding coefficient to zero disables the objective completely. We test our approach for inserting the last link horizontally into a small hole in a wall. We choose $T{=}200$ and only specify target distributions for the last 50 time steps. For those time steps, the desired mean end-effector positions along the y axis are set to 1 and the desired mean end-effector angles are set to $\frac{\pi}{2}$ inducing the desired horizontal alignment of the last link. The inserting motion is induced by setting the target mean x-coordinate of the end-effector. This target distribution is only set for time steps 175 and 200 with desired mean positions of 2 and 2.5. For all target distributions, we set the variances to $1\mathrm{e}{-4}$. The resulting movement is shown in Fig. 2.6 (left). Fig. 2.7 compares the achieved distributions and the target distributions in the vicinity of the specified time steps. Our approach achieves the desired distributions over task variables with high accuracy.

Swing Up

We performed our nonlinear inverse optimal control method for inferring the reward function for a swing-up movement based on locally optimal demonstrations. The demon-

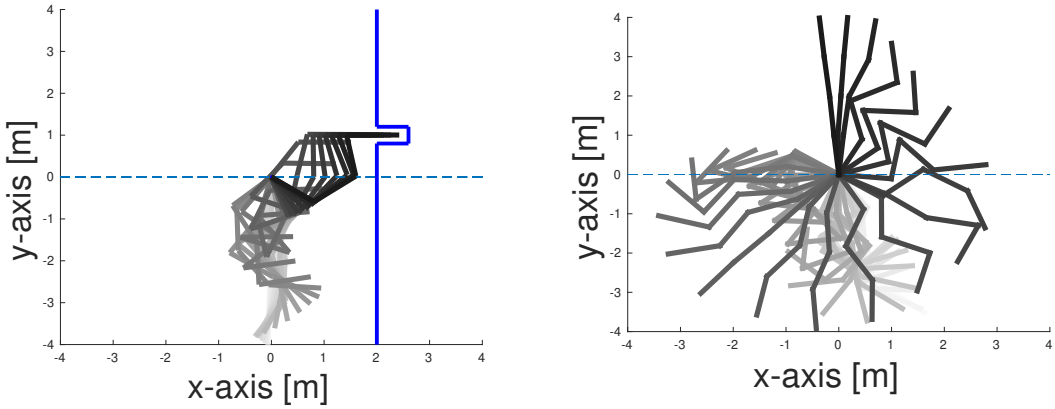


Figure 2.6.: (left) The peg-in-hole movement is performed with the specified accuracy. (right) A time-dependent reward function as well as the corresponding controller was learned from demonstrated swing-ups. ©2016 IEEE.

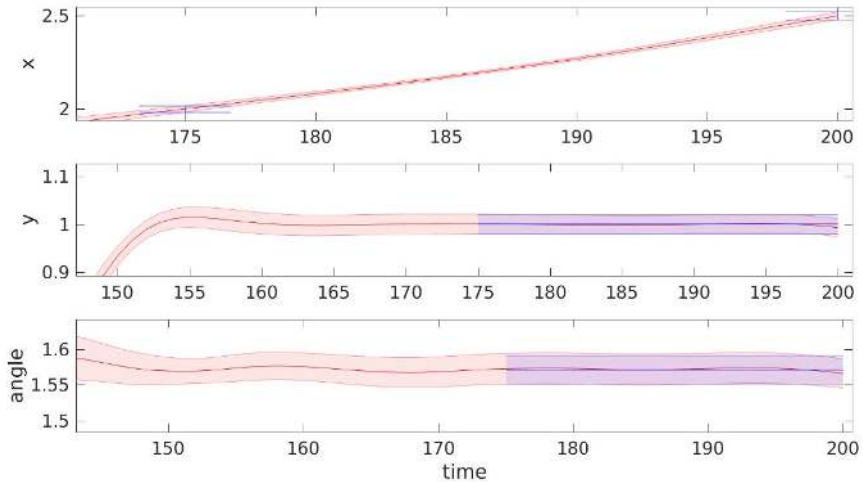


Figure 2.7.: The achieved distribution (red) was estimated based on 1000 samples and compared to the target distribution (blue) in the vicinity of specified time steps. Our approach accurately matches mean and variance for all three task variables. ©2016 IEEE.

strations were produced by a linear controller that was learned using MOTO [Akrour et al., 2016]. Furthermore, only the joint positions were presented to our algorithm. Optimizing the learned reward function produced the desired behavior as shown in Fig. 2.6 (right).

2.5. Conclusion

We presented a method that unifies optimal control and inverse optimal control in one framework by learning the controller and the corresponding reward function for matching a given distribution over trajectories. For optimal control, directly specifying the desired accuracy for given goal positions is arguably less cumbersome than specifying a reward function. For inverse optimal control, our approach is several orders of magnitudes more efficient in matching target distributions than MaxCausalEnt-IRL and allows for better regularization based on the relative entropy. Furthermore, based on incremental linearizations of the dynamics, we can perform non-linear inverse optimal control even when the states and actions are not observed directly.

3. Trust-Region I-Projections for Variational Inference

Inference from a complex distribution $p(\mathbf{x})$ is a huge problem in machine learning that is needed in many applications. Typically, we can evaluate the distribution except for the normalization factor Z, that is, we can only evaluate the unnormalized distribution $\tilde{p}(\mathbf{x})$, where

$$p(\mathbf{x}) = \tilde{p}(\mathbf{x})/Z,$$

with $Z = \int_{\mathbf{x}} \tilde{p}(\mathbf{x}) d\mathbf{x}$. For example, in Bayesian inference $\tilde{p}(\mathbf{x})$ would correspond to the product of prior and likelihood. As exact inference is often intractable, we have to rely on approximate inference.

Markov chain Monte Carlo (MCMC) is arguably the most commonly applied technique for approximate inference. Samples are drawn from the desired distribution by building Markov chains for which the equilibrium distribution matches the desired distribution $p(\mathbf{x})$. Monte Carlo estimates based on these samples are then used for inference. However, MCMC can be very inefficient, because it is difficult to make full use of function evaluations of $\tilde{p}(\mathbf{x})$ without violating the Markov assumption.

Instead, we propose a method based on variational inference, which is another commonly applied technique for approximate inference. In variational inference, the desired distribution $p(\mathbf{x})$ is approximated by a tractable distribution $q(\mathbf{x}; \boldsymbol{\theta})$ which can be used for exact inference instead of $p(\mathbf{x})$, or as a more direct alternative to MCMC for drawing samples for (possibly importance weighted) Monte Carlo estimates. The approximation $q(\mathbf{x}; \boldsymbol{\theta})$ is typically found by minimizing the reverse Kullback-Leibler (KL) divergence

$$\mathrm{KL}\left(q(\mathbf{x}; \boldsymbol{\theta}) || p(\mathbf{x})\right) = \int_{\mathbf{x}} q(\mathbf{x}; \boldsymbol{\theta}) \log\left(\frac{q(\mathbf{x}; \boldsymbol{\theta})}{p(\mathbf{x})}\right) d\mathbf{x}, \tag{3.1}$$

with respect to the parameters $\boldsymbol{\theta}$ of the approximation.

By framing inference as an optimization problem, variational inference can make better use of previous function evaluations of $\tilde{p}(\mathbf{x})$ than MCMC and is therefore computationally more efficient. However, in order to perform the KL minimization efficiently, $q(\mathbf{x}; \boldsymbol{\theta})$ is

often restricted to belonging to a simple family of models or is assumed to have non-correlating degrees of freedom [Blei et al., 2017; Peterson and Hartman, 1989], which is known as the mean field approximation. Unfortunately, such restrictions can introduce significant approximation error especially for multimodal target distributions. Comparing MCMC with variational inference, we can conclude that we should use MCMC when we require accuracy (due to its asymptotic guarantee of exactness), whereas we should prefer variational inference when we need computationally efficient solutions [Blei et al., 2017].

Hence, there is a huge interest in finding computationally efficient solutions with high sample quality. Our work aims at learning highly accurate approximations for computationally efficient variational inference methods. We use Gaussian mixture models (GMMs) as model family, because they can be sampled efficiently and are capable of representing any target distribution arbitrarily well if the number of components is sufficiently large. As the required number of components is typically not known a priori, we dynamically add or delete components during optimization.

A major challenge of learning highly accurate approximations of multimodal distributions is to achieve stable and efficient optimization of an intractable objective function. We derive a lower bound on the KL divergence (Equation 3.1) based on a decomposition that is related to the one used by the expectation-maximization procedure for fitting GMMs for density estimation. We can thus optimize the original objective by iteratively maximizing and tightening this lower bound. Maximizing the lower bound decomposes into independent sub-problems for each Gaussian component that are solved, analogously to the policy search method MORE [Abdolmaleki et al., 2015], based on local quadratic approximations. Due to its strong ties to policy search, we call our method Variational Inference by Policy Search (VIPS).

Another major challenge when striving for high quality approximations is to discover the relevant modes of the target distribution. The areas of high density are initially unknown and have to be discovered during learning based on function evaluations of $\tilde{p}(\mathbf{x})$. The unnormalized target distribution, however, is typically evaluated at locations that have been sampled from the current approximation $q(\mathbf{x};\boldsymbol{\theta})$, because these samples are well suited for the optimization, for example for approximating the objective (Equation 3.1) or its gradient. The current approximation thus serves as search distribution and needs to be adapted carefully in order to avoid erroneously discarding important regions. The conflicting goals of moving the approximation towards high density regions and evaluating $\tilde{p}(\mathbf{x})$ at unexplored regions can be seen as an instance of the exploration-exploitation dilemma that is well-known in reinforcement learning [Sutton and Barto, 1998] but currently hardly addressed by the variational inference community.

Our proposed method leverages insights from policy search [Deisenroth et al., 2013], a sub-field of reinforcement learning, by bounding the KL divergence between the updated

approximation and the current approximation at each learning step. This information-geometric trust region serves the dual-purpose of staying in the validity of the local quadratic models as well as ensuring careful exploration of the search space. By finding the best approximation within such information-geometric trust region, we limit the change in search space while making sufficient progress during each iteration. However, information-geometric trust regions only address local exploration in the vicinity of the components of the current approximation and may in practice still discard regions of the search space prematurely. In order to discover modes that are not covered by the current approximation, we dynamically create new mixture components at interesting regions. Namely, we add additional components at regions where the current approximation has little probability mass although we suspect a mode of the target distribution based on previous function evaluations.

We evaluate VIPS on several domains and compare it to state-of-the-art methods for variational inference and Markov-chain Monte Carlo. We demonstrate that we can learn high quality approximations of several challenging multimodal target distributions that are significantly better than those learned by competing methods for variational inference. Compared to sampling methods, we show that we can achieve similar sample quality while using several orders of magnitude less function evaluations. Samples from the learned approximation can therefore often be used directly for approximate inference without needing importance weighting. Still, knowing the actual generative model can be a further advantage compared to model-free samplers.

This work extends previously published work about VIPS [Arenz et al., 2018] by using more efficient sample reuse, by showcasing and fixing a failure case of the previous initialization of covariance matrices, and by several other improvements such as adaptation of regularization coefficients and KL bounds. These modifications lead to a further reduction of sample complexity by approximately one order of magnitude. We will refer to the improved version as VIPS++. We evaluate VIPS++ on additional, more challenging domains, namely Bayesian Gaussian process regression and Bayesian parameter estimation of ordinary differential equations applied to the Goodwin oscillator [Goodwin, 1965] as well as more challenging variations of the previously published *planar robot* and *Gaussian mixture model* experiments [Arenz et al., 2018]. Furthermore, we now also compare to normalizing flows [Kingma et al., 2016] and black-box variational inference [Ranganath et al., 2014].

3.1. Preliminaries

In this section we formalize the optimization problem and show its connection to policy search. We further discuss the policy search method MORE [Abdolmaleki et al., 2015] and show that a slight variation of it can be used for learning Gaussian variational approximations (GVAs) for variational inference. This variant of MORE is used by VIPS for independent component updates, which will be discussed in Section 3.2.

3.1.1. Problem formulation

Variational inference is typically framed as an information projection (I-projection) problem, that is, we want to find the parameters $\boldsymbol{\theta}$ of a model $q(\mathbf{x};\boldsymbol{\theta})$ that minimize the KL divergence between $q(\mathbf{x};\boldsymbol{\theta})$ and the target distribution $p(\mathbf{x})$,

$$\begin{aligned}\mathrm{KL}\left(q(\mathbf{x};\boldsymbol{\theta})||p(\mathbf{x})\right) &= \int_{\mathbf{x}} q(\mathbf{x};\boldsymbol{\theta}) \log\left(\frac{q(\mathbf{x};\boldsymbol{\theta})}{p(\mathbf{x})}\right) d\mathbf{x} \\ &= \int_{\mathbf{x}} q(\mathbf{x};\boldsymbol{\theta}) \log\left(\frac{q(\mathbf{x};\boldsymbol{\theta})}{\tilde{p}(\mathbf{x})}\right) d\mathbf{x} + \log Z \\ &= -L(\boldsymbol{\theta}) + \log Z.\end{aligned}$$

The normalizer Z does not affect the optimal solution for the parameters $\boldsymbol{\theta}$ as it enters the objective function as constant offset and can thus be ignored. Hence, the KL divergence can be minimized by maximizing $L(\boldsymbol{\theta})$, which is a lower bound on the log normalizer due to the non-negativity of the KL divergence. In Bayesian inference, the target distribution $p(\mathbf{x})$ corresponds to the posterior, the unnormalized distribution $\tilde{p}(\mathbf{x})$ corresponds to the product of prior and likelihood, and the normalizer corresponds to the evidence. Minimizing the KL divergence thus corresponds to maximizing a lower bound on the (log) evidence, $L(\boldsymbol{\theta})$, which is therefore commonly referred to as the evidence lower bound objective (ELBO, e.g., Blei et al. 2017).

Although VIPS is not restricted to the Bayesian setting but aims to approximate intractable distributions in general, we also frame our objective as ELBO maximization because this formulation highlights an interesting connection to policy search. We treat information projection as the problem of finding a search distribution, $q(\mathbf{x};\boldsymbol{\theta})$, over a parameter space $\mathbf{x}$, that maximizes an expected return $R(\mathbf{x}) = \log\tilde{p}(\mathbf{x})$ with an additional objective of maximizing its entropy $\mathrm{H}\big(q(\mathbf{x};\boldsymbol{\theta})\big) = -\int_{\mathbf{x}} q(\mathbf{x};\boldsymbol{\theta}) \log q(\mathbf{x};\boldsymbol{\theta}) d\mathbf{x}$, that is, we aim to solve

$$\arg\max_{\boldsymbol{\theta}} \left[L(\boldsymbol{\theta}) = \int_{\mathbf{x}} q(\mathbf{x};\boldsymbol{\theta}) \big(\log\tilde{p}(\mathbf{x}) - \log q(\mathbf{x};\boldsymbol{\theta}) \big) d\mathbf{x} = \int_{\mathbf{x}} q(\mathbf{x};\boldsymbol{\theta}) R(\mathbf{x}) d\mathbf{x} + \mathrm{H}(q(\mathbf{x};\boldsymbol{\theta})) \right].$$

Entropy objectives are also commonly used in policy search for better exploration [Abdolmaleki et al., 2015; Neu et al., 2017]. Policy search methods that support such entropy objectives can thus be applied straightforwardly for variational inference. However, many policy search methods are restricted to unimodal distributions (typically Gaussians) and are therefore not suited for learning accurate approximations of multimodal target distributions. We will now review one such policy search method, MORE [Abdolmaleki et al., 2015], and show that it can be adapted straightforwardly for learning Gaussian variational approximations.

3.1.2. Model-Based Relative Entropy Stochastic Search

Policy search methods start with an initial search distribution $q^{(0)}(\mathbf{x})$ and iteratively update it in order to increase its expected reward.[1] Areas of high reward are initially not known and have to be discovered based on evaluations of the reward function $R(\mathbf{x})$ during learning. Policy search methods, therefore, typically evaluate the reward function on samples from the current search distribution in order to identify regions of high reward, and update the search distribution to increase the likelihood of the search distribution in these areas.

In order to avoid premature convergence to poor local optima, it is crucial to start with an initial search distribution $q^{(0)}$ with sufficiently high entropy and to ensure that high reward regions are not erroneously discarded due to too greedy updates. This trade-off between further exploring the search space and focusing on high reward areas is an instance of the exploration-exploitation dilemma that several policy search methods address using information-geometric trust regions [Abdolmaleki et al., 2015, 2017; Levine and Koltun, 2013; Peters et al., 2010; Schulman et al., 2015]. These methods compute each policy update by solving a constrained optimization problem that bounds the KL divergence between the next policy and the current policy.

MORE [Abdolmaleki et al., 2015] additionally limits the entropy loss between subsequent iterations by computing the update as

$$\begin{aligned} q^{(i+1)} = \quad & \arg\max_{q} \int_{\mathbf{x}} q(\mathbf{x}) R(\mathbf{x}) d\mathbf{x}, \\ & \text{s.t.} \quad \mathrm{KL}\Big(q(\mathbf{x}) || q^{(i)}(\mathbf{x})\Big) \leq \epsilon, \qquad \mathrm{H}\Big(q(\mathbf{x})\Big) \geq \beta^{(i)}, \qquad \int_{\mathbf{x}} q(\mathbf{x}) d\mathbf{x} = 1, \end{aligned} \tag{3.2}$$

[1] Here and in the following, we indicate variables and functions at a given iteration by using superscripts that are set in parentheses.

where the lower bound on the entropy, $\beta^{(i)} = \mathrm{H}\Big(q^{(i)}(\mathbf{x})\Big) - \gamma$, is computed at each iteration based on a hyper-parameter γ, and ϵ specifies the maximum allowable KL divergence. Hence, at each iteration, the entropy of the search distribution may not decrease by more than γ.

Introducing Lagrangian multipliers η, ω and λ, the Lagrangian function corresponding to optimization problem 3.2 is given by

$$\begin{aligned}\mathcal{L}(q, \eta, \beta, \omega) = \int_{\mathbf{x}} q(\mathbf{x}) R(\mathbf{x}) d\mathbf{x} + \eta \left(\epsilon - \mathrm{KL}\Big(q(\mathbf{x}) || q^{(i)}(\mathbf{x})\Big)\right) \\ + \omega \left(\mathrm{H}\Big(q(\mathbf{x})\Big) - \beta^{(i)}\right) + \lambda \left(1 - \int_{\mathbf{x}} q(\mathbf{x}) d\mathbf{x}\right).\end{aligned}$$

Maximizing the Lagrangian with respect to the search distribution q allows us to express the optimal search distribution $q^{(i+1)}$ as a function of the Lagrangian multipliers,

$$q^{(i+1)}(\mathbf{x}) \propto q^{(i)}(\mathbf{x})^{\frac{\eta}{\eta+\omega}} \exp\left(R(\mathbf{x})\right)^{\frac{1}{\eta+\omega}}. \tag{3.3}$$

The update according to Equation 3.3 can not be computed analytically for general choices of policies q and reward functions $R(\mathbf{x})$. MORE is therefore restricted to Gaussian search distributions $q(\mathbf{x}; \boldsymbol{\theta}^{(i)}) = \mathcal{N}\big(\mathbf{x}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\Sigma}^{(i)}\big)$ and optimizes a local, quadratic reward surrogate

$$\tilde{R}(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^\top \mathbf{R}^{(i)} \mathbf{x} + \mathbf{x}^\top \mathbf{r}^{(i)} + \mathrm{const.} \tag{3.4}$$

The parameters of the reward surrogate, $\mathbf{R}^{(i)}$ and $\mathbf{r}^{(i)}$, are learned using linear regression based on samples from the current approximation. For this choice of search distribution and reward surrogate, the updated distribution according to Equation 3.3 is also Gaussian with natural parameters

$$\mathbf{Q}(\eta, \omega) = \frac{\eta}{\eta+\omega}\mathbf{Q}^{(i)} + \frac{1}{\eta+\omega}\mathbf{R}^{(i)}, \tag{3.5}$$

$$\mathbf{q}(\eta, \omega) = \frac{\eta}{\eta+\omega}\mathbf{q}^{(i)} + \frac{1}{\eta+\omega}\mathbf{r}^{(i)}, \tag{3.6}$$

which directly relate to mean $\boldsymbol{\mu} = \mathbf{Q}^{-1}\mathbf{q}$ and covariance matrix $\boldsymbol{\Sigma} = \mathbf{Q}^{-1}$. It can be seen from Equation 3.5 and 3.6 that η controls the step size, whereas ω affects the entropy by scaling the covariance matrix without affecting the mean. The optimal parameters $\eta^\star$ and $\omega^\star$ can be learned by minimizing the convex dual objective

$$\begin{aligned}\mathcal{G}(\eta, \omega) =& \eta\epsilon - \omega\beta^{(i)} + \eta \log Z(\mathbf{Q}^{(i)}, \mathbf{q}^{(i)}) - (\eta+\omega) \log Z(\mathbf{Q}(\eta, \omega), \mathbf{q}(\eta, \omega)),\end{aligned}$$

where $\log Z(\mathbf{X}, \mathbf{x}) = -\frac{1}{2}(\mathbf{x}^\top \mathbf{X}^{-1}\mathbf{x} + \log|2\pi\mathbf{X}^{-1}|)$ is the log partition function of a Gaussian with natural parameters $\mathbf{X}$ and $\mathbf{x}$. This optimization can be performed very efficiently using the partial derivatives

$$\frac{\partial \mathcal{G}(\eta,\omega)}{\partial \eta} = \epsilon - \mathrm{KL}(q_{\eta,\omega}(\mathbf{x})||q(\mathbf{x};\boldsymbol{\theta}^{(i)})), \qquad \frac{\partial \mathcal{G}(\eta,\omega)}{\partial \omega} = \mathrm{H}(q_{\eta,\omega}(\mathbf{x})) - \beta,$$

where $q_{\eta,\omega}(\mathbf{x})$ refers to the Gaussian distribution with natural parameters computed according to Equation 3.5 and Equation 3.6. In the next section we introduce a slight variant of MORE that can be used for variational inference. The derivations of that variant are shown in Appendix B.1 and can be straightforwardly extended to derive the equations shown in this section.

3.1.3. Adapting MORE to Variational Inference

Inspired by policy search methods, we want to use information-geometric trust regions for variational inference in order to achieve efficient optimization while avoiding premature convergence. Hence, we want to compute each update of the approximation by solving the constrained optimization problem

$$\begin{aligned} \boldsymbol{\theta}^{(i+1)} = \ & \arg\max_{\boldsymbol{\theta}} \quad \int_{\mathbf{x}} q(\mathbf{x};\boldsymbol{\theta}) R(\mathbf{x}) d\mathbf{x} + \mathrm{H}(q(\mathbf{x};\boldsymbol{\theta})), \\ & \text{subject to} \ \ \mathrm{KL}\Big(q(\mathbf{x};\boldsymbol{\theta})||q(\mathbf{x};\boldsymbol{\theta}^{(i)})\Big) \leq \epsilon, \\ & \int_{\mathbf{x}} q(\mathbf{x};\boldsymbol{\theta}) d\mathbf{x} = 1. \end{aligned} \tag{3.7}$$

Optimization Problem 3.7 is very similar to Optimization Problem 3.2 solved by MORE and only differs due to the fact that the entropy of the search distribution does not enter the optimization problem as constraint, but as additional term in the objective. It can be solved analogously to MORE by introducing Lagrangian multipliers and minimizing the dual problem

$$\mathcal{G}(\eta) = \eta\epsilon + \eta \log Z(\mathbf{Q}^{(i)}, \mathbf{q}^{(i)}) - (\eta+1)\log Z(\mathbf{Q}(\eta,1), \mathbf{q}(\eta,1)), \tag{3.8}$$

using the gradient

$$\frac{d\mathcal{G}(\eta)}{d\eta} = \epsilon - \mathrm{KL}(q_{\eta,1}(\mathbf{x})||q(\mathbf{x};\boldsymbol{\theta}^{(i)})). \tag{3.9}$$

Here, the natural parameters $\mathbf{Q}(\eta,1)$ and $\mathbf{q}(\eta,1)$ for a given step size η are obtained by substituting $\omega = 1$ in Equation 3.5 and 3.6. Please refer to Appendix B.1 for the full derivations.

Hence, a Gaussian variational approximation can be learned analogously to MORE by iteratively (1) fitting a local, quadratic surrogate $\tilde{R}(\mathbf{x}) \approx \log \tilde{p}(\mathbf{x})$, (2) finding the optimal step size η by convex optimization and (3) updating the approximation based on Equation 3.5 and 3.6. The update of a Gaussian variational approximation given a quadratic reward surrogate is shown in Algorithm 1.

Algorithm 1 Updating a Gaussian variational approximation based on surrogate

Require: coefficients of quadratic surrogate $\mathbf{R}, \mathbf{r}$ (equation 3.4)
Require: current mean and covariance matrix $\boldsymbol{\mu}, \boldsymbol{\Sigma}$
Require: KL bound ϵ
function GVA_UPDATE($\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{R}, \mathbf{r}, \epsilon$)
  $\mathbf{Q} \leftarrow \boldsymbol{\Sigma}^{-1}$
  $\mathbf{q} \leftarrow \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}$
  $\eta \leftarrow$ minimize dual (Equation 3.8) using the gradient (Equation 3.9)
  $\mathbf{Q}' \leftarrow \frac{\eta}{\eta+1}\mathbf{Q} + \frac{1}{\eta+1}\mathbf{R}$
  $\mathbf{q}' \leftarrow \frac{\eta}{\eta+1}\mathbf{q} + \frac{1}{\eta+1}\mathbf{r}$
  $\boldsymbol{\Sigma}' \leftarrow \mathbf{Q}'^{-1}$
  $\boldsymbol{\mu}' \leftarrow \mathbf{Q}'^{-1}\mathbf{q}'$
  return $\boldsymbol{\Sigma}', \boldsymbol{\mu}'$
end function

3.2. Variational Inference by Policy Search

We showed in Section 3.1.3 that we can learn Gaussian variational approximations using our variant of MORE [Abdolmaleki et al., 2015]. However, Gaussian approximations can lead to high modeling errors, especially for multimodal target distributions. We will now derive VIPS++, a general-purpose method for learning GMM approximations of an unnormalized target distribution $\tilde{p}(\mathbf{x})$. In Section 3.2.1 we will show that an I-projection to a GMM can be decomposed into independent I-projections for its Gaussian components using a similar decomposition as used by expectation-maximization. In combination with our variant of MORE, this result enables us to learn GMM approximations with a fixed number of components. Sections 3.2.2, 3.2.3 and 3.2.4 discuss several extensions to this procedure that are critical for efficiently learning high quality approximations in practice. Namely, we will discuss reusing function evaluations from previous iterations, selecting relevant samples and dynamically adapting the number of components.

3.2.1. Learning a GMM Approximation

In order to represent high quality approximations of multimodal distributions, we want to learn a GMM approximation,

$$q(\mathbf{x};\boldsymbol{\theta}) = \sum_o q(o;\boldsymbol{\theta}) q(\mathbf{x}|o;\boldsymbol{\theta}),$$

where o is the index of the mixture component, $q(o;\boldsymbol{\theta})$ are the mixture weights and $q(\mathbf{x}|o;\boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_o, \boldsymbol{\Sigma}_o)$ is a multivariate normal distribution with mean $\boldsymbol{\mu}_o$ and full covariance matrix $\boldsymbol{\Sigma}_o$. The parameters $\boldsymbol{\theta}$ of our variational approximation are thus given by the mixture weights, means and covariance matrices. To improve readability we will often omit the parameter $\boldsymbol{\theta}$ when referring to the distribution q.

The approximation is learned by maximizing the ELBO

$$\begin{aligned} L(\boldsymbol{\theta}) &= \sum_o q(o) \int_{\mathbf{x}} q(\mathbf{x}|o)\big(R(\mathbf{x}) - \log q(\mathbf{x})\big) d\mathbf{x} \\ &= \sum_o q(o) \int_{\mathbf{x}} q(\mathbf{x}|o)\big(R(\mathbf{x}) - \log q(o) - \log q(\mathbf{x}|o) + \log q(o|\mathbf{x})\big) d\mathbf{x} \\ &= \sum_o q(o) \Big[\int_{\mathbf{x}} q(\mathbf{x}|o)\big(R(\mathbf{x}) + \log q(o|\mathbf{x})\big) d\mathbf{x} + \mathrm{H}\big(q(\mathbf{x}|o)\big) \Big] + \mathrm{H}\big(q(o)\big), \end{aligned} \tag{3.10}$$

where we used the identity

$$\log q(\mathbf{x}) = \log q(o) + \log q(\mathbf{x}|o) - \log q(o|\mathbf{x})$$

which can be derived from Bayes' rule.

Variational Lower Bound

Unfortunately, the occurrence of the *log responsibilities*, $\log q(o|\mathbf{x})$, in Equation 3.10 prevents us from optimizing each component independently. However, we can derive a lower bound $\tilde{L}(\boldsymbol{\theta}, \tilde{q}(o|\mathbf{x}))$ on the objective by adding and subtracting an auxiliary distribution

$\tilde{q}(o|\mathbf{x})$,

$$
\begin{aligned}
L(\boldsymbol{\theta}) &= \sum_o q(o)\Big[\int_{\mathbf{x}} q(\mathbf{x}|o)\big(R(\mathbf{x}) + \log q(o|\mathbf{x})\big)d\mathbf{x} + \mathrm{H}\big(q(\mathbf{x}|o)\big)\Big] + \mathrm{H}\big(q(o)\big) \\
&= \sum_o q(o)\Big[\int_{\mathbf{x}} q(\mathbf{x}|o)\big(R(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x}) + \log q(o|\mathbf{x}) - \log \tilde{q}(o|\mathbf{x})\big)d\mathbf{x} + \mathrm{H}\big(q(\mathbf{x}|o)\big)\Big] \\
&\quad + \mathrm{H}\big(q(o)\big) \\
&= \underbrace{\sum_o q(o)\Big[\int_{\mathbf{x}} q(\mathbf{x}|o)\big(R(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x})\big)d\mathbf{x} + \mathrm{H}\big(q(\mathbf{x}|o)\big)\Big] + \mathrm{H}\big(q(o)\big)}_{\tilde{L}(\boldsymbol{\theta}, \tilde{q}(o|\mathbf{x}))} \\
&\quad + \int_{\mathbf{x}} q(\mathbf{x})\mathrm{KL}\left(q(o|\mathbf{x})||\tilde{q}(o|\mathbf{x})\right) d\mathbf{x}.
\end{aligned}
\tag{3.11}
$$

Please note, that the last term in Equation 3.11 corresponds to an expected KL divergence and is therefore non-negative which implies that

$$\tilde{L}(\boldsymbol{\theta}, \tilde{q}(o|\mathbf{x})) \leq L(\boldsymbol{\theta}).$$

The decomposition in Equation 3.11 has already been previously applied in the broad context of variational inference [Agakov and Barber, 2004; Maaløe et al., 2016; Ranganath et al., 2016; Tran et al., 2016]. However, these approaches parameterize the auxiliary distribution and are not well-suited for learning accurate GMM approximations. In contrast, we exploit that the responsibilities $q(o|\mathbf{x})$ can be computed in closed form for Gaussian mixture models, which allows us to exactly tighten the lower bound similar to expectation-maximization [Bishop, 2006]. However, whereas EM minimizes the forward KL divergence, $\mathrm{KL}(p(\mathbf{x})||q(\mathbf{x};\boldsymbol{\theta}))$, for density estimation, our approach can be used for minimizing the reverse KL divergence, $\mathrm{KL}(q(\mathbf{x};\boldsymbol{\theta})||p(\mathbf{x}))$, in a variational inference setting. The forward KL divergence can be easier optimized when samples from the target distribution are available while the (unnormalized) target density function $\tilde{p}(\mathbf{x})$ is unavailable and is therefore well suited for density estimation. In contrast, the reverse KL divergence can be more easily optimized based on samples from the model only, when assuming access to the (unnormalized) target density function and is therefore well suited for variational inference.

Following the same reasoning as EM, we can show convergence to a stationary point of the ELBO $L(\boldsymbol{\theta})$ by iteratively setting $\tilde{q}(o|\mathbf{x}) = q(o|\mathbf{x})$ (analogously to an E-step) and increasing the lower bound $\tilde{L}(\boldsymbol{\theta}, \tilde{q}(o|\mathbf{x}))$ (M-step) while keeping the auxiliary distribution fixed. Tightening the lower bound by setting $\tilde{q}(o|\mathbf{x}) = q(o|\mathbf{x})$ does not affect the ELBO

since the parameters $\boldsymbol{\theta}$ are not changed. Increasing the lower bound increases both the lower bound and the expected KL divergence and thus also increases the ELBO. Such procedure strictly increases the ELBO until we reach a fixed point of the (hierarchical) lower bound optimization, that is,

$$\boldsymbol{\theta}^{(i)} = \arg\max_{\boldsymbol{\theta}} \tilde{L}\left(\boldsymbol{\theta}, q(\mathbf{x}, \boldsymbol{\theta}^{(i)})\right).$$

At such fixed point, the gradients of both terms of Equation 3.11 are zero (since they are both at an extremum) and thus the gradient of the ELBO is also zero.

In order to ensure monotonous improvement of the approximation, we need to ensure that the lower bound indeed increases during the M-Step. The lower bound $\tilde{L}(\boldsymbol{\theta}, \tilde{q}(o|\mathbf{x}))$, however, contains intractable integrals that need to be approximated based on samples. In order to keep the resulting approximation errors low, we need to stay close to the current set of samples. We therefore combine the iterative procedure with trust region optimization by bounding the change of each component during the M-step. For sufficiently small step sizes, such trust region updates ensure monotonous improvement [Akrour et al., 2018; Schulman et al., 2015]. Furthermore, such constrained maximization does not affect the theoretical guarantees of the iterative procedure as any increase of the lower bound ensures an increase of the ELBO.

M-Step for Component Updates

Maximizing the lower bound $\tilde{L}(\boldsymbol{\theta}, \tilde{q}(o|\mathbf{x}))$ with respect to the mean and covariance matrix $\boldsymbol{\theta}_o = [\boldsymbol{\mu}_o, \boldsymbol{\Sigma}_o]$ of an individual component is not affected by the mixture coefficients $q(o)$ or the parameters of the remaining components and can be performed independently and in parallel by maximizing the term inside the square brackets of Equation 3.11, that is,

$$\begin{aligned} &\arg\max_{\boldsymbol{\theta}_o} \quad \int_{\mathbf{x}} q(\mathbf{x}|o;\boldsymbol{\theta}_o)\big(R(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x})\big)d\mathbf{x} + \mathrm{H}\big(q(\mathbf{x}|o)\big), \\ &\text{subject to} \quad \mathrm{KL}\Big(q(\mathbf{x}|o;\boldsymbol{\theta}_o)||q(\mathbf{x}|o;\boldsymbol{\theta}^{(i)})\Big) \leq \epsilon(o), \end{aligned} \tag{3.12}$$

where we already added the trust region constraint for better exploration and stability. The upper bound on the Kullback-Leibler divergence, $\epsilon(o)$, is adapted during learning. If the Monte-Carlo estimate of the component-specific objective after the component update is smaller than the Monte-Carlo estimate before the update, we decrease $\epsilon(o)$ by multiplying it by 0.8; otherwise we increase it slightly by multiplying it by 1.1. The optimization problem can be solved using our variant of MORE (Equation 3.7) with a component specific reward function $R_o(\mathbf{x}) = R(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x})$. As the auxiliary distribution $\tilde{q}(o|\mathbf{x})$

was fixed to the responsibilities $q(o|\mathbf{x}; \boldsymbol{\theta}^{(i)})$ according to the previous mixture model, the component specific part of $R_o(\mathbf{x})$ penalizes each component for putting probability mass on areas that are already covered by other components.

For applying our variant of MORE, we need to fit a quadratic reward surrogate $\tilde{R}_o(\mathbf{x}) \approx R_o(\mathbf{x})$ that approximates the component specific reward $R_o(\mathbf{x})$ in the vicinity of the respective component $q(\mathbf{x}|o)$. The surrogate can be fit using ordinary least squares, where the independent variables are samples from the respective component and the dependent variables are the corresponding function evaluations of $R_o(\mathbf{x})$. However, because we want to use the same set of samples for all component updates as well as the weight update, we use weighted least squares based on importance weights which will be discussed in greater detail in Section 3.2.2. After fitting the surrogate, the optimization problem in Equation 3.12 can be solved efficiently using L-BFGS-B [Byrd et al., 1995] to minimize the dual problem (Equation 3.8) and using the learned step size η to compute the update in closed form as outlined in Section 3.1.3.

Drawing the connection to reinforcement learning and investigating the reward function $R_o(\mathbf{x})$ for a given component reveals that the proposed algorithm treats every component update as a reinforcement learning problem, where the reward is computed based on the achieved log-densities $\log \tilde{p}(\mathbf{x})$ with a penalty for sampling in regions that are already covered by other components due to low log responsibilities. Moreover, the components strive for high entropy which prevents them from always choosing the same sample.

M-Step for Weight Updates

After updating the individual components, we can keep the learned means and covariance matrices fixed while updating the mixture coefficients $q(o)$. As shown in previous work [Arenz et al., 2018], we can also enforce an information-geometric trust region for the weight update. However, in subsequent experiments we could not show a significant effect of such constraint and will therefore only consider the unconstrained optimization. The M-step with respect to the mixture coefficients is thus framed as

$$\arg\max_{q(o)} \sum_o q(o)R(o) + \mathrm{H}\big(q(o)\big), \tag{3.13}$$

where the objective for the component update,

$$R(o) = \int_{\mathbf{x}} q(\mathbf{x}|o)\big(R(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x})\big)d\mathbf{x} + \mathrm{H}\big(q(\mathbf{x}|o)\big), \tag{3.14}$$

serves as reward for choosing component o. The reward $R(o)$ contains an intractable integral, and thus it needs to be approximated from samples. It is to note that $R(o)$

corresponds to a discrete function, which can be represented by a vector, whereas the reward function $R_o(\mathbf{x})$ used for the component update is a continuous function. It is not beneficial to approximate $R(o)$ based on a quadratic surrogate of $R_o(\mathbf{x})$, since we can estimate each element of the vector more efficiently and more accurately using a Monte-Carlo estimate

$$\tilde{R}(o) = \frac{1}{N_o} \sum_{n=1}^{N_o} \left[R(\mathbf{x}_{o,n}) + \log \tilde{q}(o|\mathbf{x}_{o,n}) \right] + \mathrm{H}(q(\mathbf{x}|o)), \tag{3.15}$$

where $\mathbf{x}_{o,n}$ refers to the nth of N_o samples from component $q(\mathbf{x}|o)$. We will discuss in Section 3.2.2 how we use importance weighting to estimate the reward of each component based on the same set of samples that is used for the component update.

Based on the approximated rewards $\tilde{R}(o)$, the optimal solution of optimization problem in Equation 3.13 is given in closed form as

$$q(o) = \frac{\exp\left(\tilde{R}(o)\right)}{\sum_o \exp\left(\tilde{R}(o)\right)}. \tag{3.16}$$

The weight optimization can also be treated as a reinforcement-learning problem, where actions correspond to choosing components and the agent gets rewarded for choosing components that sample in important regions, that do not interfere with other components and that have high entropy. The agent itself also strives for high entropy and will thus make use of every component.

The complete optimization can be treated as a method for hierarchical reinforcement learning where we learn both, a higher level policy $q(o)$ over options and Gaussian lower level policies $q(\mathbf{x}|o)$. However, since our approach does not consider time series data, it mainly relates to black-box approaches to reinforcement learning that use stochastic optimizers such as ARS, NES or MORE [Abdolmaleki et al., 2015; Mania et al., 2018; Salimans et al., 2017]. HiREPS [Daniel et al., 2012] already applied black-box optimization for learning GMM policies based on episodic REPS [Peters et al., 2010].

The basic variant of our method is shown in Algorithm 2. The individual component updates (line 3-8) are performed by sampling from the respective components (line 3), evaluating the samples on the target distribution (line 4), computing the log responsibilities $\log \tilde{q}(o|\mathbf{x})$ according to the previous approximation (line 5), fitting the reward surrogate (line 6-7) and performing the trust region update (line 8). The components can be updated in parallel since the responsibilities are computed based on the same mixture parameters $\boldsymbol{\theta}$. The weight update (line 11-17) is computed based on Equation 3.16 (line 17) using the Monte-Carlo estimates of the component rewards (line 15). Updating the parameters

of the GMM in between the component updates and the weight update (line 10) is optional and relates to an additional E-Step in EM, which does not affect the theoretical guarantees [Neal and Hinton, 1998].

Algorithm 2 Variational Inference by Policy Search (Basic Variant)

Require: number of components N_o
Require: initial mixture parameters $\boldsymbol{\theta} = \{q(o), \boldsymbol{\mu}_{o,\ldots,N_o}, \boldsymbol{\Sigma}_{o,\ldots,N_o}\}$
Require: number of iterations N_i
Require: number of samples per component N_s
1: **for** $i = 1 \ldots N_i$ **do**
2: **for** $o = 1 \ldots N_o$ **do**
3: $\mathcal{X}_o \leftarrow$SAMPLE_GAUSSIAN$(\boldsymbol{\mu}_o, \boldsymbol{\Sigma}_o, N_s)$
4: $\tilde{\mathbf{p}}_o \leftarrow \log \tilde{p}(\mathcal{X}_\mathbf{o})$ ▷ EVALUATE TARGET LOG LIKELIHOOD FOR EACH SAMPLE
5: $\tilde{\mathbf{q}}_{o|x} \leftarrow \log q(\mathcal{X}_o, o; \boldsymbol{\theta}) - \log q(\mathcal{X}_o; \boldsymbol{\theta})$ ▷ EVALUATE LOG RESPONSIBILITIES
6: $\mathbf{y}_o \leftarrow \tilde{\mathbf{p}}_o + \tilde{\mathbf{q}}_{o|x}$ ▷ COMPUTE TARGETS FOR ORDINARY LEAST SQUARES (OLS)
7: $\mathbf{R}_o$,$\mathbf{r}_o \leftarrow$OLS$(\mathcal{X}_o, \mathbf{y}_o)$ ▷ LEARN QUADRATIC SURROGATE
8: $\boldsymbol{\mu}'_o, \boldsymbol{\Sigma}'_o \leftarrow$GVA_UPDATE$(\boldsymbol{\mu_o}, \boldsymbol{\Sigma_o}, \mathbf{R}_o, \mathbf{r}_o, \epsilon_o)$ ▷ ALGORITHM 1
9: **END FOR**
10: $\boldsymbol{\theta} \leftarrow$ UPDATE_COMPONENTS$(\boldsymbol{\theta}, \boldsymbol{\mu}'_{o,\ldots,N_o}, \boldsymbol{\Sigma}'_{o,\ldots,N_o})$
11: **FOR** $o = 1 \ldots N_o$ **DO**
12: $\mathcal{X}_o \leftarrow$SAMPLE_GAUSSIAN$(\boldsymbol{\mu}_o, \boldsymbol{\Sigma}_o, N_s)$
13: $\tilde{\mathbf{p}}_o \leftarrow \log \tilde{p}(\mathcal{X}_\mathbf{o})$ ▷ EVALUATE TARGET LOG LIKELIHOOD FOR EACH SAMPLE
14: $\tilde{\mathbf{q}}_{o|x} \leftarrow \log q(\mathcal{X}_o, o; \boldsymbol{\theta}) - \log q(\mathcal{X}_o; \boldsymbol{\theta})$ ▷ EVALUATE LOG RESPONSIBILITIES
15: $\tilde{R}_o \leftarrow {N_s}^{-1}$ SUM$(\tilde{\mathbf{p}}_o + \tilde{\mathbf{q}}_{o|x}) + \mathrm{H}(\boldsymbol{\Sigma}_o)$ ▷ ESTIMATE REWARD (EQUATION 3.15)
16: **END FOR**
17: $q'(o) \leftarrow \frac{\exp(\tilde{R}_o)}{\sum_o \exp(\tilde{R}_o)}$
18: $\boldsymbol{\theta} \leftarrow$ UPDATE_WEIGHTS$(\boldsymbol{\theta}, q'(o))$
19: **END FOR**

3.2.2. Sample Reuse by Importance Weighting

VIPS relies on samples for approximating the reward for choosing a given component, $R(o)$, and for computing the quadratic surrogates for the component update. These samples need to be evaluated on the unnormalized target distribution $\tilde{p}(\mathbf{x})$ which may be costly. In order to reduce the number of function evaluations we want to also make use of samples from previous iterations, which can be achieved by using importance weighting.

We will now show how importance weights can be used to approximate the rewards for the weight updates and how to learn the quadratic surrogates for the component update based on the same subset $\boldsymbol{\mathcal{X}}_{\subset}$ of samples.

Importance Weighting for Updating the Mixture Weights

Importance sampling is a technique for estimating the expected value $\mathrm{E}_q[f(\mathbf{x})]$ of a given function $f(\mathbf{x})$ with respect to a distribution $q(\mathbf{x})$ while using samples from a different distribution $z(\mathbf{x}) \neq q(\mathbf{x})$. Assuming that the support of $z(\mathbf{x})$ covers the support of $q(\mathbf{x})$, we can express the desired expectation as

$$\mathrm{E}_q[f(\mathbf{x})] = \int_{\mathbf{x}} q(\mathbf{x}) f(\mathbf{x}) d\mathbf{x} = \int_{\mathbf{x}} z(\mathbf{x}) \frac{q(\mathbf{x})}{z(\mathbf{x})} f(\mathbf{x}) d\mathbf{x} = \mathrm{E}_z[w(\mathbf{x}) f(\mathbf{x})],$$

using *importance weights* $w(\mathbf{x}) = \frac{q(\mathbf{x})}{z(\mathbf{x})}$. Hence, the desired expectation can be approximated by using a Monte-Carlo estimate based on N_z samples from the sampling distribution $z(\mathbf{x})$,

$$\mathrm{E}_q[f(\mathbf{x})] \approx \sum_{i=1}^{N_z} \frac{1}{N_z} w(\mathbf{x}_i) f(\mathbf{x}_i). \tag{3.17}$$

Instead of using the estimator given by Equation 3.17, it is also common to use *self-normalized importance sampling*

$$\mathrm{E}_q[f(\mathbf{x})] \approx \sum_{i=1}^{N_z} \bar{w}(\mathbf{x}_i) f(\mathbf{x}_i), \qquad \bar{w}(\mathbf{x}_i) = \left(\sum_{i=1}^{N_z} \frac{q(\mathbf{x}_i)}{z(\mathbf{x}_i)} \right)^{-1} \frac{q(\mathbf{x}_i)}{z(\mathbf{x}_i)}.$$

Self-normalized importance sampling introduces a bias that is asymptotically zero since $\lim_{N_z \to \infty} \sum_{i=1}^{N_z} \frac{q(\mathbf{x}_i)}{z(\mathbf{x}_i)} = N_z$, but it has the advantages that it is consistent for different constant offsets on the function $f(\mathbf{x})$ and that it is also applicable if the target distribution is not normalized.

An important consideration for choosing the sampling distribution is the variance of the estimator. In general, the estimator's variance can be significantly worse than standard Monte-Carlo [Hesterberg, 1988]. When using samples from the desired distribution, that is, $z(\mathbf{x}) = q(\mathbf{x})$, the importance weighted estimate and the self-normalized estimate are both equivalent to standard Monte-Carlo. However, it is also possible to obtain lower variance than standard Monte-Carlo, for example, when using the optimal sampling distribution

$$z(\mathbf{x}) = \frac{1}{C} q(\mathbf{x}) |f(\mathbf{x}) - c|, \tag{3.18}$$

where C is a normalizing constant and $c = 0$ for importance sampling and $c = \mathrm{E}_q[f(\mathbf{x})]$ for self-normalized importance sampling [Hesterberg, 1988]. If the function $f(\mathbf{x})$ is positive everywhere, the former estimate has even zero variance since

$$w(\mathbf{x}_i)f(\mathbf{x}_i) = \frac{q(\mathbf{x}_i)}{z(\mathbf{x}_i)}f(\mathbf{x}_i) = C\frac{q(\mathbf{x}_i)}{q(\mathbf{x}_i)f(\mathbf{x}_i)}f(\mathbf{x}_i) = C = \int_{\mathbf{x}} q(\mathbf{x})f(\mathbf{x})d\mathbf{x} = \mathrm{E}_q[f(\mathbf{x})].$$

Although the optimal sampling distributions according to Equation 3.18 are intractable as they depend on the expectation $\mathrm{E}_q[f(\mathbf{x})]$, which is the value of interest, they can be useful for designing appropriate sampling distributions.

In order to estimate the expected reward $R(o)$ using a subset $\mathcal{X}_\subset$ of the samples from previous iterations $\mathcal{X}$ we need to evaluate the respective sampling distribution $z_\subset(\mathbf{x})$ for computing the importance weights. For that purpose, we store all samples together with the respective unnormalized target densities and the parameters of the component from which it was sampled in a database

$$\mathcal{S} = \{(\mathbf{x}_0, \log\tilde{p}(\mathbf{x_0}), \mathcal{N}_{\mathbf{x}_0}), \ldots, (\mathbf{x}_N, \log\tilde{p}(\mathbf{x_N}), \mathcal{N}_{\mathbf{x}_N})\},$$

where $N_\mathbf{x}$ refers to the Gaussian distribution that was used for obtaining the sample $\mathbf{x}$. By also storing its respective Gaussian distributions, we can represent the sampling distribution as a Gaussian mixture model $z^\subset(\mathbf{x})$ that contains for each sample $\mathbf{x}_s \in \boldsymbol{\mathcal{X}}_\subset$ the respective Gaussian distribution $\mathcal{N}_{\mathbf{x}_s}(\mathbf{x})$, that is,

$$z_\subset(\mathbf{x}) = \sum_{\mathbf{x}_s \in \boldsymbol{\mathcal{X}}_\subset} \frac{1}{|\boldsymbol{\mathcal{X}}_\subset|}\mathcal{N}_{\mathbf{x}_s}(\mathbf{x}).$$

Please note, that in practice, we represent the GMM $z_\subset(\mathbf{x})$ more concisely by exploiting that usually several samples were drawn from the same Gaussian distribution. We estimate the reward $R_o(\mathbf{x})$ for each component using self-normalized importance sampling, that is,

$$\tilde{R}(o) = \sum_{\mathbf{x}_s \in \boldsymbol{\mathcal{X}}_\subset} \bar{w}_o(\mathbf{x}_s)\big[R(\mathbf{x}_s) + \log\tilde{q}(o|\mathbf{x}_s)\big] + \mathrm{H}(q(\mathbf{x}|o)).$$

where the self-normalized importance weights for component o are given by

$$\bar{w}_o(\mathbf{x}_s) = \frac{1}{Z}\frac{q(\mathbf{x}_s|o)}{z_\subset(\mathbf{x}_s)}, \qquad Z = \sum_{\mathbf{x}_s \in \boldsymbol{\mathcal{X}}_\subset} \frac{q(\mathbf{x}_s|o)}{z_\subset(\mathbf{x}_s)}.$$

We could choose different subsets, depending on the component for which we want to estimate the reward $R(o)$. However, because we need to evaluate each sample on any component anyway in order to compute the responsibilities $q(o|\mathbf{x})$, we use the same subset $\boldsymbol{\mathcal{X}}_\subset$ for estimating all component rewards as well as the surrogate models.

Importance Weighting for Fitting the Quadratic Surrogates

For updating the individual components we need to learn local quadratic surrogates $\tilde{R}_o(\mathbf{x})$ in the vicinity of the respective components. MORE achieves locality by using samples from the respective component $q(\mathbf{x}|o)$ as independent variables for ordinary least-squares. Learning the surrogate based on samples from a different distribution $z_\subset(\mathbf{x})$ introduces covariate shift, that is, the distribution of the training data $z_\subset(\mathbf{x})$ does not match the distribution of the test data $q(\mathbf{x}|o)$. The covariate shift can be accommodated by minimizing a weighted least-squares problem [Chen et al., 2016]

$$\arg\min_{\boldsymbol{\beta}_o} \mathrm{E}_{z_\subset}\left[\underbrace{\frac{q(\mathbf{x}|o)}{z_\subset(\mathbf{x})}}_{\bar{w}_o(\mathbf{x}_s)}\left(R_o(\mathbf{x}) - \tilde{R}_o(\mathbf{x};\boldsymbol{\beta}_o)\right)^2\right],$$

where the quadratic surrogate $\tilde{R}_o(\mathbf{x};\beta_o)$ is linear in the parameters $\boldsymbol{\beta}_o$. In practice, we also perform ℓ_2-regularization with *ridge coefficient* κ_o. The optimal parameters are thus given by

$$\boldsymbol{\beta}_o = (\mathbf{X}^\top \mathbf{W}_o \mathbf{X} + \kappa_o \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{W}_o \mathbf{y},$$

where $\mathbf{X}$ is the design matrix where each row contains the linear and quadratic features for the respective sample $\mathbf{x}_s \in \boldsymbol{\mathcal{X}}_\subset$ as well as a constant feature, $\mathbf{W}_o$ is a diagonal matrix where each element relates to the respective self-normalized importance weight $w_o(\mathbf{x}_s)$, $\mathbf{y}$ is a vector containing the targets $y_s = R(\mathbf{x}_s) + \log q(o|\mathbf{x}_s)$ and $\boldsymbol{\beta}_o$ is a vector containing the elements of $\mathbf{r}_o$ and $\mathbf{R}_o$ as well as a constant offset that can be discarded. Specifying an appropriate ridge coefficient κ_o can be difficult as different components may require different amounts of regularization. We therefore adapt the coefficient during optimization by multiplying it by 10 if the matrix inversion failed and by dividing it by 2 if it succeeded.

Although we use importance weights for learning the surrogates, we do not aim to estimate an expected value. The minimum-variance sampling distributions given by Equation 3.18 are in general not useful for learning accurate surrogate models as they focus on bringing the weighted function evaluation $w(\mathbf{x})f(\mathbf{x})$ close to the expected value, rather than aiming to accurately represent the function's landscape. Instead, we aim to construct a sampling distribution $z_\subset(\mathbf{x})$ that covers all components of the current approximation well. Such sampling distribution ensures that the importance weighted estimates are not much worse than Monte-Carlo estimations, both, for estimating the expected rewards $\tilde{R}(o)$ and for learning locally valid surrogate models $\tilde{R}_o(\mathbf{x})$. In the next section, we will discuss a heuristic for constructing such sampling distribution.

3.2.3. Sample Selection

Using all previous samples in each iteration would be computationally costly. Instead, we want to select a small set of samples such that we can get good approximations of all surrogate models and component rewards while requiring only a small number of new samples from each component. A common technique that was used in CMA-ES [Shirakawa et al., 2015], MORE [Abdolmaleki et al., 2015] and VIPS [Arenz et al., 2018] is to reuse all samples from the k latest iterations, where k is a hyper-parameter to balance between sample efficiency and computational efficiency. As the components that were used for the most recent iterations were similar to the current components, the reused samples can usually provide meaningful information about the target distribution in the vicinity of the respective components. However, we noticed that such procedure can be wasteful when optimizing large GMMs if some component have already converged and others still need to improve. For example, we typically have enough samples in the database to estimate the reward and local surrogate for components that did not significantly change during several iterations even without requiring any new samples; yet, when only using the latest k samples we need to continuously sample from each component during the whole optimization in order to maintain stability.

In order to avoid discarding old samples, we could sub-sample uniformly among the sample database. However, such procedure can result in a large number of irrelevant samples and, furthermore, does not ensure that the relevant samples are evenly distributed among the components of the current approximation. A more sophisticated method was presented by Uchibe [2018] in the context of policy search. Instead of sub-sampling uniformly, they treat all components in the database as components of a mixture model, $q^{\alpha}_{\text{sampling}}(\mathbf{x})$, and optimize the corresponding mixture coefficients α such that the model is close to the optimal sampling distribution given by Equation 3.18. However, the resulting sampling distribution might not be suited for learning the surrogate models, and, furthermore, such approach would be computational intractable because, by optimizing a GMM, VIPS may add up to several hundreds of components to the database in each iteration and would also need to identify an optimal sampling distribution for each of the respective components.

Furthermore, it is hard to make use of function evaluations such as $\tilde{p}(\mathbf{x}_i)$ or $q(\mathbf{x}_i|o)$ for deciding whether to reuse a given sample $\mathbf{x}_i$ without introducing additional bias in the importance sampling estimate. When such function evaluations influence our decision to use a given sample $\mathbf{x}_i$ for importance weighting, we can no longer consider it as an unbiased draw from $\mathcal{N}_{\mathbf{x}_i}(\mathbf{x})$ and computing the importance weights based on the background distribution $z_{\subset}(\mathbf{x})$ would, thus, not be admissible.

Instead, we propose to identify for each component $q(\mathbf{x}|o)$ of the current approximation

those components in the database $\mathcal{N}_{\mathbf{x}_i}(\mathbf{x})$ that are close according to a given dissimilarity measure $d\big(q(\mathbf{x}|o), \mathcal{N}_{\mathbf{x}_i}(\mathbf{x})\big)$ that is independent of the actual samples drawn from $\mathcal{N}_{\mathbf{x}_i}(\mathbf{x})$. In order to reduce the risk of selecting the same samples in each iteration, which may result in overfitting, we iteratively sample (without replacement) components from our database according to

$$h(i, o) \propto \exp\left(-d\big(q(\mathbf{x}|o), \mathcal{N}_{\mathbf{x}_i}(\mathbf{x})\big) - n_i\right), \tag{3.19}$$

where n_i keeps track of the number of times the samples of distribution $\mathcal{N}_{\mathbf{x}_i}(\mathbf{x})$ have been reused. We add all samples from the chosen component to the active set of samples $\mathcal{X}_\subset$ and stop sampling distributions when a desired number of reused samples n_{reused} is reached. This process is performed for each component $q(\mathbf{x}|o)$ of the current approximation.

A natural choice for the dissimilarity is to use the Kullback-Leibler divergence,

$$d_{\text{KL}}\big(q(\mathbf{x}|o), \mathcal{N}_{\mathbf{x}_i}(\mathbf{x})\big) = \text{KL}\big(q(\mathbf{x}|o)||\mathcal{N}_{\mathbf{x}_i}(\mathbf{x})\big),$$

which favors sampling distributions $\mathcal{N}_{\mathbf{x}_i}(\mathbf{x})$ that cover the respective mixture component $q(\mathbf{x}|o)$ well. However, even though the KL divergence between two Gaussian distributions can be computed in closed form, computing it for every component in the current approximation with respect to every component in the database can quickly become the computational bottleneck of the whole optimization.

Instead, VIPS++ computes the dissimilarity as the negative Mahalanobis distance of the mean μ_i of the sampling distribution $\mathcal{N}_{\mathbf{x}_i}(\mathbf{x})$ with respect to the given component $q(\mathbf{x}|o)$, that is,

$$d_{\text{Mahalanobis}}\big(q(\mathbf{x}|o), \mathcal{N}_{\mathbf{x}_i}(\mathbf{x})\big) = -\log p(\boldsymbol{\mu}_i|o).$$

While neglecting the covariance matrix of the sampling distribution may appear too crude, we argue that it is necessary to stay within a reasonable computational budget for selecting relevant samples. We demonstrate in Section 3.4.2 that the proposed selection strategy is able to identify relevant samples for each component $q(\mathbf{x}|o)$ among all previous samples without adding significant computational overhead. We also compare the Mahalanobis distance to different dissimilarity measures, namely, forward and reverse KL, as well as uniform selection in Appendix B.2. Pseudo-code for identifying relevant samples is shown in Appendix B.3.

Drawing new samples

After selecting the set $\boldsymbol{\mathcal{X}}_\subset$ of samples to be reused during the current iteration, we need to draw new samples from those components that are not sufficiently covered. A useful

diagnostic for monitoring the quality of the chosen sampling distribution is the *effective sample size*

$$n_{\text{eff}}(o) = \Big(\sum_{\mathbf{x}_s \in \boldsymbol{\mathcal{X}}_\subset} \bar{w}_o(\mathbf{x}_s)^2 \Big)^{-1},$$

which approximates the number of samples that standard Monte-Carlo would require to achieve the same variance as the importance sampling estimate [Djuric et al., 2003; Kong et al., 1994].

Hence, we compute for each component the number of effective samples, and draw $n_{\text{new}}(o) = n_{\text{des}} - \lfloor n_{\text{eff}}(o) \rfloor$ new samples, such that its effective sample size should approximately match a specified desired number of effective samples n_{des}. These samples are added to the database and to the set of active samples $\boldsymbol{\mathcal{X}}_\subset$ as illustrated in Algorithm 3.

Algorithm 3 Ensure that every component has sufficiently many effective samples.

Require: database $\mathcal{S} = \{(\mathbf{x}_0, \log \tilde{p}(\mathbf{x_0}), \mathcal{N}_{\mathbf{x_0}}), \ldots, (\mathbf{x}_N, \log \tilde{p}(\mathbf{x_N}), \mathcal{N}_{\mathbf{x_N}})\}$
Require: Set of chosen samples $\boldsymbol{\mathcal{X}}_\subset$, respective self-normalized importance weights $w_o(\mathbf{x})$
Require: desired number of effective samples per component n_{des}

function SAMPLE_WHERE_NEEDED
  for $o = 1 \ldots N_o$ **do**
    $n_{\text{eff}}(o) \leftarrow \Big(\sum_{\mathbf{x}_s \in \boldsymbol{\mathcal{X}}_\subset} w_o(\mathbf{x}_s)^2 \Big)^{-1}$
    $n_{\text{new}}(o) \leftarrow n_{\text{des}} - \lfloor n_{\text{eff}}(o) \rfloor$
    $\boldsymbol{\mathcal{X}}_{\text{new},o} \leftarrow$ SAMPLE_GAUSSIAN$(\boldsymbol{\mu}_o, \boldsymbol{\Sigma_o}, n_{\text{new}}(o))$
    FOR $\mathbf{x}_s$ **IN** $\boldsymbol{\mathcal{X}}_{\text{NEW},o}$ **DO**
      $\mathcal{S} \leftarrow \mathcal{S} \cup \{(\mathbf{x}_s, \log \tilde{p}(\mathbf{x}_s), \mathcal{N}_{\mathbf{x}_s})\}$
    END FOR
    $\boldsymbol{\mathcal{X}}_\subset \leftarrow \boldsymbol{\mathcal{X}}_\subset \cup \boldsymbol{\mathcal{X}}_{\text{new},o}$
  END FOR
  RETURN $\boldsymbol{\mathcal{X}}_\subset$
END FUNCTION

3.2.4. Adapting the Number of Components

The component optimization (Algorithm 1) is a local optimization, and the component will typically converge to a nearby mode (although the trust region constraint may help to traverse several poor optima). The quality of the learned approximation thus depends crucially on the initialization of the mixture model. However, the modes of the target distribution are often not known a priori and have to be discovered during optimization.

We therefore adapt the number of components dynamically by adding new components in promising regions and by deleting components with very low weight. The number of components is adapted at the beginning of each learning iteration before obtaining new samples. By always assigning low weight to newly added components and by only deleting components that have low weight, the effect on the approximation is negligible and the stability of the optimization is thus not affected.

Deleting Bad Components

Components that have been initialized at poor locations may converge to irrelevant modes of the target distribution and get very low weights such that they do not affect the approximation in practice. As keeping such components would add unnecessary computational overhead, we delete any component that had low weight for a given number of iterations, n_{del}, and that further did not increase its expected reward $\tilde{R}(o)$ during that period.

Initializing the Mean of New Components

By adding components to the mixture model, we can increase the representational power and thus improve the quality of the approximation. Furthermore, adding components affects the search distribution and can thus be used for exploration. In either case, we want the new components to eventually contribute to the approximation and hence achieve high weight $q(o) \propto \exp(R(o))$ and thus high reward $R(o)$. We treat every sample $\mathbf{x}_s$ in the database as candidate for the initial mean of the new component and then select the most promising candidate according to an estimate of its initial reward. As we will discuss in Section 3.2.4, we will decide on the initial entropy irrespective of the initial mean, but we will choose the exact initial covariance only after deciding for an initial mean. Hence, in the following we will derive an estimate of the initial reward that depends on the initial mean and initial entropy H_{init}, but not on the covariance matrix.

Let $q_{\mathbf{x}_s}(\mathbf{x}|o_n)$ denote the new component o_n assuming that its mean was initialized at location $\boldsymbol{\mu}_n = \mathbf{x}_s$ and let $q_{\mathbf{x}_s}(\mathbf{x}) = (1 - q(o_n))q(\mathbf{x}) + q(o_n)q_{\mathbf{x}_s}(\mathbf{x}|o_n)$ denote the GMM approximation after adding the new component with initial weight $q(o_n)$. According to

Equation 3.14 the initial reward of the new component $R_{\mathbf{x}_s}(o_n)$ would be given by

$$
\begin{aligned}
R_{\mathbf{x}_s}(o_n) &= \int_{\mathbf{x}} q_{\mathbf{x}_s}(\mathbf{x}|o_n)\big(R(\mathbf{x}) + \log q_{\mathbf{x}_s}(o_n|\mathbf{x})\big)d\mathbf{x} + \mathrm{H}\big(q_{\mathbf{x}_s}(\mathbf{x}|o_n)\big) \qquad (3.20)\\
&= \int_{\mathbf{x}} q_{\mathbf{x}_s}(\mathbf{x}|o_n)\big(R(\mathbf{x}) + \log q(o_n) + \log q_{\mathbf{x}_s}(\mathbf{x}|o_n) - \log q_{\mathbf{x}_s}(\mathbf{x})\big)d\mathbf{x}\\
&\quad + \mathrm{H}\big(q_{\mathbf{x}_s}(\mathbf{x}|o_n)\big)d\mathbf{x}\\
&= \log q(o_n) + \int_{\mathbf{x}} q_{\mathbf{x}_s}(\mathbf{x}|o_n)\big(R(\mathbf{x}) - \log q_{\mathbf{x}_s}(\mathbf{x})\big)d\mathbf{x}\\
&= \log q(o_n) + \int_{\mathbf{x}} q_{\mathbf{x}_s}(\mathbf{x}|o_n)\big(R(\mathbf{x}) - \log\big((1-q(o_n))q(\mathbf{x}) + q(o_n)q_{\mathbf{x}_s}(\mathbf{x}|o_n)\big)\big)d\mathbf{x}. \qquad (3.21)
\end{aligned}
$$

Based on Equation 3.21 we can estimate the initial reward depending on the initial weight of the new component $q(o_n)$, the *current* mixture model $q(\mathbf{x})$, the target distribution $R(\mathbf{x})$, and the new component $q_{\mathbf{x}_s}(\mathbf{x}|o_n)$. The first term can be ignored because we choose the initial weight of the new component irrespective of its mean and a constant offset does not affect which initial mean achieves the maximum initial reward. The integral is intractable but can be approximated based on the sample $\mathbf{x}_s = \boldsymbol{\mu}_n$ as

$$
\tilde{R}_{\mathbf{x}_s}(o_\mathrm{n}) = R(\mathbf{x}_s) - \log\left((1-q(o_n))q(\mathbf{x}_s) + q(o_n)\exp\left(\frac{1}{2}D - \mathrm{H}_{\text{init}}\right)\right) \qquad (3.22)
$$

where we exploit that the Gaussian density at its mean can be computed based on its entropy H_{init} and the number of dimensions D, that is, $\log q_{\mathbf{x}_s}(\mathbf{x}_s|o_n) = \frac{1}{2}D - \mathrm{H}_{\text{init}}$. As the function evaluations $R(\mathbf{x}_s)$ of the target distribution are stored in the database, we only need to evaluate the current mixture model $q(\mathbf{x})$ on all candidate samples $\mathbf{x}_s$ to estimate the initial reward for these locations.

To investigate the approximated reward in Equation 3.22 we note that the second term corresponds to a log-sum-exp (LSE), that is,

$$
\begin{aligned}
\tilde{R}_{\mathbf{x}_s}(o_\mathrm{n}) &= R(\mathbf{x}_s) - \log\big((1-q(o_n))q(\mathbf{x}_s) + q(o_n)q_{\mathbf{x}_s}(\mathbf{x}_s|o_n)\big)\\
&= R(\mathbf{x}_s) - \mathrm{LSE}\big(\log(1-q(o_n)) + \log q(\mathbf{x}_s), \log q(o_n) + \log q_{\mathbf{x}_s}(\mathbf{x}_s|o_n)\big)\\
&\approx R(\mathbf{x}_s) - \max\left(\log q(\mathbf{x}_s), \log q(o_n) + \log q_{\mathbf{x}_s}(\mathbf{x}_s|o_n)\right), \qquad (3.23)
\end{aligned}
$$

where we exploit that $\mathrm{LSE}(a_1, a_2, \ldots, a_n) = \log\sum_{i=1}^{n}\exp(a_i)$ behaves similar to a maximum and that $(1-q(o_n))q(\mathbf{x}) \approx q(\mathbf{x})$, since we initialize the new component with negligible weight, $q(o_n) \approx 0$. Although the effect of the initial weight on the first operand

of the log-sum-exp is negligible, it may have considerable effect on the second operand because the logarithm of small values is a large negative value. Hence, the initial weight that we choose for a new component may affect its approximated reward, which can be explained by its effect on the responsibilities $q_{\mathbf{x}_s}(o_n|\mathbf{x}_s)$ in Eq. 3.20.

If we would add the new components with an initial weight of zero, the maximum-operator would always return the first operand and the proposed estimate (which ignores the constant offset $\log q(o_n) = -\infty$ in Eq. 3.22) of the initial reward would return the amount of missing log probability-density, $\tilde{R}_{\mathbf{x}_s}(o_\text{n}|q(o_n) = 0) = R(\mathbf{x}_s) - \log q(\mathbf{x}_s)$. Adding a new component at the location where our current approximation misses most log probability density seems sensible. However, the problem of such heuristic becomes evident when considering target distributions with heavy tails. In such cases, the amount of missing log probability density increases the farther we move away from the current approximation. The new component might, thus, be added in a region where the target distribution has low probability density, since the current approximation might have even lower probability density.

This failure case is a direct consequence of ignoring the effect of the new component on the mixture model. When considering non-zero weights, the log-responsibilities of the new component are finite and tend to increase the farther we move away from the current approximation. Yet, they saturate at $\log q_{\mathbf{x}_s}(o_n|\mathbf{x}_s) \approx 0$ for every candidate location $\mathbf{x}_s$ that is sufficiently far from the current approximation, that is, where $q(\mathbf{x}_s) \approx 0$. This behavior is reflected by the log-sum-exp in Equation 3.22, which provides additional reward based on the negative log probability density $-\log q(\mathbf{x}_s)$ of the current approximation but never much more than $-(\log q(o_n) + \log q_{\mathbf{x}_s}(\mathbf{x}_s|o_n))$.

The proposed heuristic has different effects depending on the choice of the initial weight, which upper-bounds the benefit of adding a component far from the current approximation to $-(\log q(o_n) + \log q_{\mathbf{x}_s}(\mathbf{x}_s|o_n))$ (Eq. 3.23). Small initial weights increase this threshold and, thus, the proposed heuristic becomes more explorative by tending to initialize new components far from the current approximation. However, a benefit of the proposed heuristic is that it often does not rely on a specific threshold to propose useful candidate locations. For example, when a candidate is very close to a mode of the target distribution that is currently not covered by the approximation, the heuristic will often choose it for a large range of different thresholds that might vary across several orders or magnitude. If there is no clear winner, the choice of $\log q(o_n)$ typically affects the proposed location. For relatively large initial weights, we will create the component at a location where $R(\mathbf{x}_s)$ is close to the best values that we have discovered and therefore often close to an existing component. Such component will improve our approximation with high probability by allowing the mixture model to approximate the mode more accurately, but is not likely to discover a new mode. Estimating the initial reward based on a small initial

weight, in contrary, is more likely to place the component far from the current mixture model at locations where $R(\mathbf{x}_s)$ may be significantly worse than the best discovered values. Such component might converge to an irrelevant mode, that is, a local maximum of the target distribution that is still significantly worse than the best mode. The component will then get a very low weight, such that its effect on the approximation is negligible and the computational time (e.g., function evaluations) that was spent for improving this component was mainly wasted. If, however, such component discovers a new relevant mode, it will turn out much more valuable than a component that was added close to an existing mode.

In our experiments, we always add component with an initial weight of 1×10^{-29} which results in $\log q(o_n) \approx -66.77$. However, this value is quite arbitrary because adding a new component with initial weight of 1×10^{-300} would result in essentially the same mixture model and $\log q(o_n) \approx -690.78$. Hence, we do not estimate the initial reward based on the actual initial weight, but instead choose a value in place of $\log q(o_n)$ and vary it in the range of $[-1000, -50]$. By varying the (assumed) initial weight we can maintain exploration and avoid only adding components at irrelevant locations. Please refer to Appendix B.4 for a sensitivity analysis and for details on how the initial reward in Equation 3.22 is approximated.

Initializing the Covariance Matrix of New Components

The initialization of the covariance matrix of the new component is performed in two steps. In the first step, we decide on the initial entropy; in the second step, we decide on the initial correlations.

A possible option for choosing the initial entropy is to use the same entropy that was used when initializing the mixture at the beginning of the optimization, which would typically be relatively large in accordance with an uninformed prior. Such an initialization has the benefit of maintaining broad exploration during the whole optimization, and is not very sensitive to the initialization of the mean. However, initializing new components with high entropy can also be very wasteful as it will typically take a long time until they can contribute to the approximation. Furthermore, smaller initial entropies in combination with our heuristic for initializing the mean will result in a more directed exploration of promising regions. Hence, we initialize the new component with an entropy that is similar to those of the best components in the current model, namely we choose $\mathrm{H}_{init} = \sum_o q(o)\mathrm{H}(q(\mathbf{x}|o))$ as initial entropy. The entropy of the best components will typically decrease during optimization until it reaches a problem specific level. Hence, the exploration of new components will also become more local, without falling below a reasonable level.

For deciding on the correlations among the different dimensions, we can consider restarting the local search from scratch by choosing an isotropic covariance matrix $\boldsymbol{\Sigma}_{iso} = c_{iso}\boldsymbol{I}$, and making use of the existing components by averaging their covariance matrices, that is, $\boldsymbol{\Sigma}_{avg} = c_{avg}\sum_o p(o|\boldsymbol{\mu}_{new})\boldsymbol{\Sigma}_o$, where c_{iso} and c_{avg} are appropriately chosen to obtain the desired entropy H_{init} as shown in Appendix B.5. In VIPS we always averaged the covariance matrices, which can be sensible when adding components close to existing ones, or when similar correlations occur at different locations. However, we noticed that such initialization can impair exploration and, thus, degrade performance in one of our new experiments as shown in Section 3.4.2. As it is often difficult to predict, whether the curvature at the most responsible components is similar to the curvature at the new component, we perform a line search over a step size $\alpha \in [0, 1]$ to find the best interpolation

$$\boldsymbol{\Sigma}_\alpha = \alpha\boldsymbol{\Sigma}_{iso} + (1-\alpha)\boldsymbol{\Sigma}_{avg}$$

between both candidate covariance matrices with respect to the expected reward

$$R_{\text{new}}(\alpha) = \int_{\mathbf{x}} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{new}, \boldsymbol{\Sigma}_\alpha) \log \tilde{p}(\mathbf{x}) d\mathbf{x}.$$

The expected reward can be approximated using an importance weighted Monte Carlo estimate based on samples from the mixture

$$z(\mathbf{x}) = 0.5\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{new}, \boldsymbol{\Sigma}_{iso}) + 0.5\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{new}, \boldsymbol{\Sigma}_{avg}).$$

These samples and the respective function evaluations are also stored in the database $\mathcal{S}$ and can thus be reused during subsequent learning iterations.

Flow charts for the basic variant and the modified version are shown in Figure 3.1. An open-source implementation is available online[2]. In comparison to VIPS, VIPS++ makes better use of previous function evaluations and initializes new components based on a line search. Furthermore, VIPS++ uses fewer hyper-parameters by automatically adapting the bounds on the KL-divergences for the individual component updates and the regularization coefficients for fitting the reward surrogates. The number of hyper-parameters was further reduced by simplifications of the algorithms; namely, by performing an unconstrained optimization for the weight updates and by performing a single EM-like iteration on a given set of samples.

[2]The implementation can be found at `https://github.com/OlegArenz/VIPS`.

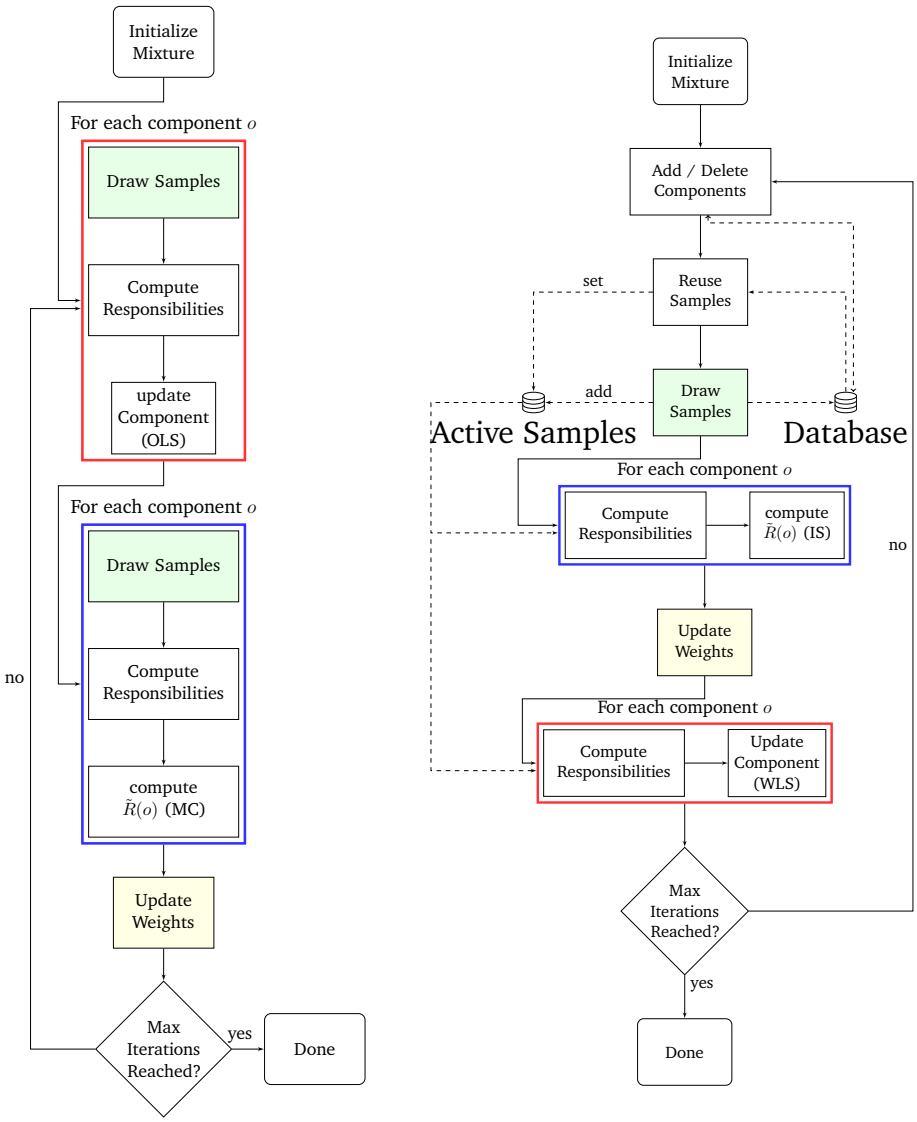


Figure 3.1.: We show flow charts for the basic variant (left) and VIPS++ (right). The basic variant updates the individual components by learning surrogates using ordinary least-squares (OLS) and uses Monte-Carlo (MC) for estimating the component's reward $\tilde{R}(o)$. VIPS++ adapts the number of component and uses the same set of samples for computing the components' reward using importance sampling (IS) and for updating the individual components using weighted least squares. The order of the weight and component updates has been swapped on the right flow chart to match the actual implementation.

3.3. Related Work

We will now discuss related work in the fields of variational inference, sampling and policy search.

3.3.1. Variational Inference

Traditionally, variational inference was applied for learning coarse approximations of high dimensional distributions, typically by assuming that the individual dimensions of the random variable are uncorrelated—the so-called mean-field assumption—and by choosing the variational distribution based on the target distribution. For example, Saul et al. [1996] approximated the hidden nodes of sigmoid belief networks with Bernoulli distributions, enabling them to maximize a lower bound on the ELBO in closed form. An iterative procedure was used for improving this lower bound. As such approach can only model unimodal distributions, it was later extended to mixtures of mean field distributions [Bishop et al., 1998; Jaakkola and Jordan, 1998].

However, relying on a variational distribution that can be fitted in closed form can be restrictive and the necessary derivations can be a major burden when applying such variational inference approaches to different models. Hence, Gershman et al. [2012] introduced non-parametric variational inference (NPVI), a black-box approach to variational inference that can be applied to any twice-differentiable target distribution. NPVI is restricted to GMMs with uniform weights and isotropic components that are iteratively optimized using first-order and second-order Taylor approximations. Although such variational approximation can in principle approximate any target distribution arbitrarily well, NPVI is in general not suited for learning highly accurate approximations with a reasonable number of components as shown in our comparisons.

Similar to VIPS, several black box approaches to variational inference rely on function evaluations of the target distributions that are chosen by sampling the variational approximation. Ranganath et al. [2014] apply the log-derivative trick, which is well-known in reinforcement learning [Williams, 1992], to variational inference in order to estimate the gradient of the ELBO with respect to the policy parameters. The gradient estimation does not require the gradient of the reward $\log \tilde{p}(\mathbf{x})$ but typically suffers from high variance. Ranganath et al. [2014], thus, suggest control variates and Rao-Blackwellization (for which they assume a mean-field approximation) for variance reduction. If the target distribution is differentiable and the variational approximation is reparameterizable, it is usually preferable to estimate the gradient with the reparameterization trick [Kingma and Welling, 2014; Rezende et al., 2014] which typically has much lower variance. Such approach can, for example, be used to train normalizing flows [Dinh et al., 2014]. Nor-

malizing flows are likelihood-based models that transform a simple distribution through one or several non-linear mappings. The probability density of the transformed distribution can be evaluated using the change-of-variables formula, which requires that the transformations are invertible and that the (log-)determinants of their Jacobians can be efficiently computed. Rezende and Mohamed [2015] proposed transformations that contract the density with respect to a learned hyperplane or to a point. The expressiveness of these planar and radial flows is rather limited and thus many flows has to be stacked to obtain rich approximations. However, several more expressive flows have been recently proposed [Dinh et al., 2016; Grathwohl et al., 2019; Huang et al., 2018; Kingma and Dhariwal, 2018; Kingma et al., 2016; Papamakarios et al., 2017]. Most of these flows make use of autoregressive transformations. For example, *inverse autoregressive flows* [IAF, Kingma et al., 2016] shift and scale each dimension of an input, x_i, by quantities that are computed based on the previous input dimensions $x_{j<i}$. As the resulting Jacobian matrices are triangular, the log determinants can be efficiently computed based on the diagonal elements. Rich approximations can be learned by stacking several such flows and shuffling the dimensions in-between based on fixed random or learned [Kingma and Dhariwal, 2018] permutations, which can also be seen as normalizing flows. In order to ensure the autoregressive property, IAFs use a technique that was previously used for autoregressive auto-encoders [Germain et al., 2015]. Namely, a mask is applied to a fully connected neural network in order to cut weights such that each output y_i is only connected to inputs x_j if $j < i$. Although such flows are invertible by construction, computing the inverse can be expensive because the different dimensions have to be inverted sequentially. Hence, evaluating the probability density of a sample that was produced by different distribution can be inefficient. *Masked autoregressive flows* [Papamakarios et al., 2017], thus, parameterize the inverse transformation (compared to IAFs) which makes them more efficient for density estimation at the cost of less efficient sampling. In general, normalizing flows are very popular nowadays, because they scale to high dimensions, allow for rich representations and are reparameterizable whenever the initial distribution is reparameterizable. However, we argue that such purely gradient-based optimization is not suited for learning accurate approximations of multimodal target distributions due to insufficient exploration. In our experiments, we compare against IAFs, which are well-suited for variational inference because we only need to evaluate the density of samples that were drawn from the normalizing flow.

Hessian-free stochastic Gaussian variational inference (HFSGVI, Fan et al. 2015) and TrustVI [Regier et al., 2017] can be used for learning Gaussian variational approximations. HFSGVI [Fan et al., 2015] learns GVAs with full covariance matrices using fast second order optimization. This idea has been extended by Regier et al. [2017] to trust region optimization. However, in difference to our approach, a euclidean trust region is used in

parameter space of the variational distribution. Such approach requires the computation of the Hessian of the objective which is only tractable for mean-field approximations of single Gaussian distributions. In contrast, we use the trust regions directly on the change of the distributions instead of the change of the parameters of the distribution. The information geometric trust regions in this paper allow for efficient estimation of GMMs with full covariance matrices without requiring gradient information from $\tilde{p}(\mathbf{x})$.

Information geometric trust regions and related methods such as certain proximal point methods as well as methods based on natural gradient descent have already been applied to variational inference. Salimans and Knowles [2013] derive a fixed point update of the natural parameters of a distribution from the exponential family that corresponds to a Monte-Carlo estimate of the gradient of Equation 3.1 preconditioned by the inverse of their empirical covariance. By making structural assumptions on the target distribution, they extend their method to mixture models and show its applicability to bivariate GMMs. Hoffman et al. [2013] consider mean-field variational inference and assume a certain structure on the target distribution. Namely, they consider models that consist of a product of conditionally independent distributions parameterized by local parameters that are correlated through global parameters. Furthermore, all distributions are assumed to belong to the exponential family and the distribution of the global parameters is assumed to be conjugate for computational reasons. They show that the natural gradient of the corresponding mean-field approximation can be efficiently computed, and approximated from mini-batches. Theis and Hoffman [2015] extended their approach by enforcing a trust-region based on the KL-divergence for better exploration. Khan et al. [2015] consider slightly more general models where optimizing the ELBO can be computationally expensive. They propose to apply the proximal point method by adding a penalty to the ELBO based on the reverse Kullback-Leibler divergence to the current iterate. They decompose the ELBO into easy and difficult parts and linearize the difficult parts. The derivations where extended by Khan et al. [2016] to other divergences and to stochastic gradients making the approach applicable to posterior approximations based on mini-batches. Altosaar et al. [2018] propose a slightly more general framework that can penalize derivations from a moving average instead of derivations from the last iterate, which can further help in avoiding bad local optima.

Several methods use the same hierarchical bound as VIPS in the broad context of variational inference. The first usage seems to date back to 2004, where Agakov and Barber [2004] proposed the bound for learning an optimal weighting between several mean-field approximations. Ranganath et al. [2016] proposed Hierarchical variational methods (HVM) where the lower-level distributions $q(\mathbf{x}|\mathbf{o})$ where again mean-field distributions. In their setting, the latent variable $\mathbf{o}$ corresponds to a parameter vector that fully specifies the mean-field distribution. They learned complex priors $q(\mathbf{o})$ over these parameters,

namely GMMs and normalizing flows, in order to allow for rich variational approximations. However, in contrast to the responsibilities in VIPS the conditional $q(\mathbf{o}|\mathbf{x})$ is not tractable and thus has to be approximated and learned along the variational distribution. Our EM-inspired approach based on exact tightening of the hierarchical lower bound would thus not be applicable in their setting. Although Ranganath et al. [2016] learned Gaussian mixture models to model the upper-level distribution $q(\mathbf{o})$, they did not apply the hierarchical bound for this, but optimized the parameters directly using stochastic gradient descent. As we will show in our experiments, such black-box approach is not suited for learning variational GMM approximations. Tran et al. [2016] consider a similar setup for their *variational Gaussian process*. For the mean-field factors $p(x_n|o_n)$ of their lower-level components they consider degenerated point masses specified by their scalar parameter value o_n. As Ranganath et al. [2016], they optimize the hierarchical lower bound with respect to the prior distribution $q(\mathbf{o})$ and the conditional $q(\mathbf{o}|\mathbf{x})$. Their main contribution is the representation of the prior distribution. Each parameter value o_n is sampled by evaluating a Gaussian process [GP, Rasmussen and Williams, 2006] on an input that was sampled from a fixed distribution. The parameters of the prior are given by the kernel hyper-parameters of the GP as well as the *variational data* that is interpolated by the GP. Whereas all these methods only consider mean-field distributions for the lower-level components that are fully specified by the latent variable, Maaløe et al. [2016] represent them using inference networks, that is, neural networks that take a data point as input and output the parameters of a (typically diagonal) Gaussian distribution. They consider variational autoencoders VAE and aim to learn more expressive approximations of the latent code z. They also introduce an additional latent variable representing class labels in order to train a classifier end-to-end while optimizing the variational autoencoder in semi-supervised fashion. In contrast to these previous applications of the hierarchical lower bound, VIPS shows that it can also be used to learn accurate variational approximations without having to approximate the inverse model $p(o|\mathbf{x})$. This enables us to optimize the ELBO by alternately maximizing and (exactly) tightening the hierarchical lower bound.

Closely related to our work are two recent approaches for variational inference that concurrently explored the idea of applying boosting to make the training of GMM approximations tractable [Guo et al., 2016; Miller et al., 2017]. These methods start by minimizing the ELBO objective for a single component and then successively add and optimize new components and learn an optimal weighting between the previous mixture and the newly added component. However, because these methods can not adapt previously added components or their relative weighting, they can require an unnecessary large number of components to learn accurate approximations. Furthermore, they do not use information-geometric trust regions to efficiently explore the sample space and therefore have problems finding all the modes as well as accurate estimates of the covariance

matrices. GMMs are also used by Zobay [2014] where an approximation of the GMM entropy is used to make the optimization tractable. The optimization is gradient-based and does not consider exploration of the sample space. It is therefore limited to rather low dimensional problems.

The work of Weber et al. [2015] already explored the use of reinforcement learning for VI by formalizing VI as sequential decision problem. However, only simple policy gradient methods have been proposed in this context which are unsuitable for learning GMMs.

3.3.2. Sampling

Although MCMC samplers can not directly be used for approximating distributions, they are for many applications the main alternative to VI. Especially, when applying VIPS as a model-based sampler, that is, if we do not have direct interest in learning a GMM approximation, it should be compared to other zero-order sampling methods that do not need gradient information from the target density. The most prominent methods to use here are MCMC methods such as slice sampling [Neal, 2003], elliptical slice sampling [Murray et al., 2010] or generalized elliptical slice sampling [Nishihara et al., 2014]. MCMC methods define a Markov chain for the sampling process, that is, the current sample defines the state of the chain, and we define a conditional distribution how to generate new samples from the current state.

Slice sampling introduces an auxiliary variable y to define this conditional distribution. The variable y is always sampled between 0 and the unnormalized target density of the current sample. The random variable $\mathbf{x}$ is only accepted if the new target density is larger than y. In case of rejection, the area where a new $\mathbf{x}$ sample is generated is reduced to limit the number of rejections. However, the sampling process is still very inefficient for higher dimensional random variables. Elliptical slice sampling [Murray et al., 2010] is a special case of slice sampling and defines the slice by an ellipse defined by the current state $\mathbf{x}$ and a random sample from a Gaussian prior (with origin 0). Such ellipse allows for more efficient sampling and rejection in high dimensional spaces but relies on a strong Gaussian prior.

If the gradient of the target distribution is available, Hamiltonian MCMC [Duane et al., 1987] and the Metropolis-adjusted Langevin algorithm [Roberts and Stramer, 2002] are also popular choices. The No-U-Turn sampler (NUTS) [Hoffman and Gelman, 2014] is a notable variant of Hamiltonian MCMC that is appealing for not requiring hyper-parameter tuning.

While many of these MCMC methods have problems with multimodal distributions in terms of mixing time, other methods use multiple chains and can therefore better explore multimodal sample spaces [Calderhead, 2014; Earl and Deem, 2005; Neal, 1996;

Nishihara et al., 2014]. Parallel tempering MCMC [Earl and Deem, 2005] runs multiple chains, where each chain samples the target distribution at a different temperature. Each step consists either of updating each chain independently, or swapping the state between two neighboring chains which allows for more efficient mixing between isolated modes. However, because only one chain samples the target distribution at the correct temperature, PTMCMC can be inefficient if the number of chains and their respective temperatures are not adequately tuned for the sampling problem. Generalized elliptical slice sampling [Nishihara et al., 2014] uses multiple Markov chains simultaneously using massive parallel computing. The current state of the Markov chains is used to learn a more efficient proposal distribution, where either Student-t distributions or Gaussian mixture models can be used. Yet, learning such distributions in high dimensional spaces using maximum likelihood is prone to overfitting and the GMM approach has not been evaluated on practical examples. Moreover, the approach requires a massive amount of sample evaluations. In this paper, we want to minimize the amount of sample evaluations.

Rainforth et al. [2018] explicitly consider the exploration-exploitation trade-off. They use a method similar to Monte-Carlo tree search [Coulom, 2006] to build a tree for partitioning the search space. By covering regions where the target distribution has high density more finely, the resulting *inference trees* (IT) are well-suited for inference on multimodal distributions, for example, in combination with sequential Monte-Carlo [Doucet et al., 2001].

Stein variational gradient descent (SVGD) [Liu and Wang, 2016] is a sampling method that closely relates to variational inference. However, instead of optimizing the parameters of a model, SVGD directly optimizes an initial set of particles. By framing sampling as optimization problem, SVGD inherits the computational advantages of variational inference and because it is non-parametric, it is capable of approximating multimodal distributions. However, this method requires to construct the Gram matrix of the particles and is thus not suitable for drawing large number of samples. Furthermore, defining appropriate kernels can be challenging for high-dimensional problems.

3.3.3. Policy Search

Our algorithm shares a lot of ideas with information-geometric policy search algorithms such as REPS [Peters et al., 2010], HiREPS [Daniel et al., 2016] and MORE [Abdolmaleki et al., 2015]. In difference to policy search, where we want to maximize an average reward objective, we want to minimize the KL-divergence to a target distribution. REPS introduces the first time information-geometric policy updates, while the MORE algorithm introduces closed form updates for single Gaussians using compatible function approximation and additional entropy regularization terms that yields an optimization problem similar to KL

minimization.

The HiREPS [Daniel et al., 2016] and LaDiPS [End et al., 2017] algorithms extended the REPS and MORE ideas to mixture distributions such that multiple modes can be represented. However, the used updates were based on approximations or heuristics and can not optimize the entropy of the complete mixture model.

3.4. Experiments

In this section we will evaluate VIPS++ with respect to the quality of the learned approximation and relate it to a variety of state-of-the-art methods in variational inference and Markov chain Monte Carlo. We start with a description of the considered sampling problems in Section 3.4.1. The effects of the most important hyper-parameters and algorithmic choices are examined in Section 3.4.2. Section 3.4.3 contains an illustrative experiment to show how VIPS++ approximates a two-dimensional, multimodal target distribution by starting with a single component and iteratively adding more components according to our heuristic. The selected methods for our comparisons, and the selection of their hyper-parameters are discussed in Section 3.4.4 and Section 3.4.5. The results of the quantitative experiments are presented and discussed in Section 3.4.6.

3.4.1. Sampling Problems

We will evaluate VIPS++ on typical sampling problems such as Bayesian logistic regression, Bayesian Gaussian process regression and posterior sampling of a multi-level Poisson generalized linear model. We further approximate the posterior distribution over the parameters of a system of ordinary differential equations known as the Goodwin model, which can be used for modeling oscillating gene-protein interaction. As these problems tend to have concentrated modes, we devised several more challenging problems that require careful exploration of the sampling space. Namely, we consider sampling from unknown GMMs with distant modes and sampling the joint configurations of a planar robot such that it reaches given goal positions.

Bayesian Logistic Regression

We perform two experiments for binary classification that have been taken from Nishihara et al. [2014] using the *German credit* and *breast cancer* data sets [Lichman, 2013]. The *German credit* data set has twenty-five parameters and 1000 data points, whereas the *breast cancer* data set is thirty-one dimensional and contains 569 data points. We standardize

both data sets and perform linear logistic regression where we put zero-mean Gaussian priors with variance 100 on all parameters.

Multi-Level Poisson GLM

We also took an experiment from the related work VBOOST [Miller et al., 2017]. For this experiment we want to sample the posterior of a hierarchical Poisson GLM on the 37-dimensional *stop-and-frisk* data set, where we refer to Miller et al. [2017] for the description of the hierarchical model.

GP Regression

We perform Bayesian Gaussian process regression on the ionosphere data set [Lichman, 2013] as described by Nishihara et al. [2014]. Namely, we use 100 data points and want to sample the hyper-parameters of a squared exponential kernel where we put a gamma prior with shape 1 and rate 0.1 on the 34 length-scale hyper-parameters. We initialize VIPS with a single Gaussian component, $\mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{I})$ and sample in log-space to ensure positive values for the hyper-parameters.

Goodwin Model

Similar to Calderhead and Girolami [2009], we want to sample the posterior over the parameters of a Goodwin oscillator [Goodwin, 1965] based on noisy observations. The Goodwin oscillator is a system of nonlinear ordinary differential equations (ODE) that models the oscillatory behavior between protein expression and mRNA transcription in enzymatic control processes. We consider a Goodwin oscillator with ten unknown parameters and put a Gamma prior with shape 2 and rate 1 on each of these. The likelihood of 41 observations is computed by numerically integrating the ODE and assuming Gaussian observation noise with zero mean and variance $\sigma^2 = 0.2$. Please refer to Appendix B.6 for more details on the ODE and the experimental setup.

Gaussian Mixture Model

In order to evaluate how VIPS++ can explore and approximate multimodal probability distributions with distant modes, we consider the problem of approximating an unknown GMM comprising 10 components. We consider different number of dimensions, namely $D = 20$, $D = 40$ and $D = 60$. For each component, we draw each dimension of the mean uniformly in the interval $[-50, 50]$. The covariance matrices are given by $\mathbf{\Sigma} = \mathbf{A}^\top \mathbf{A} + \mathbf{I}_D$

where each entry of the $D \times D$-dimensional matrix $\mathbf{A}$ is sampled from a normal distribution with mean 0 and standard deviation $0.1D$. Note that each component of the target distribution can have a highly correlated covariance matrix, which is even a problem for the tested MCMC methods.

Planar Robot

In order to test VIPS++ on a multimodal problem with non-Gaussian modes we devised a challenging toy task where we want to sample the joint configurations of a planar robot with 10 links of length 1 such that it reaches desired goal positions. The robot base is at position $(0, 0)$ and the joint configuration describes the angles of the links in radian. In order to induce smooth configurations, we put a zero mean Gaussian prior on the joint configurations where we use a variance of 1 for the first joint and a variance of 4×10^{-2} for the remaining joints. Deviations from the nearest goal position are penalized based on a likelihood that is given by a Gaussian distribution in the Cartesian end-effector space, with a variance of 1×10^{-4} in both directions. We consider two experiments that differ in the number of goal positions. For the first experiment, we want to reach a single goal-position at position $x = 7$ and $y = 0$. For the second experiment we want to reach four goal positions at positions $(7, 0)$, $(0, 7)$, $(-7, 0)$ and $(0, -7)$. Please refer to Appendix B.7 for details on how the target distribution is computed.

Ground-truth samples for both experiments are shown in Figure 3.2. Each goal position can be reached from two different sides, either up and down, or left and right. Other configurations that would reach the goal position, for example some zig-zag configurations, are not relevant due to the smoothness prior and can create poor local optima. Although there are only two relevant ways for reaching each goal position, closely approximating these modes can require many mixture components, because the small variance of the Cartesian likelihood term enforces components with small variance. We therefore also evaluate slightly different hyper-parameters for VIPS++, where we add a new component at every iteration.

3.4.2. Ablations

In this subsection we will evaluate the effects of some algorithmic choices. Namely, we will show that adapting the number of components can be crucial for discovering relevant modes of multimodal target distributions, that the previously proposed initializing of covariance matrices can have detrimental effects, and that the sample reusage of VIPS++ can significantly increase sample efficiency.

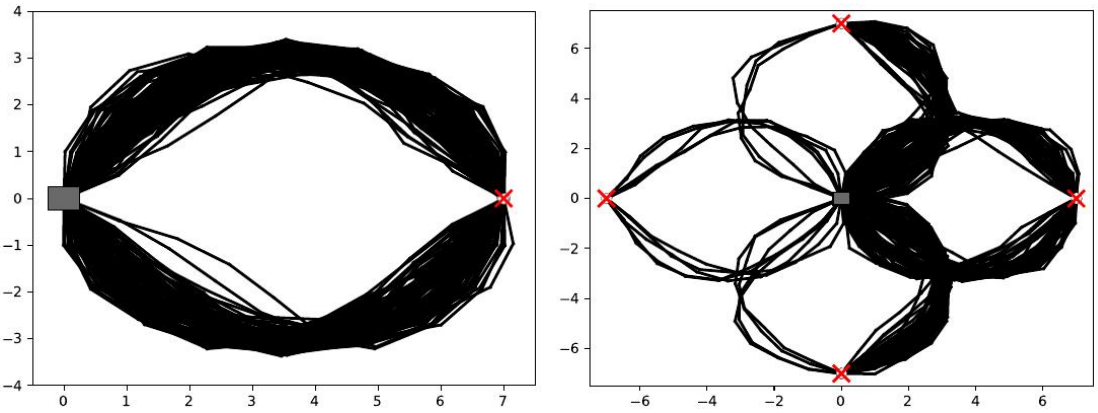


Figure 3.2.: The plots show 200 ground-truth samples for both *planar robot* experiments that have been generated using generalized elliptical slice sampling. The base of the planar robot is shown as a gray box and the end-effector positions are shown as circles.

Adapting the number of components

As discussed in Section 3.2.4, VIPS automatically adapts the number of components during learning for better exploration, which enables it to improve on local optimal solutions. We evaluate the effect of this adaptation by comparing VIPS++ with a variant that keeps the number of components fixed on the *breast cancer* experiment and the 20-dimensional *GMM* experiment. We initialize the non-adaptive variant with different numbers of initial components, where each mean is drawn from an isotropic Gaussian $\mathcal{N}(\mathbf{0}, \alpha \boldsymbol{I})$. We use $\alpha = 100$ for the *breast cancer* experiment and $\alpha = 1000$ for the *GMM* experiments. For VIPS++ we start with a single component with mean $\mathbf{0}$. All covariance matrices are initialized as $\Sigma = \alpha \boldsymbol{I}$. The achieved MMDs are shown in Figure 3.3. The non-adaptive variant converges to better approximations when increasing the number of components on the *breast cancer* experiment. However, the required number of function evaluations until convergence scales approximately linearly with the number of components. VIPS++ can learn good approximations with few function evaluations and further improves while increasing the size of the mixture model. On the *GMM* experiment, all tested variants would in principle be able to model the target distribution exactly. However, depending on the initialization, several components may converge to the same mode which results in bad local optima. We therefore needed at least 25 initial components for occasionally learning good approximations during this experiment and even when initializing with 100

components the non-adaptive variant would sometimes fail to discover all true modes. In contrast, by adaptively adding new components at interesting regions VIPS++ reliably discovers all ten modes. Please refer to Appendix B.8 for a plot of the average number of components that are learned by VIPS++ for all experiments in the test bed.

Initializing the covariance matrices

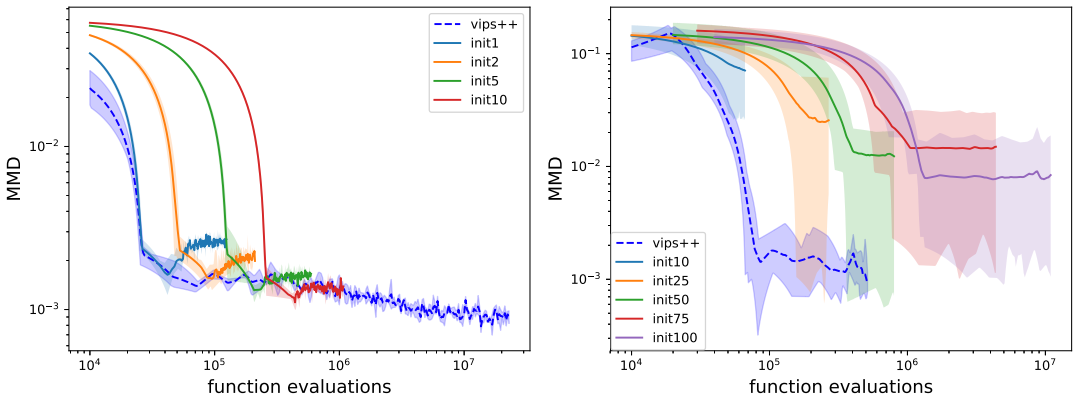
We also evaluate the different strategies for initializing the covariance matrix of a newly added component, which were discussed in Section 3.2.4. We compare the proposed line search used by VIPS++ with the interpolation used by VIPS as well as an isotropic initialization. Figure 3.4 compares the different strategies on the *Goodwin* experiment and the *planar robot* experiment (with four goal positions). The *planar robot* experiment shows, that interpolating based on the responsibilities can seriously impair the performance on multimodal problems. We believe that interpolating based on the responsibilities can lead to highly anisotropic initial covariance matrices that do not sufficiently explore along relevant directions which would explain the detrimental effects. Although we could not show a benefit of the line search compared to the isotropic initialization, we opted for the line search for the quantitative experiments, because it seems sensible and did not perform significantly worse in our experiments.

Sample Reusage

Compared to VIPS, VIPS++ uses a more sophisticated method for reusing samples from previous iteration—as detailed in Section 3.2.2 and 3.2.3—by identifying relevant samples among all previous function evaluations and by controlling the number of new samples from each component based on its number of effective samples. We compare the new sample strategy with the previously employed method of always using the samples of the three most recent iterations. Figure 3.5 evaluates the different strategies on the *Goodwin* experiment and the 20-dimensional *GMM* experiment. The proposed strategy of VIPS++ significantly outperforms the previous method by reducing the sample complexity by approximately one order of magnitude.

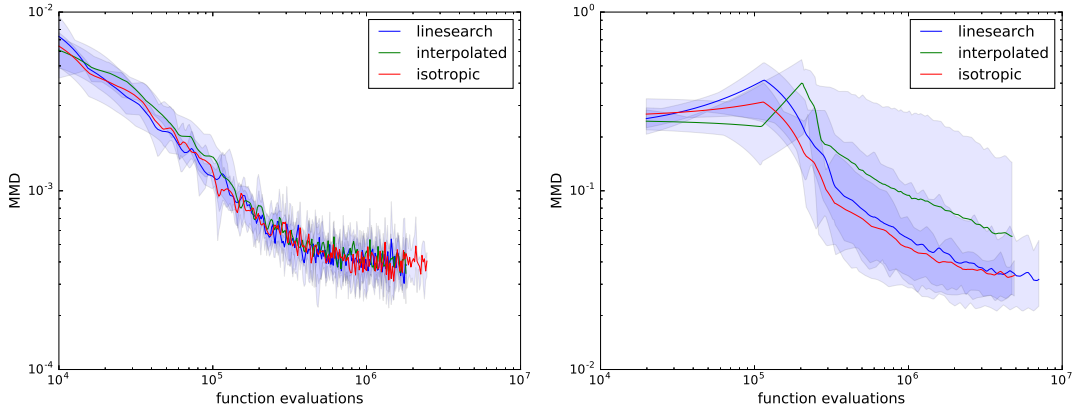
3.4.3. Illustrative Experiment

We start with a qualitative two-dimensional experiment to illustrate the sample reusage and the adaptation of the number of components. The target distribution is given by a Gaussian mixture model with ten components similar to the higher-dimensional GMM experiments. We use the same hyper-parameters as in the remaining experiments and



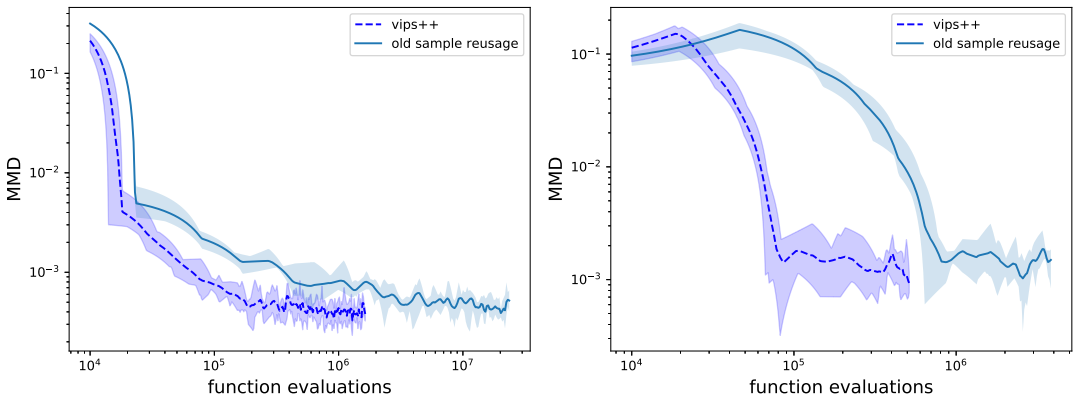
(a) breast cancer (b) GMM

Figure 3.3.: We compare VIPS++ with a variant that does not add or delete components. On the *breast cancer* experiment, VIPS++ converges to a good approximation as fast as the variant that learns a single component, but it refines the approximation by adding more components. When not adapting the number of components on the *GMM* experiment, the quality of the approximation strongly depends on the initialization and even 100 initial components would sometimes fail to detect all modes.



(a) Goodwin model (b) planar robot

Figure 3.4.: We compare different strategies for initializing the covariance matrices of newly added components. Interpolating the covariance matrices of the current model based on the responsibilities can have detrimental effects as shown in the planar robot experiment.



(a) Goodwin model (b) 20-dimensional GMM

Figure 3.5.: The sample reusage of VIPS++ is approximately one order of magnitude more efficient than the sample reusage of VIPS.

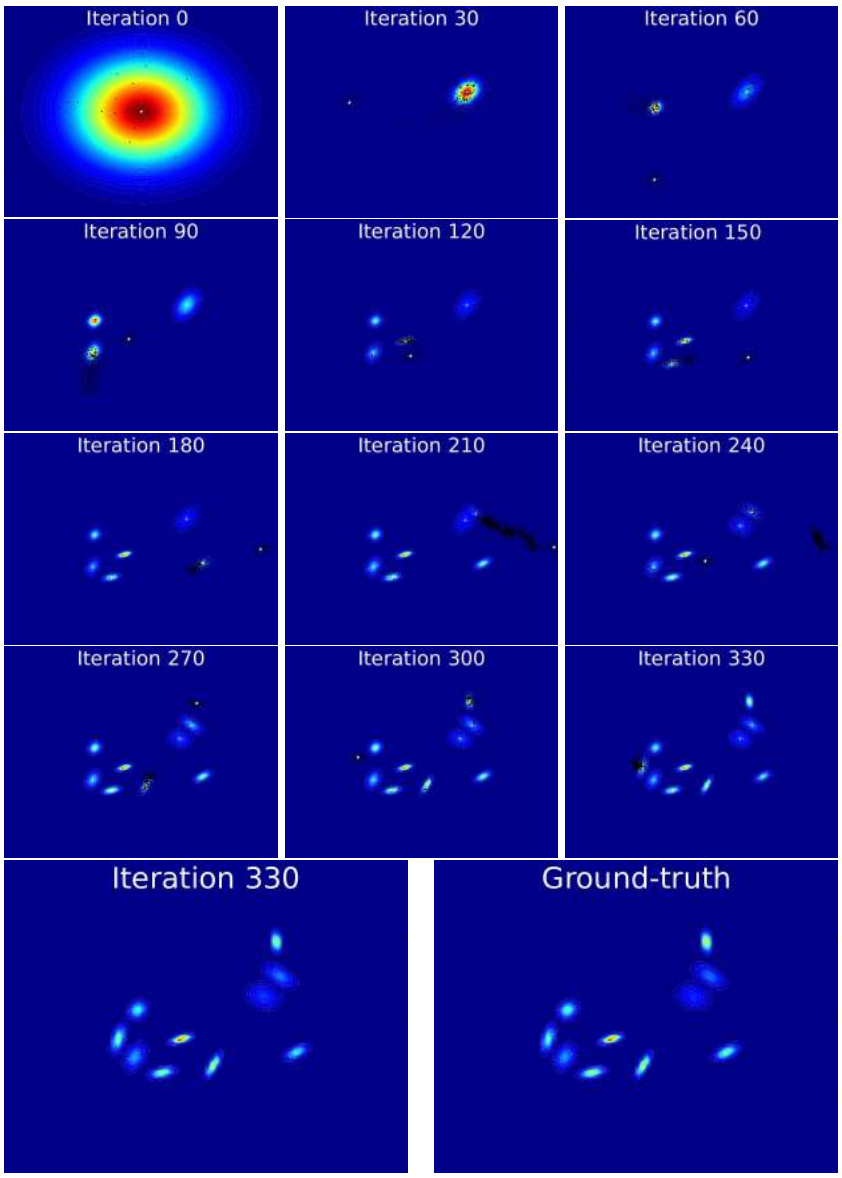


Figure 3.6.: The first 12 plots show the learned approximation for the illustrative experiment every 30 iterations, directly after adding a new component. The means of the Gaussian mixture model are indicated with a white plus except for the newest component which is marked by a star. Black dots indicate all samples that have been drawn except for those that have already been shown at previous plots. The last two plots compare the learned approximation and the target distribution.

start with a single component. Figure 3.6 shows the target distribution as well as the learned approximation directly after adding each new component. The new components are often added close to missing modes and components are typically not sampled after they have converged. The learned model closely approximates the target distribution.

3.4.4. Considered Competitors

We compare VIPS++ to the closely related methods variational boosting [VBOOST, Miller et al., 2017] and non-parametric variational inference [NPVI, Gershman et al., 2012] as well as state-of-the-art methods in variational inference and MCMC, namely inverse autoregressive flows [IAF, Kingma et al., 2016], Stein variational gradient descent [SVGD, Liu and Wang, 2016], Hamiltonian Monte Carlo [HMC, Duane et al., 1987], elliptical slice sampling [ESS, Murray et al., 2010], parallel tempering MCMC [PTMCMC, Earl and Deem, 2005] and slice sampling [Neal, 2003]. We also compare to naive gradient based optimization of a Gaussian mixture model (with fixed but tuned number of components). As GMMs are not exactly reparameterizable, we compute their stochastic gradients using black-box variational inference (BBVI). Please refer to Appendix B.9 for details on the specific implementations. Due to the high computational demands, we do not compare to every method on each experiment but rather select promising candidates based on the sampling problem or on the preliminary experiments that we had to conduct for hyper-parameter tuning. We present our justification for each omitted experiment in Appendix B.10, where we also present a table that shows the competitors we compared against on each test problem.

Instead of using a variant of MORE (which we denote as VIPS1), it would also be possible to update the individual components using the reparameterization trick [Kingma and Welling, 2014; Rezende et al., 2014]—which assumes that the target distribution is differentiable—or black-box variational inference [Ranganath et al., 2014]. We evaluated these options by comparing VIPS1, black-box variational inference and the reparameterization trick for learning Gaussian variational approximations on the *breast cancer* experiment and the *planar robot* experiment. The results are presented in Appendix B.11 and show that VIPS1 is not only more efficient than black-box variational inference, but also one to two orders of magnitude more efficient than the reparameterization trick.

3.4.5. Hyper-Parameters

For the competing methods, we tuned the hyper-parameters independently for each test problem. We typically tuned the hyper-parameters based on our test metric, the maximum mean discrepancy (MMD). However, in all our experiments black-box variational inference

and inverse autoregressive flows collapsed to single modes on multimodal test problems which increased the MMD. In these cases, we tuned the hyper-parameters with respect to the ELBO, rather than setting the learning rate to zero which would perform better on our test metric. For VIPS++, we use the same set of hyper-parameters on all experiments. However, for the planar robot experiment which can profit from large GMMs with several hundred components, we add a new component at every iteration. Learning such large mixture models for simpler, unimodal problems would be wasteful, and we thus use a slower adding rate $n_{\text{add}} = 30$ for the remaining experiments. The remaining hyper-parameters are shown in Appendix B.12.

3.4.6. Results

We compare the different methods in terms of efficiency, regarding both, the number of function evaluations and wall clock time, and in terms of sample quality which we assess by computing the maximum mean discrepancy [MMD, Gretton et al., 2012] with respect to ground-truth samples. The MMD is a nonparametric divergence between mean embeddings in a reproducible kernel Hilbert space. Please refer to Appendix B.13 on how the MMD and the ground-truth samples are computed.

Figure 3.7 shows plots of the MMD over the number of function evaluations for the different sampling problems in the test bed. We perform five runs for each method and linearly interpolate the MMD values to produce continuous curves. The plots show the mean of these curves, as well as the smallest and largest value as shaded area. The tested methods are apparent from the legends. VBOOST can make use of low-rank approximations for learning the covariance matrices and we indicate the chosen ranks in the legends. The *German credit*, *breast cancer*, *stop-and-frisk* and the 20-dimensional *GMM* experiment, as well as the *planar robot* experiment with a single goal position were also used in our previous work [Arenz et al., 2018] and we use some of the previous results. For example, we directly compare VIPS++ with the previously published results of VIPS. Unlike VIPS++, VIPS bounds the maximum number of components by stopping to add new components if the current number of components matches a given threshold. This threshold is indicated in the respective legends. Figure 3.8 presents the results with respect to computational time for the *ionosphere* and *Goodwin model* experiment as well as the *planar robot* experiment with a single goal position. As the results are similar compared to the evaluations with respect to the number of function evaluations, we show the remaining plots in Appendix B.14.

Furthermore, for our comparisons with the variational inference methods BBVI and IAF we also present learning curves regarding the ELBO in Appendix B.15.

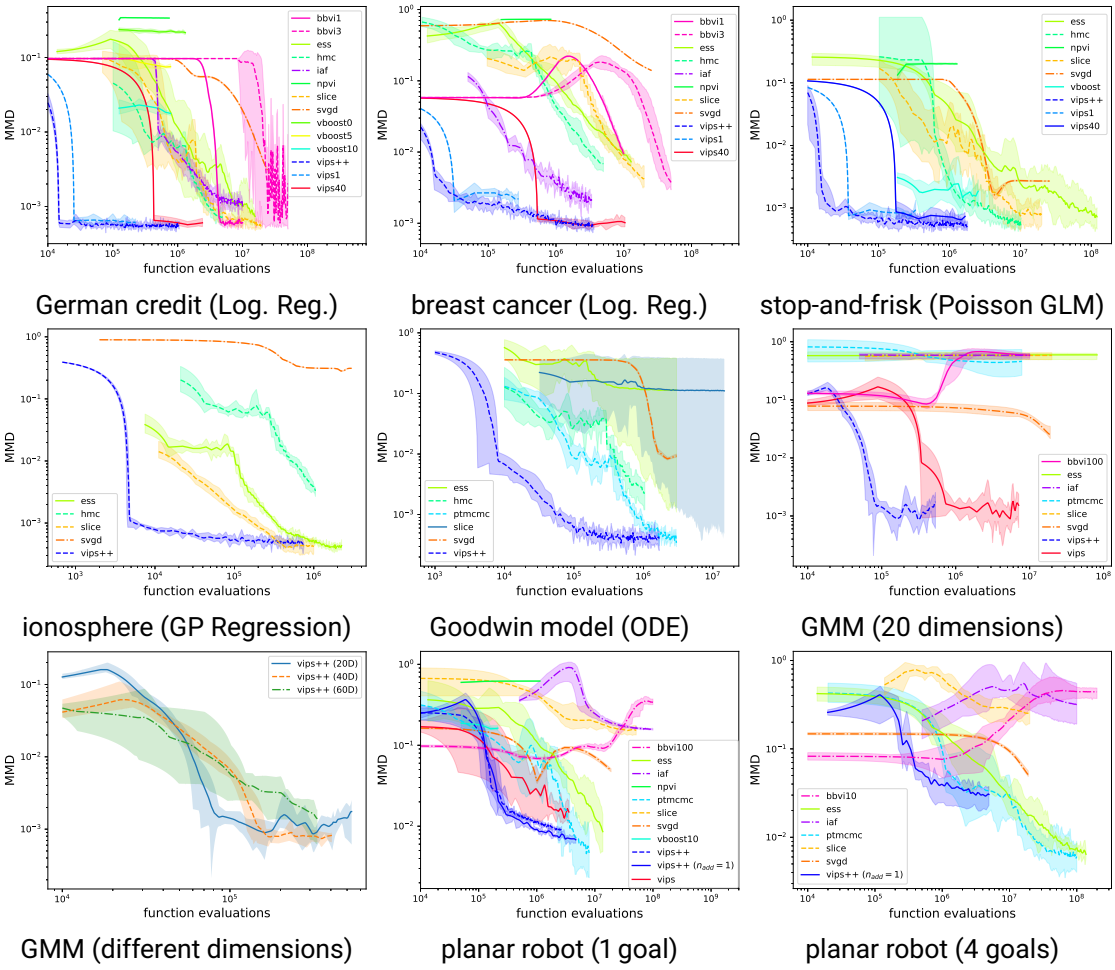


Figure 3.7.: The maximum mean discrepancy with respect to ground-truth samples is plotted over the number of function evaluations on log-log plots for the different sampling problems in the test bed. VIPS++ achieves in most cases a sample quality that is on par with the best MCMC sampler while requiring up to three orders of magnitude fewer function evaluations.

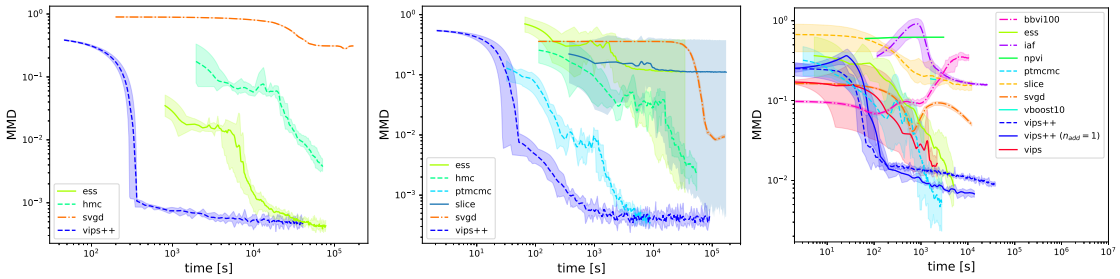


Figure 3.8.: Evaluating the methods with respect to computational time yields comparable results as evaluating with respect to the number of function evaluations. These results show that VIPS++ can also be competitive to MCMC in terms of computational time.

Discussion

The sample quality achieved by VIPS is unmatched by any variational inference method on all considered experiments and in most cases on par with the best MCMC sampler. VIPS requires significantly fewer function evaluations and computational resources for producing such high quality samples. VIPS++ is approximately one order of magnitude more efficient than VIPS and two to three orders of magnitude more efficient than the remaining methods. VIPS and VIPS++ were also the only methods that could produce good results on the 20-dimensional *GMM* experiment, where they were able to reliably discover and approximate all ten modes of the target distribution. We therefore only evaluated VIPS++ on the higher-dimensional *GMM* experiments where it also approximated the target distribution with high accuracy. However, on the *planar robot* experiment with four goal positions ESS and PTMCMC could produce significantly better samples than VIPS++. We believe that learning highly accurate GMM approximations would require a very large number of components for this experiment. Already on the *planar robot* experiment with a single goal position, we could slightly improve the learned approximations by adding new components more frequently. Compared to the default adding rate, which learned GMMs with approximately 150 components, the faster adding rate resulted in GMMs with approximately 350 components. We believe that VIPS++ would require significantly more components to achieve comparable sample quality to the MCMC samplers on the more challenging *planar robot* experiment. However, learning very large mixture models can become infeasible, because computing the (log-)responsibilities $\log q(o|\mathbf{x})$ exactly can become prohibitive. Figure 3.9 visualizes the weights and means of the learned approxi-

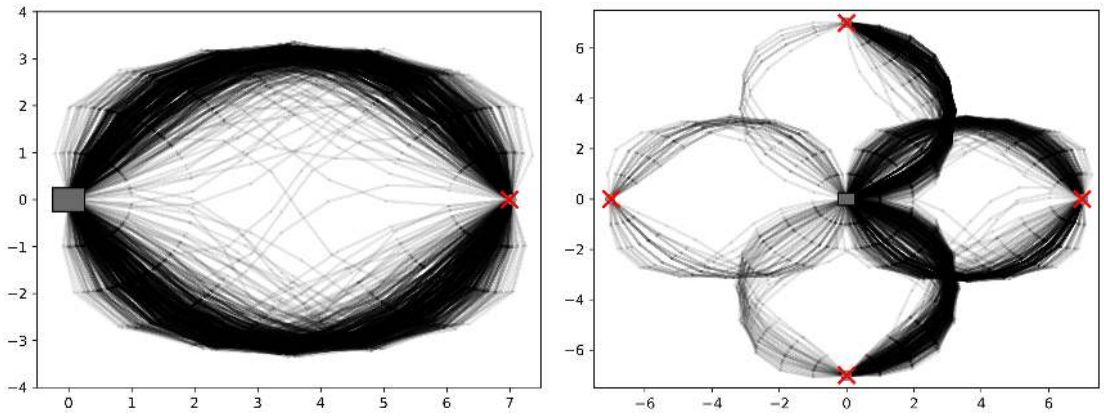


Figure 3.9.: The plots visualize the weights and means of the mixture models learned by VIPS++ for each of the planar robot experiments when adding new components with adding rate $n_{\text{add}} = 1$. The gray box indicates the base of the robot; the red crosses indicate the goal positions. Components with larger weight are drawn darker. The visualized mixture models comprise 333 and 360 components for the experiments with one goal position (left) and four goal positions (right), respectively.

mation of the first run for both *planar robot* experiments when adding new components at every iteration. We can see that the learned components are still of very good quality. Samples from the learned models are shown in Appendix B.16 and compared to those obtained by BBVI, IAF, PTMCMC.

3.5. Conclusion and Future Work

We proposed VIPS++, a method for learning GMM approximations of intractable probability distributions that exploits the connection between variational inference and policy search. We introduced a variant of MORE [Abdolmaleki et al., 2015] that can be efficiently used for learning Gaussian variational approximations. We further derived a lower bound on the I-projection to latent variable models that can be used for learning a local optimum of the true objective, similar to expectation-maximization. By applying this decomposition to Gaussian mixture models, the I-projection can be performed independently for each component, allowing us to improve the GMM approximation by independently updating the components using our variant of MORE. We argue that a good trade-off between exploration and exploitation is essential for efficiently learning accurate multimodal approximations. We tackle the exploration-exploitation dilemma locally for each component by updating them using information-geometric trust regions. For global exploration, we dynamically add new components at interesting regions.

For target distributions that can be well approximated with few components, VIPS does not only outperform existing methods for variational inference, but is also several orders of magnitude more efficient than Markov chain Monte Carlo at drawing samples. We also showed that VIPS can learn large mixture models comprising several hundred components. However, learning very large GMMs is computationally expensive and MCMC methods can be more efficient at drawing samples.

Learning Gaussian components with full covariance matrices can become intractable for high dimensional problems, and we thus applied VIPS only for medium-scaled problems with up to 60 dimensions. For significantly higher-dimensional problems, learning low-rank approximations and using gradient information for the component updates are interesting routes of future work. It is also interesting to further investigate the strong ties between variational inference and policy search. Using our decomposition we can learn GMMs of policy parameters for the black-box reinforcement learning setting where time-series data is not assumed and exploited. In order to apply VIPS for multimodal reinforcement learning with time-series data, we aim to contextualize the GMM parameterization on the state of an MDP to directly learn GMM policies. Furthermore, it is interesting to investigate how our decomposition can be applied to different problems such as clustering or density

estimation, or to other latent variable models.

3.6. Follow-Ups

In this section, we want to briefly mention two follow-up works that fit well to the scope of this thesis. Namely, we will discuss the work by Ewerton, Arenz, and Peters [2020] where we applied VIPS for robotic teleoperation and the work by Becker, Arenz, and Neumann [2020] where we applied the lower bound decomposition of VIPS to the problem of density estimation, which—although interesting by itself—can be regarded as an intermediate step to applying the decomposition to imitation learning, which will be the focus of Chapter 4.

3.6.1. Assisted Teleoperation in Changing Environments with a Mixture of Virtual Guides

In contrast to humans, robots can be safely deployed to hazardous environments and may possess greater strength or precision. On the other hand, they lack the capability of high-level reasoning that is often necessary to autonomously perform a desired task in unstructured environments, for example, when segregating nuclear waste [Abi-Farraj et al., 2019]. Teleoperation allows us to combine these advantages of humans and robots. Instead of using standard joysticks for controlling the robot, it is common to use special haptic devices [for example, as used by Ewerton et al., 2019]—which themselves can be characterized as a robot—or even standard collaborative robots [Singh et al., 2020]. Such device (the leader) can be used to directly control several degrees of freedom (DoFs) of the teleoperated robot (the follower)—for example, six DoFs, when controlling the end-effector position and orientation. Furthermore, they can provide haptic feedback or cues to the operator. An example for such haptic device, the Virtuose Tao 6D, is illustrated in Figure 3.10. When learning a controller that applies wrenches to the handle of the leader in order to assist the operator during teleoperation, we obtain a shared control setting, where both, the operator and the learned controller, control the follower by overlaying wrenches at the leader's handle. Compared to an unassisted teleoperation setup, such setup has the advantage that it can reduce stress and cognitive load of the operator, for example, by guiding the movement around obstacles, while ensuring that the operator remains in control of the system (when the wrenches of the controller are sufficiently limited). However, learning a suitable controller faces several challenges.

- We need to ensure that the guiding wrenches change smoothly (in time and space) because sudden changes could catch the operator off guard.

Figure 3.10.: An operator controls a robot (not shown) using the Virtuose Tao 6D haptic device. Image was taken from [Ewerton et al., 2020].

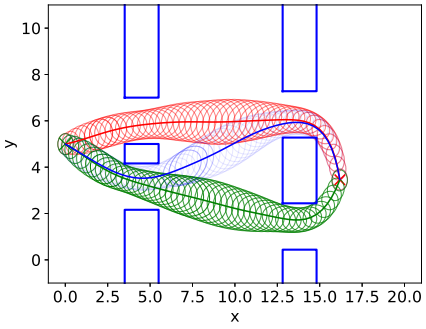


Figure 3.11.: We learn a mixture of three ProMPs for guiding a point mass to a goal position (marked by a red cross). Each ProMP corresponds to a trajectory with variable stiffness. Image was taken from [Ewerton et al., 2020].

- Often, several (movement-)plans are possible, and we can not know beforehand which one is preferred by the operator. Hence, we want to discover different solutions and potentially re-plan online if the operator does not seem to follow any of the learned plans.
- The desired strength of the guiding wrench may vary during a given plan. For example, when grasping a small object we need high precision and thus strong guiding wrenches, whereas we might want to provide little feedback while approaching the object freely in space.
- The cause of the applied wrenches—that is the inferred plan—should be comprehensible to the operator in order to enable them to coordinate with the learned controller.

We address these challenges by providing wrenches based on a mixture of probabilistic movement primitives (ProMPs, Paraschos et al. 2013). A ProMP corresponds to a Gaussian distribution over trajectories that is represented as a Gaussian distribution over trajectory parameters. A single ProMP, thus, encodes a plan (given by the mean trajectory) with variable stiffness (based on the covariance matrix). Furthermore, a Gaussian mixture model over parameters corresponds to a mixture of ProMPs and thus several variable stiffness plans, as depicted in Figure 3.11. Such GMM can be learned by solving an entropy regularized, episodic reinforcement learning problem (that is, a variational inference

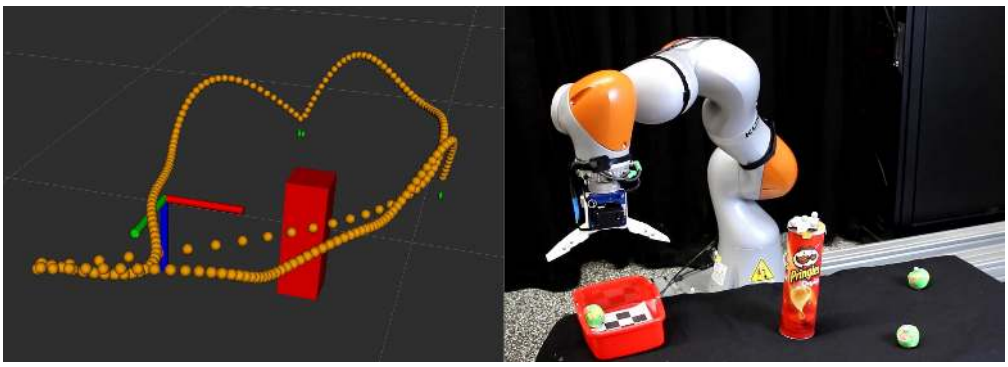


Figure 3.12.: We performed experiments on a Kuka IIWA R820 robot where we re-planned the mixture of ProMPs online. Re-planning was triggered when new obstacles appeared or when objects where grasped or released. The left image shows the visualization of the learned plans that was shown to the operator. For better visualization, the (variable) stiffness was not visualized. Image was taken from [Ewerton et al., 2020].

problem), where the entropy regularization is introduced for obtaining different plans, and the reward function (that is, the unnormalized target distribution) is assumed to be given. We applied VIPS for learning the mixture of ProMPs which—thanks to its sample efficiency—enabled us to re-plan during teleoperation. Furthermore, the negated log-density of the GMM corresponds to a smoothed minimum of convex quadratic functions. By applying a wrench proportional to the gradient of this energy landscape, we can ensure that the wrenches vary smoothly and guide the operator along the closest planned trajectory. We applied the approach to a pick-and-place task as depicted in Figure 3.12. Please refer to the original publication [Ewerton et al., 2020] for details.

3.6.2. Expected Information Maximization: Using the I-Projection for Mixture Density Estimation

For VIPS, we considered the problem of variational inference where we want to minimize the reverse KL divergence (RKL) to an unnormalized target distribution $R(\mathbf{x}) = \tilde{p}(\mathbf{x})$. In contrast, Becker et al. [2020] apply similar ideas for RKL-based density esimation, which instead of assuming that the (unnormalized) target distribution $\tilde{p}(\mathbf{x})$ is known, assumes that samples from $p(\mathbf{x})$ are given. It is well-known that the M-projection (maximum likelihood estimation) for density estimation with latent variable models—such as GMMs—can be performed using expectation-maximization [Dempster et al., 1977]. Becker et al. [2020] established a similar procedure, called expected information maximization (EIM), for performing the I-projection. Compared to the M-projection, the I-projection results in models that generate samples that are harder to distinguish from samples from the target distribution at the cost of having less variability (entropy).

To derive a lower bound on the negated RKL, we revisit the lower bound of VIPS (Eq. 3.11),

$$\tilde{L}(\boldsymbol{\theta}, \tilde{q}(o|\mathbf{x})) = \sum_o q(o) \Big[\int_{\mathbf{x}} q(\mathbf{x}|o) \big(\log p(\mathbf{x}) + \log \tilde{q}(o|\mathbf{x}) \big) d\mathbf{x} + \mathrm{H}\big(q(\mathbf{x}|o)\big) \Big] + \mathrm{H}\big(q(o)\big), \tag{3.24}$$

where we replaced the unnormalized target distribution $R(\mathbf{x}) = \log \tilde{p}(\mathbf{x})$ with the (unknown) true target distribution $p(\mathbf{x})$. As $p(\mathbf{x})$ is not known, we can not directly optimize the lower bound given by Equation 3.24. Substituting

$$\log \tilde{q}(o|\mathbf{x}) = \log \tilde{q}(o) + \log \tilde{q}(\mathbf{x}|o) - \log \tilde{q}(\mathbf{x})$$

in Equation 3.24 we obtain

$$\tilde{L}(\boldsymbol{\theta}, \tilde{q}(o|\mathbf{x})) = \sum_o q(o) \Big[\int_{\mathbf{x}} q(\mathbf{x}|o) \underbrace{\log \left(\frac{p(\mathbf{x})}{\tilde{q}(\mathbf{x})} \right)}_{\phi(\mathbf{x})} d\mathbf{x} + D_{\mathrm{KL}}\big(q(\mathbf{x}|o)\big) || \tilde{q}(\mathbf{x}|o) \big) \Big] + D_{\mathrm{KL}}\big(q(o) || \tilde{q}(o)\big), \tag{3.25}$$

where the log density-ratio $\phi(\mathbf{x})$ now serves as reward function. We directly estimate the log density ratio $\phi(\mathbf{x})$ based on samples from $p(\mathbf{x})$ and our model $\tilde{q}(\mathbf{x})$, which can be framed as a classification problem [Menon and Ong, 2016]. Namely, we train a classifier to discriminate samples from $p(\mathbf{x})$ and $\tilde{q}(\mathbf{x})$. As shown by Sugiyama et al. [2012], and to be discussed in Section 4.1.2, the logits of an optimal classifier approximate the log

density-ratio $\phi(\mathbf{x})$. Comparing the different formulations of the lower bound given by Equation 3.24 and Equation 3.25, we can see that Equation 3.24 encourages entropy using entropy-objectives and locality by providing the log-responsibility $\log \tilde{q}(o|\mathbf{x})$ as additional reward, whereas Equation 3.25 encourages locality using KL objectives and entropy by providing the current log-densities $\log \tilde{q}(\mathbf{x})$ as additional cost. Using the approximate log density-ratio $\tilde{\phi}(\mathbf{x})$ as reward function, EIM optimizes the lower bound (Eq. 3.25) similar to VIPS by using MORE to update the individual components and a variant of episodic REPS [Peters et al., 2010] for updating the weights.

By only requiring samples from the target distribution, EIM brings the lower bound that was used by VIPS one step closer to being applicable to imitation learning and inverse reinforcement learning. However, these problem settings are characterized by the presence of time-series data. How to make use of a similar lower bound when data is generated by a Markov decision process will be the focus of the next chapter.

4. Non-Adversarial Imitation Learning and its Connections to Adversarial Inverse Reinforcement Learning

Imitation learning [IL, Osa et al., 2018; Schaal, 1999] and inverse reinforcement learning [IRL, Ng and Russell, 2000] are two related areas of research that aim to teach agents by providing demonstrations of the desired behavior. Whereas imitation learning aims to learn a *policy* that results in a similar behavior, inverse reinforcement learning focuses on inferring a *reward function* that might have been optimized by the demonstrator, aiming to better generalize to different environments. Both areas of research are often formalized as distribution-matching, that is, the learned policy (or the optimal policy for IRL) should induce a distribution over states and actions that is close to the expert's distribution with respect to a given (usually non-metric) distance. Commonly applied distances are the forward Kullback-Leibler (KL) divergence [e.g., Ziebart, 2010], which maximizes the likelihood of the demonstrated state-action pairs under the agent's distribution, and the reverse Kullback-Leibler (RKL) divergence [e.g., Arenz et al., 2016; Fu et al., 2018; Ghasemipour et al., 2020] which minimizes the expected discrimination information [Kullback and Leibler, 1951] of state-action pairs sampled from the agent's distribution. However, since the emergence of generative adversarial networks [GANs, Goodfellow et al., 2014] as a solution technique for both areas, other divergences have been investigated such as the Jensen-Shannon divergence [Ho and Ermon, 2016], the Wasserstein distance [Xiao et al., 2019] and general f-divergences [Ghasemipour et al., 2020; Ke et al., 2019]. Although GANs are typically not applied to time-series data, their application for imitation learning is surprisingly straightforward. The discriminator can typically be trained in the exact same way—agnostically to the data-generation process—by aiming to discriminate state-action samples of the agent and the expert. The generator objective, on the other hand, can be typically solved using an off-the-shelf reinforcement learning [RL, Sutton and Barto, 1998] algorithm. Apart from this flexibility, the adversarial formulation is also appealing for scaling to neural network policies and reward functions and for its efficiency, since unlike some previous approaches [e.g., Ratliff et al., 2006; Ziebart, 2010], they do not

require to solve a full reinforcement learning problem iteratively but only perform few policy updates at each iteration. However, it is often difficult to achieve stable optimization with adversarial approaches. Firstly, the discriminator typically relies on an estimate of the probability density ratio which is difficult to approximate for high-dimensional problems, especially for those areas of the state-action-space that are not encountered by the current policy or the expert. Secondly, the reward signal provided by the discriminator is specific to the current policy and, thus, can quickly become invalid if the generator is updated to greedily.

In this work, we directly address the latter problem. We derive an upper bound on the reverse Kullback-Leibler divergence between the agent's and expert's distribution which allows us to guarantee improvement even for large policy updates (when assuming an optimal discriminator). By iteratively tightening and optimizing this bound similar to expectation-maximization, we can show convergence to the optimal solution. Similar to adversarial methods, the reward signal in our non-adversarial formulation is based on a density-ratio that is learned by training a discriminator to classify samples from the agent's distribution and the expert's distribution. However, in contrast to the adversarial formulation, our reward function is explicitly defined with respect to the density ratio based on the agent's previous distribution which is achieved by adding a term that penalizes the divergence to the last policy. However, our non-adversarial formulation is not only closely connected to adversarial imitation learning, but also to inverse reinforcement learning: As our reward signal is not specific to the current policy, it can serve as reward function for which any maximizing policy matches the expert demonstrations.

Indeed, we show that adversarial inverse reinforcement learning [AIRL, Fu et al., 2018] learns and optimizes the lower bound reward function of our non-adversarial formulation suggesting that AIRL is better viewed as a non-adversarial method. To the best of our knowledge, the theoretical justification of AIRL is currently not well understood. For example, Fu et al. [2018] justify AIRL as an instance of maximum causal entropy inverse reinforcement learning [MaxCausalEnt-IRL, Ziebart, 2010] by relating the update of their reward function—which corresponds to an energy-based model of the policy—with the gradient of the maximum likelihood (forward KL) objective of MaxCausalEnt-IRL. However, we clarify that the gradients of the different objectives only coincide after convergence and, furthermore, only when the expert demonstrations are perfectly matched and, thus, any divergence is minimized. Furthermore, AIRL is not a typical adversarial method, since the discriminator directly depends on the policy and is, thus, not held constant during the generator update. To the best of our knowledge, it has never been investigated whether the theoretical analyses of generative adversarial nets apply to this setting. However, based on our non-adversarial formulation, we show that AIRL indeed enjoys stronger convergence guarantees than adversarial methods since we can drop the requirement of

sufficiently small policy updates at each iteration and instead only assume improvement with respect to the current reward function.

Apart from deepening our understanding of AIRL, our non-adversarial formulation gives rise to novel algorithms for imitation learning and inverse reinforcement learning. For example, we show that the Q-function for our non-adversarial reward function can be estimated offline based on DualDice [Nachum et al., 2019], a method for offline density-ratio estimation. The resulting algorithm is closely connected to the recently proposed imitation learning method ValueDice [Kostrikov et al., 2020] but does not involve solving a saddle point problem.

The contributions of this work include

- introducing non-adversarial imitation learning (NAIL), a framework for imitation learning that resembles adversarial imitation learning but enjoys stronger convergence guarantees by not involving a saddle point problem,
- presenting an alternate derivation for adversarial inverse reinforcement learning based on our non-adversarial formulation,
- introducing offline non-adversarial imitation learning (O-NAIL), an instance of non-adversarial imitation learning that does not involve interactions with the environment, and
- presenting a more general derivation of several adversarial imitation learning methods [Ghasemipour et al., 2020; Ho and Ermon, 2016; Torabi et al., 2018] that is based on matching noisy trajectory observations.

The remainder of this article is structured as follows. We formally specify our more general formulation for imitation learning and discuss adversarial imitation learning and AIRL in Section 4.1. Our derivations for non-adversarial imitation learning are presented in Section 4.2. In Section 4.3.1, we investigate AIRL through the lens of non-adversarial imitation learning. In Section 4.3.2, we present an offline imitation learning algorithm based on our non-adversarial formulation, and in Section 4.4 we present experimental results. The main insights from our work are discussed in Section 4.5.

4.1. Preliminaries

After formalizing the problem setting, we will discuss how GANs can be applied for imitation learning. Furthermore, we will discuss the modifications employed by AIRL for extracting a reward function.

4.1.1. Problem Formulation

We consider a Markov decision process for the discounted, infinite horizon setting. At each time step t, an agent observes the state $\mathbf{s}_t$ and uses a stochastic policy $\pi(\mathbf{a}_t|\mathbf{s}_t)$ to choose an action $\mathbf{a}_t$. Afterwards, with probability $\gamma < 1$, the agent transitions to the next state $\mathbf{s}_{t+1}$ according to stochastic system dynamics $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$, which we do not assume to be known. With probability $1 - \gamma$, the episode ends and the environment gets reset to an initial state $\mathbf{s}_0$ drawn from the initial state distribution $p_0(\mathbf{s})$. By assuming such environment resets, we introduce a discounting of future rewards—which is commonly applied in practice—and ensure existence of a stationary distribution, which includes transient behavior [van Hoof et al., 2017]. We refer to the tuple containing the states and actions encountered during an episode as trajectory $\boldsymbol{\tau}_i = (\mathbf{s}_0, \mathbf{a}_0, \ldots, \mathbf{s}_{T_i}, \mathbf{a}_{T_i})$, where $\mathbf{s}_{T_i}$ and $\mathbf{a}_{T_i}$ correspond to the last state and action encountered at episode i. A given policy π induces a distribution over trajectories $p^\pi(\boldsymbol{\tau})$ and for each time step t a distribution over states and actions $p_t^\pi(\mathbf{s}, \mathbf{a})$ (which can be computed from $p^\pi(\boldsymbol{\tau})$ by marginalization). The stationary distribution over states and actions induced by a policy π is given by $p^\pi(\mathbf{s}, \mathbf{a}) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t p_t^\pi(\mathbf{s}, \mathbf{a})$.

In a reinforcement learning setting, the agent would further obtain a reward $r(\mathbf{s}_t, \mathbf{a}_t)$ at each time step t and would aim to find a policy that maximizes the expected reward $J_{\text{RL}} = \int_{\mathbf{s},\mathbf{a}} p^\pi(\mathbf{s}, \mathbf{a}) r(\mathbf{s}, \mathbf{a}) d\mathbf{s} d\mathbf{a}$. However, for the purpose of this work, we do not assume that a reward function is available. Instead, we consider the problem of imitation learning from observations, which generalizes state-action based imitation learning. Namely, we assume that the agent observes a set of N expert demonstrations $\mathcal{D} = \{\mathbf{o}_i\}_{1 \leq i \leq N}$ in some observation space $\mathcal{O}$. We further assume that $p(\mathbf{o}|\boldsymbol{\tau})$, the probabilistic mapping from the agent's trajectory to a distribution of observations, is given. For example, if the observation space is given by the end-effector pose of a robot and the state-space includes the robot's joint position, the mapping from trajectory to observation space would be given as the distribution over end-effector poses during the trajectory, which can be computed from the states using the robot's forward kinematics. We do not assume that the states and actions of the expert are observed nor do we assume that the expert acts in the same MDP as the agent. In the aforementioned example, the demonstrations might be recorded by tracking the hand pose of a human expert. Furthermore, depending on the probabilistic mapping, we can match distributions over full trajectories, individual steps or transitions. This formulation generalizes the setting of several imitation learning methods such as, GAN-GCL [Finn et al., 2016a], GAIL [Ho and Ermon, 2016], AIRL [Fu et al., 2018] or GAIfO [Torabi et al., 2018]. In imitation learning, we aim to minimize a given divergence $D(p^\pi(\mathbf{o})||q(\mathbf{o}))$ between the agent's distribution over observations $p^\pi(\mathbf{o})$ and the expert's

distribution over observations $q(\mathbf{o})$, that is,

$$J_{\text{IL}} = \min_{\pi} D(p^{\pi}(\mathbf{o})||q(\mathbf{o})). \tag{4.1}$$

Optimizing Objective 4.1 is complicated by the fact that the agent's distribution over observations $p^{\pi}(\mathbf{o})$ is not analytically known and can only be controlled implicitly by changing the agent's policy.

4.1.2. Generative Adversarial Nets

We will now briefly review (Jensen-Shannon-)GANs and their generalization to general f-divergences. We will also discuss the connection between density-ratio estimation and optimizing a discriminator.

Jensen-Shannon-GANs

Generative adversarial nets [Goodfellow et al., 2014] are a popular technique to train implicit distributions to produce samples that are similar to samples from an unknown target distribution. Implicit distributions are distributions for which the density function is implicitly defined by a sampling procedure but not explicitly modeled. Goodfellow et al. [2014] proposed to minimize the Jensen-Shannon Divergence $D_{\text{JS}}(p(\mathbf{x})||q(\mathbf{x}))$ between an implicit distribution $p(\mathbf{x})$ and a data distribution $q(\mathbf{x})$ by solving the saddle point problem

$$J_{\text{GAN}}(p, D) = \min_{p} \max_{D} \mathrm{E}_{\mathbf{x}\sim q} \left[\log D(\mathbf{x})\right] + \mathrm{E}_{\mathbf{x}\sim p(\mathbf{x})} \left[\log \left(1 - D(\mathbf{x})\right)\right]. \tag{4.2}$$

Here, $D(\mathbf{x})$ (typically a neural network) is called *discriminator* and assigns a scalar value in the range $]0, 1[$ to each sample $\mathbf{x}$. The *generator* $p(\mathbf{x})$ is typically represented as a neural network $G(\mathbf{z})$ that transforms Gaussian input noise $\mathbf{z} \sim \mathcal{N}(\mathbf{z})$. However, the derivations provided by Goodfellow et al. [2014] also apply for general implicit distributions $p(\mathbf{x})$. Goodfellow et al. [2014] show that solving the saddle point problem (Eq. 4.2) minimizes the Jensen-Shannon divergence and that the optimal solution can be found by alternating between optimizing the discriminator to convergence and performing small generator updates. However, in practice, the discriminator also obtains only slight updates at each iteration in order to avoid vanishing gradients [Arjovsky and Bottou, 2017]. The discriminator objective corresponds to minimizing the binary cross-entropy loss which is commonly used for binary classification. Hence, the discriminator is trained to classify samples from the data distribution and samples from the generator. The generator objective does not involve the probability density of the generator and can, thus, also

be optimized for implicit distributions. Whereas, the original formulation by Goodfellow et al. [2014] minimizes the Jensen-Shannon Divergence, similar ideas can also be applied to other divergences [Arjovsky et al., 2017; Nowozin et al., 2016]. We will briefly review f-GANs [Nowozin et al., 2016], which can minimize general f-divergences because the family of f-divergences also includes the reverse Kullback-Leibler divergence which is central to this work.

f-GANs

The family of f-divergences is defined for a given convex, lower-semicontinuous function f as

$$D_f(q(\mathbf{x})||p(\mathbf{x})) = \mathrm{E}_p\left[f\Big(\frac{q(\mathbf{x})}{p(\mathbf{x})}\Big)\right].$$

Nowozin et al. [2016] showed that we can minimize the f-divergence between the data distribution q and an implicit distribution p by solving the saddle point problem

$$J_{\text{F-GAN}}(p, D) = \min_p \max_D \mathrm{E}_{\mathbf{x}\sim q}\left[D(\mathbf{x})\right] - \mathrm{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[f^*(D(\mathbf{x}))\right], \tag{4.3}$$

where

$$f^*(t) = \sup_{u\in \mathrm{dom}_f} ut - f(u)$$

is the convex conjugate of f. Depending on the choice of f, we need to ensure that the output of the discriminator respects the domain of the convex conjugate, that is $\forall_{\mathbf{x}} : D(\mathbf{x}) \in \mathrm{dom}_{f^*}$, which can be achieved by applying an appropriate output activation. Nowozin et al. [2016] provide for several common choices of f their respective convex conjugates and suitable output activations. Nowozin et al. [2016] also note, that the optimal discriminator $D^\star(\mathbf{x})$ for a given generator $p(\mathbf{x})$ corresponds to the derivative of f evaluated at the density-ratio $\frac{q(\mathbf{x})}{p(\mathbf{x})}$, that is,

$$D^\star(\mathbf{x}) = f'\Big(\frac{q(\mathbf{x})}{p(\mathbf{x})}\Big).$$

Hence, for any f-divergence and when assuming the discriminator to be optimal, optimizing the f-GAN objective given by Equation 4.3 with respect to the generator,

$$\max_p \mathrm{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[f^*(D^\star(\mathbf{x}))\right] = \max_p \mathrm{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[f^* \circ f'\Big(\frac{q(\mathbf{x})}{p(\mathbf{x})}\Big)\right] = \max_p \mathrm{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[g\Big(\frac{q(\mathbf{x})}{p(\mathbf{x})}\Big)\right], \tag{4.4}$$

corresponds to maximizing the expected value of a function $g = f^\star \circ f'$ of the density-ratio $\phi(\mathbf{x}) = \frac{q(\mathbf{x})}{p(\mathbf{x})}$, which closely connects the problem of minimizing f-divergences with density-ratio estimation.

A Connection to Density-Ratio Estimation

Density-ratio estimation considers the problem of estimating the density-ratio $\phi(\mathbf{x})$ based on samples from $q(\mathbf{x})$ and $p(\mathbf{x})$. A naive approach would perform density estimation to independently estimate both distributions and then approximate the density ratio based on these estimates. However, while it is possible to compute the density ratio from the individual density, it is not possible to recover the individual densities from their ratio, and, thus, density-ratio estimation is a simpler problem than density estimation [Sugiyama et al., 2012]. Indeed, density-ratio estimation is closely related to binary classification [Menon and Ong, 2016].

To illustrate this connection, consider the following binary classification task. Assume that we aim to classify samples that have been drawn with probability $z(Q)$ from the distribution $q(\mathbf{x})$ and with probability $z(\neg Q) = 1 - z(Q)$ from the distribution $p(\mathbf{x})$. Hence, we consider the mixture model

$$z(\mathbf{x}) = z(Q)q(\mathbf{x}) + (1 - z(Q))p(\mathbf{x}),$$

where the class frequencies $z(Q)$ and $z(\neg Q)$ are known, and we want to learn a model $\tilde{z}(Q|\mathbf{x})$ to approximate the conditional class probabilities $z(Q|\mathbf{x})$—for example, by minimizing the expected cross entropy between $z(Q|\mathbf{x})$ and $\tilde{z}(Q|\mathbf{x})$ as in the inner maximization in Eq. 4.2. The model is typically represented by squashing learned logits $\nu(\mathbf{x})$ with a sigmoid, that is

$$\tilde{z}(Q|\mathbf{x}) = \frac{1}{1 + \exp\left(-\nu(\mathbf{x})\right)} \Rightarrow \nu(\mathbf{x}) = \log\Big(\frac{\tilde{z}(Q|\mathbf{x})}{1 - \tilde{z}(Q|\mathbf{x})}\Big).$$

At the optimum, where $\tilde{z}(Q|\mathbf{x}) \approx z(Q|\mathbf{x})$, we have

$$\nu(\mathbf{x}) \approx \log\Big(\frac{z(Q|\mathbf{x})}{1 - z(Q|\mathbf{x})}\Big) = \log\Big(\frac{z(Q)q(\mathbf{x})}{z(\neg Q)p(\mathbf{x})}\Big) = \log\Big(\frac{z(Q)}{z(\neg Q)}\Big) + \log\Big(\phi(\mathbf{x})\Big). \tag{4.5}$$

Hence, when choosing equal class frequencies, that is $z(Q) = z(\neg Q) = 0.5$, the logits $\nu(\mathbf{x})$ of the optimal classifier approximate the log density-ratio $\log(\phi(\mathbf{x}))$.

4.1.3. Adversarial Imitation Learning

As generative adversarial nets can be used to minimize a variety of different divergences between an implicit distribution and the data distribution, they are suitable also for imitation learning. When applying GANs to imitation learning, the generator is given by the distribution over observations induced by the agent's policy, $p^\pi(\mathbf{o}) = \int_{\boldsymbol{\tau}} p^\pi(\boldsymbol{\tau}) p(\mathbf{o}|\boldsymbol{\tau}) d\boldsymbol{\tau}$. Ho and Ermon [2016] introduced this idea when they proposed generative adversarial imitation learning (GAIL), where they applied the Jensen-Shannon-GAN objective (Eq. 4.2). However, we will directly consider the more general f-GAN objective given by Equation 4.3, which was congruently proposed for imitation learning by Ke et al. [2019] and Ghasemipour et al. [2020].

f-GANs for Imitation Learning

When using an f-GAN for imitation learning, the saddle-point problem is given by

$$J_{\text{AIL}}(\pi, D) = \min_{\pi} \max_{D} \mathrm{E}_{\mathbf{o} \sim q} \left[D(\mathbf{o}) \right] - \mathrm{E}_{\mathbf{o} \sim p^\pi(\mathbf{o})} \left[f^*(D(\mathbf{o})) \right]. \tag{4.6}$$

Please note that the minimization is still performed over $p^\pi(\mathbf{o})$ which we can only affect through π. Optimizing the discriminator is performed in the same way as for standard f-GANs using samples from $p^\pi(\mathbf{o})$ that can be obtained by executing the policy π. In the trajectory-centric formulation, where we make no further assumptions than $p^\pi(\mathbf{o}) = \int_{\boldsymbol{\tau}} p^\pi(\boldsymbol{\tau}) p(\mathbf{o}|\boldsymbol{\tau}) d\boldsymbol{\tau}$, optimizing the generator corresponds to an episodic reinforcement learning problem

$$\max_{\pi} \mathrm{E}_{\boldsymbol{\tau} \sim p^\pi(\boldsymbol{\tau})} \left[r_{\text{ep}}(\boldsymbol{\tau}) \right], \tag{4.7}$$

for the episodic reward

$$r_{\text{ep}}(\boldsymbol{\tau}) = \int_{\mathbf{o}} p(\mathbf{o}|\boldsymbol{\tau}) f^*(D(\mathbf{o})) d\mathbf{o}.$$

As we can only obtain a reward signal for full trajectories, we can not apply standard reinforcement learning algorithms that typically assume Markovian rewards $r(\mathbf{s}, \mathbf{a})$ or $r(\mathbf{s}, \mathbf{a}, \mathbf{s}')$, where $\mathbf{s}'$ is the state at the next time step. Instead, the policy can be optimized using stochastic search or black-box optimizers [Abdolmaleki et al., 2015; Hansen et al., 2003].

Hence, adversarial imitation learning methods are typically restricted to step-based distribution matching. Furthermore, they often assume direct observations of states and actions, that is, they aim to match $q(\mathbf{s}, \mathbf{a})$ [Ho and Ermon, 2016; Xiao et al., 2019], $q(\mathbf{s})$ [Ghasemipour et al., 2020], or $q(\mathbf{s}, \mathbf{s}')$ [Torabi et al., 2018]. We can incorporate such restrictions by making further assumptions on the form of $p(\mathbf{o}|\boldsymbol{\tau})$ as shown by Proposition 1.

Proposition 1. *Assume that the distribution over observations at a given time step t is completely characterized by the state $\mathbf{s}_t$ and action $\mathbf{a}_t$ at that time step, and hence,*

$$p(\mathbf{o}|\boldsymbol{\tau}, t) = p(\mathbf{o}|\mathbf{s}_t^{\boldsymbol{\tau}}, \mathbf{a}_t^{\boldsymbol{\tau}}).$$

Then, the episodic reinforcement learning problem given by Equation 4.7 can be solved by maximizing the Markovian reward function

$$r_{adv}(\mathbf{s}, \mathbf{a}) = \int_{\mathbf{o}} p(\mathbf{o}|\mathbf{s}, \mathbf{a}) f^*(D(\mathbf{o})) d\mathbf{o}. \tag{4.8}$$

Proof. See Appendix C.2. Intuitively, when the distribution over observations can be computed independently for each time step using $p(\mathbf{o}|\mathbf{s}, \mathbf{a})$, we can obtain $p(\mathbf{o})$ also by marginalizing based on the stationary distribution $p^\pi(\mathbf{s}, \mathbf{a})$, which turns the optimization over the policy in Eq. 4.6 into a step-based reinforcement learning problem. □

While the restriction to step-based distribution matching is necessary for obtaining Markovian rewards, assuming direct observerations of states and action is in general not necessary. Indeed, dropping this assumption greatly generalizes the problem formulation of imitation learning, for example, by enabling us to imitate certain features (e.g. hand poses) of a human demonstration without assuming that their states and actions can be observed or mapped to the agent's MDP. However, while our derivations for non-adversarial and adversarial imitation learning are formulated for general observations, both instances of NAIL that we will discuss in Section 4.3 assume direct state-action observations which are exploited by the optimization.

Proposition 1 can be straightforwardly extended to also include the next state by assuming that $p(\mathbf{o}|\boldsymbol{\tau}, t) = p(\mathbf{o}|\mathbf{s}_t^{\boldsymbol{\tau}}, \mathbf{a}_t^{\boldsymbol{\tau}}, \mathbf{s}_{t+1}^{\boldsymbol{\tau}})$, and when maximizing the reward with respect to the stationary distribution $p^\pi(\mathbf{s}, \mathbf{a}, \mathbf{s}')$. However, the observation model needs to treat the last time step of an episode $\boldsymbol{\tau}_i$ differently, because we can not make use of the next state $\mathbf{s}'$ which corresponds to the first time step of the next episode $\boldsymbol{\tau}_{i+1}$ and thus can not affect $p(\mathbf{o}|\boldsymbol{\tau}_i)$. Instead, we could emit a distinct observation $\mathbf{o}_{reset}$—for example, by setting a special flag [Kostrikov et al., 2018]—for final time steps. However, in the following, we will only consider the more common imitation learning setting, where the observations and, thus, the rewards at time step t only depend on $\mathbf{s}_t$ and $\mathbf{a}_t$.

Choosing a Divergence

The exact form of the reward function depends on the choice of f and, thus, the divergence that we want to minimize. When assuming that the expert's distribution can be matched

exactly, all divergences yield the same optimal solution $p^\pi(\mathbf{o}) = q(\mathbf{o})$. However, especially due to restriction on the agent's policy—which is often Gaussian in the actions for a given state—matching the expert's distribution exactly is often not feasible. In such cases, the choice of divergence has large impact on the solution. For example, it is well-known that the forward KL divergence

$$D_{\text{KL}}(q(\mathbf{o})||p^\pi(\mathbf{o})) = D_f^{\text{KL}}(p^\pi(\mathbf{o})||q(\mathbf{o})) = \mathrm{E}_q\left[\log\left(\frac{q(\mathbf{o})}{p^\pi(\mathbf{o})}\right)\right],$$

which corresponds to $f(u) = u\log(u)$, results in a mode averaging behavior. The agent's policy will maximize the likelihood of all expert demonstrations, which may require putting most of its probability mass on areas where we do not have expert observations. As the resulting behavior can potentially be dangerous, it is often argued that the reverse KL divergence should be preferred for imitation learning [Finn et al., 2016a; Ghasemipour et al., 2020]. The reverse KL divergence,

$$D_{\text{KL}}(p^\pi(\mathbf{o})||q(\mathbf{o})) = D_f^{\text{RKL}}(p^\pi(\mathbf{o})||q(\mathbf{o})) = \mathrm{E}_{p^\pi}\left[\log\left(\frac{p^\pi(\mathbf{o})}{q(\mathbf{o})}\right)\right],$$

is obtained when choosing $f(u) = -\log(u)$ and typically results in a mode-seeking behavior. The resulting policy will assign high likelihood to as many expert observations as possible while avoiding putting much probability mass on areas where we do not have expert observations. Although the resulting behavior might not exhibit the same variety as the expert, it will avoid producing any observations that are significantly different from the expert demonstrations, resulting in a safer behavior. Hence, we also focus on the reverse KL divergence for deriving our non-adversarial imitation learning formulation. The convex conjugate and the derivative for $f_{\text{RKL}}(u) = -\log(u)$ are given by

$$f^*_{\text{RKL}}(t) = -1 - \log(-t) \qquad \text{and} \qquad f'_{\text{RKL}}(u) = -\frac{1}{u}.$$

For an optimal discriminator, the observation-based reward function is, thus,

$$r(\mathbf{o}) = f^*_{\text{RKL}} \circ f'_{\text{RKL}}\Big(\frac{q(\mathbf{o})}{p^\pi(\mathbf{o})}\Big) = -1 + \log\left(\frac{q(\mathbf{o})}{p^\pi(\mathbf{o})}\right)$$

or

$$r_{adv}(\mathbf{s},\mathbf{a}) = -1 + \log\left(\frac{q(\mathbf{s},\mathbf{a})}{p^\pi(\mathbf{s},\mathbf{a})}\right) \tag{4.9}$$

when we perform step-based matching and directly observe state and actions. As the domain of f^*_{RKL} is $\mathbb{R}^-$, a suitable output activation for the discriminator is $D(\mathbf{o}) = -\exp(\nu(\mathbf{o}))$

[Nowozin et al., 2016]. The discriminator loss based on Equation 4.3 for the reverse KL-divergence is then given by

$$J_{\text{RKL-FGAN}}(\nu) = \max_{\nu} \mathrm{E}_{\mathbf{o}\sim p^{\pi}(\mathbf{o})}\left[\nu(\mathbf{o})\right] - \mathrm{E}_{\mathbf{o}\sim q(\mathbf{o})}\left[\exp \nu(\mathbf{o})\right] + 1, \tag{4.10}$$

where the optimal solution

$$\nu^{\star}(\mathbf{o}) = \log\left(-D^{\star}(\mathbf{o})\right) = \log\left(-f'_{\text{RKL}}\Big(\frac{q(\mathbf{o})}{p^{\pi}(\mathbf{o})}\Big)\right) = \log\Big(\frac{p^{\pi}(\mathbf{o})}{q(\mathbf{o})}\Big) = -r(\mathbf{o}) - 1$$

corresponds to the negated log density-ratio and thus to the cost function $-r(\mathbf{o})$, apart from a constant offset that does not affect the generator optimization. Indeed, the f-GAN discriminator objective for the reverse KL is also known as a generalized form [Menon and Ong, 2016] of Kullback-Leibler importance estimation (KLIEP) [Sugiyama et al., 2008], which is well-known loss for density-ratio estimation. Instead of learning the density-ratio based on Equation 4.10, we can also use the discriminator objective of the Jensen-Shannon-GAN (Eq. 4.2), where—as shown in Section 4.1.2—the logits of the discriminator also converge to the log density-ratio.

We will now review adversarial inverse reinforcement learning (AIRL) which also minimizes the reverse KL, albeit it uses a reward function that subtly differs from the adversarial formulation given by Equation 4.9.

4.1.4. Adversarial Inverse Reinforcement Learning

Adversarial inverse reinforcement learning was derived as an instance of maximum causal entropy inverse reinforcement learning (MaxEnt-IRL, [Ziebart et al., 2008]), which minimizes the *forward* KL to the expert distribution. Hence, we will briefly discuss MaxEnt-IRL.

Interlude: Maximum Causal Entropy IRL

MaxCausalEnt-IRL assumes the expert to use the policy

$$\pi_{\text{expert}}(\mathbf{a}|\mathbf{s}) = \exp\Big(Q^{\text{soft}}(\mathbf{s},\mathbf{a}) - V^{\text{soft}}(\mathbf{s})\Big) = \exp\big(A^{\text{soft}}\big). \tag{4.11}$$

Here, the soft-Q function Q^{soft}, the soft-value function V^{soft} and the soft-advantage function A^{soft} are defined as [Ziebart, 2010]

$$\begin{aligned} Q^{\text{soft}}(\mathbf{s},\mathbf{a}) &= r_{\text{expert}}(\mathbf{s},\mathbf{a}) + \gamma \int_{\mathbf{s}'} p(\mathbf{s}'|\mathbf{s},\mathbf{a}) V^{\text{soft}}(\mathbf{s}') d\mathbf{s}', \\ V^{\text{soft}}(\mathbf{s}) &= \log \int_{\mathbf{a}} \exp\big(Q^{\text{soft}}(\mathbf{s},\mathbf{a})\big) d\mathbf{a}, \\ A^{\text{soft}}(\mathbf{s},\mathbf{a}) &= Q^{\text{soft}}(\mathbf{s},\mathbf{a}) - V^{\text{soft}}(\mathbf{s}). \end{aligned} \tag{4.12}$$

The soft-value function $V^{\text{soft}}(\mathbf{s})$ is the log-normalizer of π, but also corresponds to the value (which includes the expected entropy to come) of state $\mathbf{s}$ for the optimal policy of the entropy-regularized reinforcement learning problem

$$\max_{\pi} J_{\text{maxent-RL}}(\pi) = \max_{\pi} \int_{\mathbf{s},\mathbf{a}} p^{\pi}(\mathbf{s},\mathbf{a}) \left(r_{\text{expert}}(\mathbf{s},\mathbf{a}) - \log \pi(\mathbf{a}|\mathbf{s}) \right) d\mathbf{s} d\mathbf{a}. \tag{4.13}$$

The expert model (Equation 4.11) is indeed well-motivated as it follows the principle of maximum (causal) entropy [Jaynes, 1957; Ziebart, 2010]. Namely, Ziebart [2010] showed that this expert model corresponds to the maximum entropy policy that matches given empirical features $\tilde{\mathbf{f}}(\mathbf{s},\mathbf{a})$ in expectation for a linear reward function

$$r_{\text{expert}}(\mathbf{s},\mathbf{a}) = \boldsymbol{\theta}^{\top} \mathbf{f}(\mathbf{s},\mathbf{a}),$$

where the feature function $\mathbf{f}$ is assumed to be given.

Maximum causal entropy IRL learns the parameters $\boldsymbol{\theta}$ of the expert's reward function by maximizing the likelihood of the expert demonstrations

$$\begin{aligned}
\mathcal{L} &= \mathrm{E}_{\mathbf{s},\mathbf{a}\sim q(\mathbf{s},\mathbf{a})} \left[\left(r_{\boldsymbol{\theta}} + \gamma \mathrm{E}_{\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})} V^{\text{soft},\boldsymbol{\theta}}(\mathbf{s}') - V^{\text{soft},\boldsymbol{\theta}}(\mathbf{s}) \right) \right] \\
&= \mathrm{E}_{\mathbf{s},\mathbf{a}\sim q(\mathbf{s},\mathbf{a})} \left[r_{\boldsymbol{\theta}} \right] + (1-\gamma) \Big(\sum_{t=1}^{\infty} \gamma^t \mathrm{E}_{\mathbf{s}\sim q_t(\mathbf{s})} \left[V^{\text{soft},\boldsymbol{\theta}}(\mathbf{s}) \right] - \sum_{t=0}^{\infty} \gamma^t \mathrm{E}_{\mathbf{s}\sim q_t(\mathbf{s})} \left[V^{\text{soft},\boldsymbol{\theta}}(\mathbf{s}) \right] \Big) \\
&= \mathrm{E}_{\mathbf{s},\mathbf{a}\sim q(\mathbf{s},\mathbf{a})} \Big[r_{\boldsymbol{\theta}} \Big] - (1-\gamma) \mathrm{E}_{\mathbf{s}\sim p_0(\mathbf{s})} \left[V^{\text{soft},\boldsymbol{\theta}}(\mathbf{s}) \right].
\end{aligned} \tag{4.14}$$

As shown by Ziebart [2010], the gradient of the log-likelihood (Eq. 4.14) is given by

$$\begin{aligned}
\frac{d\mathcal{L}}{d\boldsymbol{\theta}} &= \mathrm{E}_{\mathbf{s},\mathbf{a}\sim q(\mathbf{s},\mathbf{a})} \Big[\frac{dr_{\boldsymbol{\theta}}}{d\boldsymbol{\theta}} \Big] - \mathrm{E}_{\mathbf{s},\mathbf{a}\sim p^{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})} \Big[\frac{dr_{\boldsymbol{\theta}}}{d\boldsymbol{\theta}} \Big] \\
&= \mathrm{E}_{\mathbf{s},\mathbf{a}\sim q(\mathbf{s},\mathbf{a})} \Big[\mathbf{f}(\mathbf{s},\mathbf{a}) \Big] - \mathrm{E}_{\mathbf{s},\mathbf{a}\sim p^{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})} \Big[\mathbf{f}(\mathbf{s},\mathbf{a}) \Big],
\end{aligned} \tag{4.15}$$

where $p^{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})$ is the state-action distribution induced by the optimal policy of the entropy regularized reinforcement learning problem (Eq. 4.13) for the reward function $r_{\boldsymbol{\theta}}$. The first term of the gradient (Eq. 4.15) can be approximated based on samples from the expert distribution. For estimating the second term one can either learn the optimal policy at each iteration [Ziebart, 2010]—which is only feasible for simple problems such as low-dimensional discrete problems [Levine et al., 2011] or linear-quadratic regulators [Monfort et al., 2015]—or employ importance sampling [Boularias et al., 2011; Finn et al., 2016b].

Adversarial Inverse Reinforcement Learning

AIRL is based on an adversarial formulation and models the reward function $r_{\boldsymbol{\theta}}$ as a neural network which is trained as a special type of discriminator. Fu et al. [2018] show that the gradient of their discriminator objective coincides with the maximum likelihood gradient (Eq. 4.15) when the generator is optimal, that is $p^{\pi}(\mathbf{s}, \mathbf{a}) = p^{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})$. AIRL uses a special type of discriminator, which was suggested by Finn et al. [2016a] for the trajectory-centric case. Namely, the logits $\nu(\mathbf{s}, \mathbf{a})$ are given by

$$\nu(\mathbf{s}, \mathbf{a}) = \bar{\nu}_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) - \log \pi(\mathbf{a}|\mathbf{s}) \tag{4.16}$$

and thus directly depend on the policy π. Actually, Fu et al. [2018] propose a second modification to the discriminator so that their discriminator is given by

$$\nu(\mathbf{s}, \mathbf{a}, \mathbf{s}') = f_{\boldsymbol{\theta}}(\mathbf{s}) + \gamma g(\mathbf{s}') - g(\mathbf{s}) - \log \pi(\mathbf{a}|\mathbf{s})$$

which aims to recover a "disentangled", state-only reward function $f_{\boldsymbol{\theta}}(\mathbf{s})$ under the assumption of deterministic system dynamics. However, the latter modification is not relevant for our theoretical analysis, and we will, thus, focus on the discriminator given by Equation 4.16. The optimization is very similar to adversarial imitation learning by alternating between updating the discriminator using the binary cross entropy loss, and updating the policy using the discriminator logits $\nu(\mathbf{s}, \mathbf{a})$ as reward function. As shown in Section 4.1.3—and also noted by Ghasemipour et al. [2020]—such procedure would in general minimize the reverse KL divergence. However, during the discriminator update at iteration i, the discriminator logits $\nu(\mathbf{s}, \mathbf{a}) = \bar{\nu}_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) - \log \pi^{(i)}(\mathbf{a}|\mathbf{s})$ are trained such that for the current policy $\pi^{(i)}$ they approximate the density ratio $\frac{q(\mathbf{s},\mathbf{a})}{p^{\pi^{(i)}}(\mathbf{s},\mathbf{a})}$. Hence, the generator update according to the adversarial formulation, would optimize the reward that is computed based on the fixed policy $\pi^{(i)}$, that is,

$$r(\mathbf{s}, \mathbf{a}) = \nu(\mathbf{s}, \mathbf{a}) = \bar{\nu}_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) - \log \pi^{(i)}(\mathbf{a}|\mathbf{s}).$$

Instead, AIRL treats the discriminator as a direct function of π by optimizing the entropy-regularized reinforcement learning problem

$$J_{\text{AIRL-maxent}} = \max_{\pi} \int_{\mathbf{s},\mathbf{a}} p^{\pi}(\mathbf{s}, \mathbf{a}) \left(\bar{\nu}_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) - \log \pi(\mathbf{a}|\mathbf{s})\right) d\mathbf{s} d\mathbf{a}, \tag{4.17}$$

which does not correspond to the generator objective of any known adversarial formulation. Fu et al. [2018] argue that the gradient of the discriminator update corresponds to an unbiased estimate of the gradient of the MaxCausalEnt-IRL objective (Eq. 4.15) when

assuming that the entropy-regularized reinforcement learning problem (Eq. 4.17) is solved at each iteration. However, as shown in Appendix C.1 we would not only need to assume that the policy is optimal for the given reward function $\bar{\nu}_{\boldsymbol{\theta}}$, but also that this policy matches the expert distribution $q(\mathbf{s}, \mathbf{a})$ to show that the gradients coincide. Hence, we can only show that the optimal reward function of the maximum-likelihood objective is also a stationary point of the discriminator once the algorithm has converged—if the demonstrations can be perfectly matched—, but we can not show convergence of the algorithm by relating it to MaxCausalEnt-IRL. Hence, we argue that although AIRL showed some promising results, its theoretical justification is currently not well understood. We will now derive a non-adversarial formulation for imitation learning (NAIL), which is neither based on MaxCausalEnt-IRL nor on generative adversarial nets, and show that adversarial inverse reinforcement learning is indeed an instance of this non-adversarial formulation.

4.2. Non-Adversarial Imitation Learning

We aim to match the observations by minimizing the reverse KL-divergence between the agent's and the expert's respective distribution of observations, that is,

$$\begin{aligned}
&\max_{\pi} - \int_{\mathbf{o}} p^{\pi}(\mathbf{o}) \log \frac{p^{\pi}(\mathbf{o})}{q(\mathbf{o})} d\mathbf{o} \\
&= \max_{\pi} \int_{\boldsymbol{\tau}, \mathbf{o}} p^{\pi}(\boldsymbol{\tau}) p(\mathbf{o}|\boldsymbol{\tau}) \log \frac{q(\mathbf{o})}{p^{\pi}(\mathbf{o})} d\boldsymbol{\tau} d\mathbf{o} \\
&= \max_{\pi} \int_{\boldsymbol{\tau}} p^{\pi}(\boldsymbol{\tau}) r(\boldsymbol{\tau}, \pi) d\boldsymbol{\tau}.
\end{aligned} \tag{4.18}$$

Optimization Problem 4.18 resembles the optimization problem solved by (episodic) reinforcement learning. However, note that the reward function

$$r(\boldsymbol{\tau}, \pi) = \int_{\mathbf{o}} p(\mathbf{o}|\boldsymbol{\tau}) \log \frac{q(\mathbf{o})}{\int_{\boldsymbol{\tau}} p^{\pi}(\boldsymbol{\tau}) p(\mathbf{o}|\boldsymbol{\tau}) d\boldsymbol{\tau}} d\mathbf{o} \tag{4.19}$$

depends on the agent's policy via its induced trajectory distribution $p^{\pi}(\boldsymbol{\tau})$. The dependency on the current policy has two main drawbacks. Firstly, the reward function can only be used to minimize the KL-divergence if we perform sufficiently small policy updates. Secondly, the reward function given by Equation 4.19 does not solve the inverse reinforcement learning problem, since maximizing the expected reward will in general not match the expert distribution. We will now show that we can remove this dependency by formulating a lower bound on Optimization Problem 4.18. This lower bound (which corresponds to a

negated upper bound on the reverse KL) enables us to replace $p^\pi(\boldsymbol{\tau})$ in Equation 4.19 by a fixed, auxiliary distribution $\tilde{p}(\boldsymbol{\tau})$. In Section 4.2.2, we will further show that this lower bound can be used to maximize the original objective given by Equation 4.18.

4.2.1. An Upper Bound on the Reverse KL

In order to derive the lower bound, we express $p^\pi(\boldsymbol{\tau})$ in terms of an auxiliary distribution $\tilde{p}(\boldsymbol{\tau})$ as follows:

$$\begin{aligned} p^\pi(\mathbf{o}) &= \frac{p^\pi(\boldsymbol{\tau})p(\mathbf{o}|\boldsymbol{\tau})}{p^\pi(\boldsymbol{\tau}|\mathbf{o})} = \frac{p^\pi(\boldsymbol{\tau})p(\mathbf{o}|\boldsymbol{\tau})}{\tilde{p}(\boldsymbol{\tau}|\mathbf{o})}\frac{\tilde{p}(\boldsymbol{\tau}|\mathbf{o})}{p^\pi(\boldsymbol{\tau}|\mathbf{o})} = \tilde{p}(\mathbf{o})\frac{p^\pi(\boldsymbol{\tau})p(\mathbf{o}|\boldsymbol{\tau})}{\tilde{p}(\boldsymbol{\tau})p(\mathbf{o}|\boldsymbol{\tau})}\frac{\tilde{p}(\boldsymbol{\tau}|\mathbf{o})}{p^\pi(\boldsymbol{\tau}|\mathbf{o})} \\ &= \tilde{p}(\mathbf{o})\frac{p^\pi(\boldsymbol{\tau})}{\tilde{p}(\boldsymbol{\tau})}\frac{\tilde{p}(\boldsymbol{\tau}|\mathbf{o})}{p^\pi(\boldsymbol{\tau}|\mathbf{o})}. \end{aligned} \tag{4.20}$$

Based on Equation 4.20 we can express Optimization Problem 4.18 in terms of the auxiliary distribution, that is,

$$\begin{aligned} &\max_\pi \int_{\boldsymbol{\tau},\mathbf{o}} p^\pi(\boldsymbol{\tau})p(\mathbf{o}|\boldsymbol{\tau}) \log \frac{q(\mathbf{o})}{p^\pi(\mathbf{o})} d\boldsymbol{\tau} d\mathbf{o} \\ &= \max_\pi \int_{\boldsymbol{\tau},\mathbf{o}} p^\pi(\boldsymbol{\tau})p(\mathbf{o}|\boldsymbol{\tau}) \left(\log \frac{q(\mathbf{o})}{\tilde{p}(\mathbf{o})} - \log \frac{p^\pi(\boldsymbol{\tau})}{\tilde{p}(\boldsymbol{\tau})} + \log \frac{p^\pi(\boldsymbol{\tau}|\mathbf{o})}{\tilde{p}^\pi(\boldsymbol{\tau}|\mathbf{o})} \right) d\boldsymbol{\tau} d\mathbf{o} \\ &= \max_\pi \int_{\boldsymbol{\tau},\mathbf{o}} p^\pi(\boldsymbol{\tau})p(\mathbf{o}|\boldsymbol{\tau}) \log \frac{q(\mathbf{o})}{\tilde{p}(\mathbf{o})} d\boldsymbol{\tau} d\mathbf{o} - \mathrm{KL}\left(p^\pi(\boldsymbol{\tau})||\tilde{p}(\boldsymbol{\tau})\right) \\ &\quad + \mathrm{E}_{p^\pi(\mathbf{o})}\left[\mathrm{KL}\left(p^\pi(\boldsymbol{\tau}|\mathbf{o})||\tilde{p}(\boldsymbol{\tau}|\mathbf{o})\right)\right] \\ &= \max_\pi \int_{\boldsymbol{\tau},\mathbf{o}} p^\pi(\boldsymbol{\tau})p(\mathbf{o}|\boldsymbol{\tau}) \log \frac{q(\mathbf{o})}{\tilde{p}(\mathbf{o})} d\boldsymbol{\tau} d\mathbf{o} \\ &\quad - (1-\gamma)\sum_t \gamma^t \int_{\mathbf{s}} p_t^\pi(\mathbf{s}) \int_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \log \frac{\pi(\mathbf{a}|\mathbf{s})}{\tilde{\pi}(\mathbf{a}|\mathbf{s})} d\mathbf{a} d\mathbf{s} + \mathrm{E}_{p^\pi(\mathbf{o})}\left[\mathrm{KL}\left(p^\pi(\boldsymbol{\tau}|\mathbf{o})||\tilde{p}(\boldsymbol{\tau}|\mathbf{o})\right)\right] \end{aligned} \tag{4.21}$$

$$\begin{aligned} &= \max_\pi H(\pi) + \int_{\boldsymbol{\tau},\mathbf{o}} p^\pi(\boldsymbol{\tau})p(\mathbf{o}|\boldsymbol{\tau}) \log \frac{q(\mathbf{o})}{\tilde{p}(\mathbf{o})} d\boldsymbol{\tau} d\mathbf{o} \\ &\quad + (1-\gamma)\sum_t \gamma^t \int_{\mathbf{s}} p_t^\pi(\mathbf{s}) \int_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \log \tilde{\pi}(\mathbf{a}|\mathbf{s}) d\mathbf{a} d\mathbf{s} + \mathrm{E}_{p^\pi(\mathbf{o})}\left[\mathrm{KL}\left(p^\pi(\boldsymbol{\tau}|\mathbf{o})||\tilde{p}(\boldsymbol{\tau}|\mathbf{o})\right)\right]. \end{aligned} \tag{4.22}$$

Here $H(\pi)$ denotes the discounted causal entropy of policy π, which is defined as

$$H(\pi) = -(1-\gamma)\sum_{t=0}^{\infty} \gamma^t \int_{\mathbf{s}} p_t^\pi(\mathbf{s}) \int_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \log \pi(\mathbf{a}|\mathbf{s}) d\mathbf{a} d\mathbf{s}.$$

As the last term of Eq. 4.22 is an expected KL divergence and thus non-negative, we can omit it to obtain the lower bound

$$\mathcal{L} = H(\pi) + \int_{\mathbf{o}} p^{\pi}(\mathbf{o}) \log \frac{q(\mathbf{o})}{\tilde{p}(\mathbf{o})} d\mathbf{o} + (1-\gamma) \sum_{t} \gamma^{t} \int_{\mathbf{s}} p_{t}^{\pi}(\mathbf{s}) \int_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s}) \log \tilde{\pi}(\mathbf{a}|\mathbf{s}) d\mathbf{a} d\mathbf{s}. \quad (4.23)$$

For step-based, noiseless observations of the states and action, this corresponds to an entropy-regularized reinforcement learning problem

$$\max_{\pi} J_{\text{NAIL},\tilde{\pi}}(\pi) = \max_{\pi} \int_{\mathbf{s},\mathbf{a}} p^{\pi}(\mathbf{s},\mathbf{a}) \Big(\underbrace{\log \frac{q(\mathbf{s},\mathbf{a})}{p^{\tilde{\pi}}(\mathbf{s},\mathbf{a})} + \log \tilde{\pi}(\mathbf{a}|\mathbf{s})}_{r_{\text{lb}}^{\tilde{\pi}}(\mathbf{s},\mathbf{a})} - \log \pi(\mathbf{a}|\mathbf{s}) \Big) d\mathbf{s} d\mathbf{a}. \quad (4.24)$$

In the following we will only consider this step based setting. However, the analysis presented in this section (including Theorem 1) can be straightforwardly extended to general observations.

Lemma 1. *Let $r_{lb}^{\tilde{\pi}}(\mathbf{s},\mathbf{a}) = \log\left(\frac{q(\mathbf{s},\mathbf{a})}{p^{\tilde{\pi}}(\mathbf{s},\mathbf{a})}\right) + \log \tilde{\pi}(\mathbf{a}|\mathbf{s})$ denote the lower bound reward function for policy $\tilde{\pi}$. Any policy π that improves on the lower bound objective (Eq. 4.24) compared to $\tilde{\pi}$, also decreases the reverse Kullback-Leibler divergence to the expert distribution, that is,*

$$J_{NAIL,\tilde{\pi}}(\pi) > J_{NAIL,\tilde{\pi}}(\tilde{\pi}) \implies D_{RKL}(p^{\pi}(\mathbf{s},\mathbf{a})||q(\mathbf{s},\mathbf{a})) < D_{RKL}(p^{\tilde{\pi}}(\mathbf{s},\mathbf{a})||q(\mathbf{s},\mathbf{a})).$$

Proof. See Appendix C.3. The proof is similar to the proof for expectation-maximization presented by Bishop [2006]. Namely, since the lower bound $J_{\text{NAIL},\tilde{\pi}}(\pi)$ is tight for the policy $\tilde{\pi}$, improving on the lower bound also increases the expected KL term that had been omitted for deriving the lower bound. □

Comparing the policy objective for the non-adversarial formulation (Eq. 4.24) with the adversarial objective for the reverse KL (following Eq. C.3 and Eq. 4.9),

$$\max_{\pi} J_{\text{AIL-RKL}}(\pi) = \int_{\mathbf{s},\mathbf{a}} p^{\pi}(\mathbf{s},\mathbf{a}) \underbrace{\log \frac{q(\mathbf{s},\mathbf{a})}{p^{\pi}(\mathbf{s},\mathbf{a})}}_{r_{\text{adv}}(\mathbf{s},\mathbf{a})} d\mathbf{s} d\mathbf{a},$$

we can identify the following differences. Due to the last term in Eq. 4.24, the reinforcement learning problem turned into an entropy regularized reinforcement learning problem. Furthermore, the lower bound reward function $r_{\text{lb}}^{\tilde{\pi}}(\mathbf{s},\mathbf{a})$ contains an additional term, namely, $\log \tilde{\pi}(\mathbf{a}|\mathbf{s})$. Together, these changes correspond to an additional KL objective that penalizes deviations from the policy $\tilde{\pi}$ that was used for training the discriminator.

Furthermore, the density-ratio in the lower bound reward (and consequently also the lower bound reward itself) explicitly depends on the auxiliary policy $\tilde{\pi}$ and not on the policy π that is optimized. This key difference to the adversarial formulation, enables us to greedily optimize the policy with respect to the lower bound reward function $r_{\text{lb}}^{\tilde{\pi}}(\mathbf{s}, \mathbf{a})$ and ensures that, upon convergence, the lower bound reward function solves the inverse reinforcement learning problem.

Our upper bound on the reverse KL divergence is similar to bounds that have been previously applied for learning hierarchical models for variational inference [Agakov and Barber, 2004; Arenz et al., 2018; Maaløe et al., 2016; Ranganath et al., 2016; Tran et al., 2016] and I-projection-based density estimation [Becker et al., 2020]. Our derivations are most related to the work by Becker et al. [2020] since we only assume samples from the target distribution $q(\mathbf{o})$. However, in contrast to Becker et al. [2020], we consider time-series data and do not use the bound for learning a hierarchical model, but for learning a reward function that is independent of the current policy.

4.2.2. Iteratively Tightening the Bound

Based on Lemma 1 we propose a framework for non-adversarial imitation learning that is sketched by Algorithm 4. Theorem 1 shows that Algorithm 4 solves the imitation learning problem when assuming that the density-ratio estimator (discriminator) is optimal. Although this assumption is very strong, it is also typical for adversarial methods [Goodfellow et al., 2014; Nowozin et al., 2016]. An important difference to the adversarial formulation stems from the fact, that we can show convergence for any policy improvement, that is, we do not require "sufficiently small" generator updates. However, the algorithmic differences compared to the adversarial formulation are quite modest: the generator update of NAIL has an additional reward term $\log \tilde{\pi}(\mathbf{a}|\mathbf{s})$ and solves an entropy-regularized reinforcement learning problem. Together these changes correspond to an additional KL-penalty, $\mathrm{KL}\left(p^{\pi}(\boldsymbol{\tau})||\tilde{p}(\boldsymbol{\tau})\right)$ (see Eq. 4.21), that penalizes large deviations from the policy $\tilde{\pi}$ that was used for training the discriminator. Due to the similarity of both formulations, it can be difficult to show significant differences between the adversarial and the non-adversarial approach in practice. However, we will show in the next section that the non-adversarial formulation can help in better understanding existing (adversarial) methods and that it can be used to derive novel algorithms.

Theorem 1. *When ignoring approximation errors of the density-ratio estimator, Algorithm 4 converges to a stationary point of the reverse KL imitation learning objective.*

Proof. See Appendix C.4 □

Algorithm 4 Non-Adversarial Imitation Learning

Require: Expert demonstrations $\mathcal{D} = \{(\mathbf{s}_i, \mathbf{a}_i)\}_{i=1\dots N}$
Require: Initial policy $\tilde{\pi}$
function NAIL
  repeat
    $\tilde{\phi}(\mathbf{s}, \mathbf{a}) =$DensityRatioEstimator$(\tilde{\pi}, \mathcal{D})$ ▷ Estimate $\frac{q(\mathbf{s},\mathbf{a})}{p^{\tilde{\pi}}(\mathbf{s},\mathbf{a})}$, e.g. using samples from $p^{\tilde{\pi}}$
    $\tilde{\pi} \leftarrow$ PolicyImprovement$(\tilde{\pi}, \tilde{\phi}(\mathbf{s}, \mathbf{a}))$ ▷ improve on $J_{\text{NAIL},\tilde{\pi}}$ using $\tilde{\phi}$ to estimate $r^{\tilde{\pi}}_{\text{lb}}$
  until converged
end function

4.3. Instances of Non-Adversarial Imitation Learning

We will discuss two instances of non-adversarial imitation learning. In Section 4.3.1, we will focus on an existing method, AIRL, and show that it is an instance of non-adversarial imitation learning even though it has been originally formulated as adversarial method. In Section 4.3.2, we present a novel algorithm based on our non-adversarial formulation, which is suitable for offline imitation learning.

4.3.1. Adversarial Inverse Reinforcement Learning

We showed in Section 4.1.4 that AIRL is neither a typical adversarial algorithm nor can it be derived from maximum causal entropy IRL. However, AIRL can be straightforwardly justified as an instance of NAIL, as shown in Corollary 1.

Corollary 1. *Adversarial inverse reinforcement learning is an instance of non-adversarial imitation learning (Algorithm 4).*

Proof. Let $\tilde{\pi}$ denote the current policy during the discriminator update at any given iteration. As shown in Section 4.1.2, the logits of the discriminator trained with the binary cross-entropy loss approximate the log density-ratio $\log \frac{q(\mathbf{s},\mathbf{a})}{p^{\tilde{\pi}}(\mathbf{s},\mathbf{a})}$. Hence, we have

$$\nu(\mathbf{s}, \mathbf{a}) = \bar{\nu}_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) - \log \tilde{\pi}(\mathbf{s}, \mathbf{a}) \approx \log \frac{q(\mathbf{s}, \mathbf{a})}{p^{\tilde{\pi}}(\mathbf{s}, \mathbf{a})}$$

and thus

$$\bar{\nu}_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) \approx \log \frac{q(\mathbf{s}, \mathbf{a})}{p^{\tilde{\pi}}(\mathbf{s}, \mathbf{a})} + \log \tilde{\pi}(\mathbf{s}, \mathbf{a}) = r_{\text{lb}}^{\tilde{\pi}}(\mathbf{s}, \mathbf{a}).$$

The policy objective $J_{\text{AIRL-maxent}}(\pi)$ (Eq. 4.17) solved by AIRL, thus, corresponds to an approximation of $J_{\text{NAIL},\tilde{\pi}}(\pi)$ (Eq. 4.24) based on the density-ratio estimator $\widetilde{\phi}(\mathbf{s}, \mathbf{a}) = \nu(\mathbf{s}, \mathbf{a})$. Hence, AIRL implements Algorithm 4. □

Intuitively, NAIL solves the inverse reinforcement learning problem because any policy that maximizes the lower bound reward function $r_{\text{lb}}^{\tilde{\pi}}(\mathbf{s}, \mathbf{a})$ matches the expert demonstration at least as good as the learned policy $\tilde{\pi}$. After convergence, when $p^{\tilde{\pi}}(\mathbf{s}, \mathbf{a}) \approx q(\mathbf{s}, \mathbf{a})$, maximizing the lower bound reward function, thus, matches the expert demonstrations. In contrast, the adversarial reward function after convergence $r_{\text{adv}}(\mathbf{s}, \mathbf{a}) = \log \frac{q(\mathbf{s},\mathbf{a})}{p^{\tilde{\pi}}(\mathbf{s},\mathbf{a})} \approx 0$ is specific to the current policy and converges to a constant, which is not suitable for matching the expert demonstrations. However, as noted by Fu et al. [2018], when modeling the expert according to Equation 4.11, the reward function learned by AIRL/NAIL corresponds to the advantage function $A_{\text{expert}}^{\text{soft}}(\mathbf{s}, \mathbf{a}) = \log \pi_{\text{expert}}(\mathbf{a}|\mathbf{s})$ if the expert's distribution can be matched exactly by the agent. To be more specific, we can see from the definition of $r_{\text{lb}}^{\tilde{\pi}}(\mathbf{s}, \mathbf{a})$ that, in general, it additionally contains a correction-term based on the mismatch of the induced state distributions, namely,

$$r_{\text{lb}}^{\tilde{\pi}}(\mathbf{s}, \mathbf{a}) = \log \left(\frac{q(\mathbf{s}, \mathbf{a})}{p^{\tilde{\pi}}(\mathbf{s}, \mathbf{a})} \right) + \log \tilde{\pi}(\mathbf{a}|\mathbf{s}) = \log \left(\frac{q(\mathbf{s})}{p^{\tilde{\pi}}(\mathbf{s})} \right) + \log \pi_{\text{expert}}(\mathbf{a}|\mathbf{s}).$$

As the advantage function strongly depends on the system dynamics, the learned reward function is typically not transferable to different dynamical systems. Hence, Fu et al. [2018] proposed to parameterize the discriminator as

$$\nu(\mathbf{s}, \mathbf{a}, \mathbf{s}') = f_{\boldsymbol{\theta}}(\mathbf{s}) + \gamma g_{\eta}(\mathbf{s}') - g_{\eta}(\mathbf{s}) - \log \pi(\mathbf{a}|\mathbf{s})$$

and showed that $f_{\boldsymbol{\theta}}(\mathbf{s})$ approximates the reward function of the expert, when assuming that the system dynamics are deterministic and that the true reward function does not depend on the actions. Note that this modification is also covered by our derivation which does not make specific assumptions on the parameterization of $\bar{\nu}(\mathbf{s}, \mathbf{a})$.

4.3.2. Offline Non-Adversarial Imitation Learning

We will now use the non-adversarial formulation to derive a novel algorithm for offline imitation learning. Offline imitation learning considers the imitation learning problem,

with the additional restriction that we are not able to interact with the environment during learning. Behavioral cloning (BC) is a common approach for offline imitation learning that frames imitation learning as supervised learning, for example, by maximizing the likelihood of the states and actions, that is,

$$\max_{\pi} \Big[J_{\text{bc}}(\pi) = \mathrm{E}_{\mathbf{s},\mathbf{a}\sim q(\mathbf{s},\mathbf{a})} \left[\log \pi(\mathbf{a}|\mathbf{s}) \right] \Big].$$

However, as observed by Pomerleau [1989] and further examined by Ross and Bagnell [2010] compounding errors can lead to covariate shift, that is, the learned policy may reach states that significantly differ from the states encountered by the expert. As the policy was not trained on these states, it is often not able to recover from such mistakes.

Interlude: ValueDice

Our method is inspired by ValueDice [Kostrikov et al., 2020] which is able to outperform behavioral cloning for offline imitation learning. ValueDice aims to match the expert distribution $q(\mathbf{s}, \mathbf{a})$ by expressing the reverse KL divergence using the Donsker-Varadhan representation [Donsker and Varadhan, 1983], that is,

$$\min_{\pi} \mathrm{KL}(p^{\pi}(\mathbf{s},\mathbf{a})||q(\mathbf{s},\mathbf{a})) = \min_{\pi} \max_{\nu} -\log \mathrm{E}_{\mathbf{s},\mathbf{a}\sim q(\mathbf{s},\mathbf{a})} \left[\exp\left(\nu(\mathbf{s},\mathbf{a})\right) \right] + \mathrm{E}_{\mathbf{s},\mathbf{a}\sim p^{\pi}(\mathbf{s},\mathbf{a})} \left[\nu(\mathbf{s},\mathbf{a}) \right]. \tag{4.25}$$

Similar to the KLIEP loss or the binary cross-entropy loss, the optimal solution of the inner maximization problem is given by

$$\nu^{\star}(\mathbf{s},\mathbf{a}) = \log \frac{p^{\pi}(\mathbf{s},\mathbf{a})}{q(\mathbf{s},\mathbf{a})} + \text{const}, \tag{4.26}$$

which closely relates the saddle point problem given by Equation 4.25 to the previously discussed adversarial methods for minimizing the reverse KL divergence. Indeed, a similar algorithm to ValueDice could be straightforwardly derived from the f-GAN formulation for the reverse KL (Eq. 4.10), which differs from Optimization Problem 4.25 by not taking the logarithm of the first expectation.

Optimization Problem 4.25 can not directly be used for offline imitation learning since we can not get samples from the agent's state-action distribution $p^{\pi}(\mathbf{s}, \mathbf{a})$ to approximate the second expectation. Hence, Kostrikov et al. [2020] applied a trick for offline density-ratio estimation [Nachum et al., 2019] by using a change of variables to express $\nu(\mathbf{s}, \mathbf{a})$ in terms of the Q-Function Q^{π}_{ν} for policy π where the reward function is given by $\nu(\mathbf{s}, \mathbf{a})$, that is

$$\nu(\mathbf{s},\mathbf{a}) = Q^{\pi}_{\nu}(\mathbf{s},\mathbf{a}) - \gamma \mathrm{E}_{\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})} \mathrm{E}_{\mathbf{a}'\sim\pi(\mathbf{a}'|\mathbf{s}')} \left[Q^{\pi}_{\nu}(\mathbf{s}',\mathbf{a}') \right]. \tag{4.27}$$

When applying the change of variables (Eq. 4.27) and optimizing Eq. 4.25 with respect to the Q-function, the expected "reward" given by the second expectation can be computed based on the initial state distribution and the Q-function, that is,

$$\begin{aligned}
&\min_{\pi} \max_{Q_\nu^\pi} -\log \mathrm{E}_{\mathbf{s},\mathbf{a}\sim q(\mathbf{s},\mathbf{a})} \left[\exp\left(Q_\nu^\pi(\mathbf{s},\mathbf{a}) - \gamma \mathrm{E}_{\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})} \mathrm{E}_{\mathbf{a}'\sim\pi(\mathbf{a}'|\mathbf{s}')}\left[Q_\nu^\pi(\mathbf{s}',\mathbf{a}')\right]\right)\right] \\
&\qquad\qquad + \mathrm{E}_{\mathbf{s},\mathbf{a}\sim p^\pi(\mathbf{s},\mathbf{a})}\left[Q_\nu^\pi(\mathbf{s},\mathbf{a}) - \gamma \mathrm{E}_{\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})} \mathrm{E}_{\mathbf{a}'\sim\pi(\mathbf{a}'|\mathbf{s}')}\left[Q_\nu^\pi(\mathbf{s}',\mathbf{a}')\right]\right] \\
&= \min_{\pi} \max_{Q_\nu^\pi} -\log \mathrm{E}_{\mathbf{s},\mathbf{a}\sim q(\mathbf{s},\mathbf{a})} \left[\exp\left(Q_\nu^\pi(\mathbf{s},\mathbf{a}) - \gamma \mathrm{E}_{\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})} \mathrm{E}_{\mathbf{a}'\sim\pi(\mathbf{a}'|\mathbf{s}')}\left[Q_\nu^\pi(\mathbf{s}',\mathbf{a}')\right]\right)\right] \\
&\qquad\qquad + (1-\gamma)\mathrm{E}_{\mathbf{s}_0\sim p_0(\mathbf{s})} \mathrm{E}_{\mathbf{a}_0\sim\pi(\mathbf{a}_0|\mathbf{s}_0)}\left[Q_\nu^\pi(\mathbf{s}_0,\mathbf{a}_0)\right].
\end{aligned} \tag{4.28}$$

As shown by Kostrikov et al. [2020], a (biased) Monte-Carlo estimate of Eq. 4.28 can be optimized based on triplets $(\mathbf{s},\mathbf{a},\mathbf{s}')$ sampled from the expert demonstrations, initial states $\mathbf{s}_0$ sampled from the initial state distribution and actions $\mathbf{a}'$ and $\mathbf{a}_0$ sampled from the policy. Compared to the other adversarial methods discussed earlier, ValueDice differs mainly by directly learning the Q-function for the adversarial reward $r_{\text{adv}}(\mathbf{s},\mathbf{a}) = \log\left(\frac{q(\mathbf{s},\mathbf{a})}{p^\pi(\mathbf{s},\mathbf{a})}\right)$ which does not only make it applicable to the offline setting, but—conveniently—also greatly simplifies the reinforcement learning problem. However, common to prior methods for adversarial imitation learning, ValueDice involves solving a saddle point problem (Eq. 4.28) which may require careful tuning of the step size for the policy updates to avoid too greedy updates.

ONAIL

Instead of framing distribution matching as a saddle-point problem, we use the non-adversarial formulation to derive a (soft) actor critic algorithm. That is, we alternate between a critic update, where we learn the soft Q-function $Q_{\text{lb}}^{\text{soft},\,\pi^{(i)}}$ for the current policy $\pi^{(i)}$ and lower bound reward function $r_{\text{lb}}^{\pi^{(i)}}(\mathbf{s},\mathbf{a})$, and an actor update where we use the soft Q-function to find a policy $\pi^{(i+1)}$ that improves on the lower bound objective $J_{\text{NAIL},\pi^{(i)}}$ (Eq. 4.24).

The soft Q-function for the lower bound reward function $r_{\text{lb}}^{\pi^{(i)}}(\mathbf{s},\mathbf{a})$ can be learned analogously to the (standard) Q-function for the adversarial reward function by applying a change of variables to the KLIEP-loss (Eq. 4.10) or Donsker-Varadhan-loss (Eq. 4.25). However, as shown by Lemma 2, the soft Q-function for the lower bound reward can also be directly computed from the Q-function of the adversarial reward.

Lemma 2. *Let $Q_r^{\tilde{\pi}}$ be the Q-function for policy $\tilde{\pi}$ and a given reward function r. Further, let $Q_{r_{lb}}^{soft,\tilde{\pi}}$ be the soft Q-function for policy $\tilde{\pi}$ and reward function $r_{lb}(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \log\tilde{\pi}(\mathbf{a}|\mathbf{s})$. Then, $Q_{r_{lb}}^{soft,\tilde{\pi}}$ can be expressed in terms of $Q_r^{\tilde{\pi}}$ as follows:*

$$Q_{r_{lb}}^{soft,\tilde{\pi}}(\mathbf{s},\mathbf{a}) = Q_r^{\tilde{\pi}}(\mathbf{s},\mathbf{a}) + \log\tilde{\pi}(\mathbf{a}|\mathbf{s}).$$

Proof. see Appendix C.5. □

Lemma 2 might seem surprising at first because in general it is neither straightforward to transform a Q-function to a soft Q-function nor to transform the Q function for one reward function to the Q-function for a different reward function. However, intuitively, the expected negative cross-entropy to come (originating from the change of reward function) and the expected entropy to come (originating from the change to the soft Q-function) cancel out, since both terms are computed for policy $\tilde{\pi}$. Hence, both Q functions only differ by the change of the immediate reward. Lemma 2, thus, enables us to implement the critic update by performing the inner maximization of the saddle point solved by ValueDice 4.28, that is,

$$\begin{aligned} -Q_{r_{\text{adv}}}^{\tilde{\pi}} = \arg\max_{Q_\nu^{\tilde{\pi}}} &- \log \mathrm{E}_{\mathbf{s},\mathbf{a}\sim q(\mathbf{s},\mathbf{a})}\left[\exp\left(Q_\nu^{\tilde{\pi}}(\mathbf{s},\mathbf{a}) - \gamma \mathrm{E}_{\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})}\mathrm{E}_{\mathbf{a}'\sim\tilde{\pi}(\mathbf{a}'|\mathbf{s}')}\left[Q_\nu^{\tilde{\pi}}(\mathbf{s}',\mathbf{a}')\right]\right)\right] \\ &+ (1-\gamma)\mathrm{E}_{\mathbf{s}_0\sim p_0(\mathbf{s})}\mathrm{E}_{\mathbf{a}_0\sim\tilde{\pi}(\mathbf{a}_0|\mathbf{s}_0)}\left[Q_\nu^{\tilde{\pi}}(\mathbf{s}_0,\mathbf{a}_0)\right]. \end{aligned} \tag{4.29}$$

Please note, that we changed the sign for $Q_{r_{\text{adv}}}^{\tilde{\pi}}$ because ν (Eq. 4.26) approximates the adversarial cost function $\nu(\mathbf{s},\mathbf{a}) \approx -r_{\text{adv}}(\mathbf{s},\mathbf{a}) + \text{const}$.

In contrast to ValueDice, we do not need to solve a saddle point problem, since we can replace the policy update of ValueDice (based on Eq. 4.28) with an update based on the loss

$$\begin{aligned} \mathcal{L}_{\text{actor},\pi^{(i)}}(\pi) &= -\mathrm{E}_{\mathbf{s}\sim z(\mathbf{s})}\left[\int_{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s})\left(Q_{\text{adv}}^{\pi^{(i)}} + \log\pi^{(i)}(\mathbf{a}|\mathbf{s}) - \log\pi(\mathbf{a}|\mathbf{s})\right) d\mathbf{a}\right] \\ &= -\mathrm{E}_{\mathbf{s}\sim z(\mathbf{s})}\left[\mathcal{L}_{\text{actor},\pi^{(i)}}(\pi,\mathbf{s})\right], \end{aligned} \tag{4.30}$$

which can be optimized using the reparameterization trick [Kingma and Welling, 2014; Rezende et al., 2014]. The state distribution $z(\mathbf{s})$ used for training the policy should ideally have full support on every state that is encountered by the new policy π. In our experiments we found it sufficient to only use the states encountered during the expert demonstrations, that is $z(\mathbf{s}) = q(\mathbf{s})$. However, it would also be possible to use artificial states, e.g., by adding noise to the expert demonstrations, or to build a replay buffer by

Algorithm 5 Offline Non-Adversarial Imitation Learning

Require: Expert demonstrations $\mathcal{D} = \{(\mathbf{s}_i, \mathbf{a}_i)\}_{i=1...N}$
Require: Initial policy $\tilde{\pi}$
function ONAIL
  repeat
    $Q^{\tilde{\pi}}_{r_{\text{adv}}} \leftarrow \text{IMPLICITDENSITYRATIOESTIMATOR}(\tilde{\pi}, \mathcal{D})$ ▷ approximately solve Optimization Problem 4.29
    $\tilde{\pi} \leftarrow \text{POLICYIMPROVEMENT}(\tilde{\pi}, Q^{\tilde{\pi}}_{r_{\text{adv}}})$ ▷ improve on $J_{\text{NAIL},\tilde{\pi}}$ by decreasing loss $\mathcal{L}_{\text{actor},\tilde{\pi}}(\pi)$ (Eq. 4.30)
  until converged
end function

rolling out the policy after each iteration (which would leave the offline-regime). The conceptual difference between O-NAIL and ValueDice enables us to show convergence even for large improvement on the loss (Eq. 4.30) based on Lemma 3.

Lemma 3. *When ignoring approximation errors of the Q-function $Q^{\tilde{\pi}}_{r_{adv}}$, any policy π that achieves lower or equal loss $\mathcal{L}_{actor,\pi^{(i)}}(\pi, \mathbf{s})$ (Eq. 4.30) than $\pi^{(i)}$ on any state $\mathbf{s}$ also improves on $\pi^{(i)}$ with respect to the lower bound objective $J_{NAIL,\pi^{(i)}}(\pi)$ (Eq. 4.24), that is,*

$$\forall_{\mathbf{s}} : \mathcal{L}_{actor,\pi^{(i)}}(\pi, \mathbf{s}) \leq \mathcal{L}_{actor,\pi^{(i)}}(\pi^{(i)}, \mathbf{s}) \implies J_{NAIL,\pi^{(i)}}(\pi) \geq J_{NAIL,\pi^{(i)}}(\pi^{(i)}).$$

Proof. See Appendix C.6. □

The resulting algorithm (Alg. 5) is, thus, an instance of non-adversarial imitation learning based on the implicit density-ratio estimator

$$\widetilde{\phi}(\mathbf{s}, \mathbf{a}) = \exp\left(Q^{\tilde{\pi}}_{r_{\text{adv}}}(\mathbf{s}, \mathbf{a}) - \gamma \mathrm{E}_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})} \mathrm{E}_{\mathbf{a}' \sim \pi(\mathbf{a}'|\mathbf{s}')} \left[Q^{\tilde{\pi}}_{r_{\text{adv}}}(\mathbf{s}', \mathbf{a}')\right]\right).$$

Theorem 2. *When ignoring approximation errors of the Q-function $Q^{\tilde{\pi}}_{r_{adv}}$, Algorithm 5 converges to a stationary point of the reverse KL imitation learning objective.*

Proof. Based on Lemma 3, the theorem can be derived analogously to Theorem 1. □

4.4. Experiments

We compared the policy updates of O-NAIL and ValueDice on the Mujoco [Todorov et al., 2012] *Hopper*, *Walker2d*, *HalfCheetah* and *Ant* experiment. We obtained 50 expert

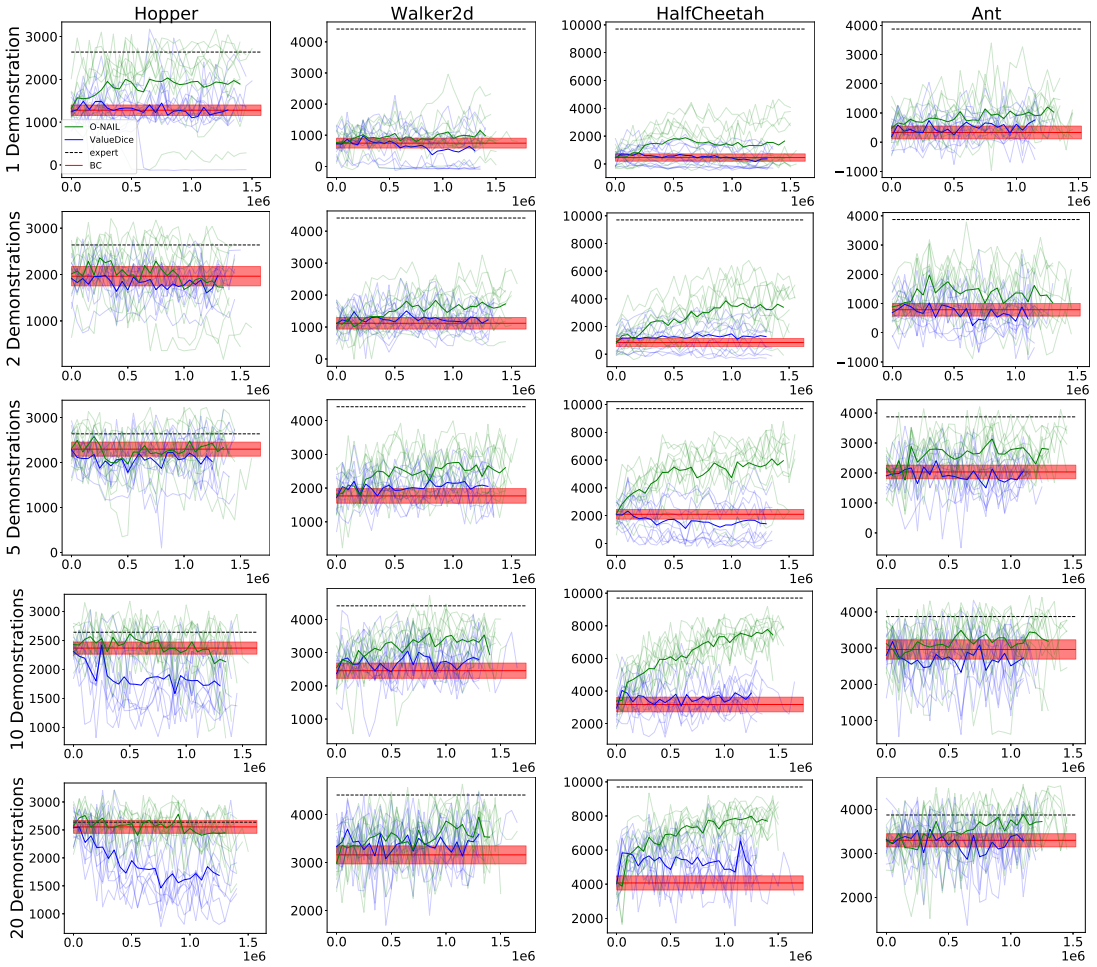


Figure 4.1.: The plots show the approximated return (average of ten roll-outs) of the stochastic policy for ten trials for O-NAIL and ValueDice for different numbers of demonstrations and environments. For behavioral cloning, the shaded region corresponds to the 0.95 confidence region based on the empirical standard error of the mean (when Gaussianity is assumed).

HYPER-PARAMETER	VALUE (VALUEDICE)	VALUE (O-NAIL)
η_π	1×10^{-5}	1×10^{-4}
η_Q	1×10^{-3}	1×10^{-3}
N_π	1	$10\,000$
N_Q	5	1000

Table 4.1.: The table shows the hyper-parameters used by O-NAIL and ValueDice for the *HalfCheetah* experiment. O-NAIL performed better for large number of gradient steps, whereas ValueDice performed best for few gradient updates before switching policy updates and Q-function updates.

demonstrations—each corresponding to one trajectory consisting of thousand steps—from a policy that was trained using soft actor-critic [SAC, Haarnoja et al., 2018]. We initialized the policy using behavioral cloning. More precisely, we used ten per cent of the training data as validation data for early-stopping, and trained the policy by maximizing the likelihood of the remaining expert data. Both algorithms are evaluated based on the same implementation[1] and differ only due to the different policy loss and the chosen hyper-parameters. For ValueDice, the policy updates are performed based on a mini-batch approximation of the saddle point problem given by Equation 4.28. For O-NAIL, we use a mini-batch approximation of the I-projection-loss (Eq. 4.30). Both algorithms use a mini-batch size of 256. The Q-function and policy are each represented by a neural networks with two hidden layers of 256 units. The policy network outputs the parameters of a Gaussian distribution (with a diagonal state-dependent covariance matrix) and the sampled action are squashed by a $\tanh$ to respect action bounds, as discussed by Haarnoja et al. [2018].

We tuned the learning rates η_π and η_Q for the policy and Q-function as well as the number of gradient steps N_π and N_Q for the policy optimization and Q-optimization. We performed a grid-search on these hyper-parameters and present the chosen hyper-parameters in Table 4.1.

The results of our experiments are shown in Figure 4.1. Although O-NAIL seems to clearly outperform ValueDice on these experiment, we want to stress, that the purpose of our evaluation was not to show practical advantages, but rather to demonstrate that we can derive novel algorithms based on our non-adversarial formulation that achieve similar performance compared to adversarial methods, while working at a significantly different modus operandi—namely, by using three to four orders of magnitude more update step

[1]The code can be downloaded from `https://www.github.com/OlegArenz/O-NAIL`.

for the Q-function and policy. Regarding the performance in practice, we want to point out the following shortcomings of our evaluations:

- **Lack of regularization.** We did not perform any regularization on the Q-function or policy. It is well-known that adversarial methods sometimes can significantly benefit from regularizing the discriminator (or Q-function for ValueDice). For example, ValueDice applied a gradient penalty similar to the one that was introduced for Wasserstein-GANs [Gulrajani et al., 2017] on the Q-function. We actually tried this gradient penalty also on our experiment and were not able to show a benefit. Still, although Kostrikov et al. [2020] focused on the online setting, they also performed experiments in the offline setting, where they could show that ValueDice can outperform behavioral cloning. We believe that by introducing some type of regularization, both algorithms could potentially also achieve significantly better performance on our implementation. However, since these improvements are often achieved by introducing inductive biases, which can widen the gap between theory and practice, we have limited the evaluation to a simple implementation of the respective algorithms.

- **Initialization.** We initialized the policies using behavioral cloning. When performing the same experiment with randomly initialized policies, ValueDice would perform similarly, while O-NAIL would fail to learn at all with the chosen hyperparameters. The lack of convergence of O-NAIL is to be expected because using many discriminator updates to estimate a density-ratio between distributions with non-overlapping support typically results in sharp decision-boundaries that do not possess meaningful gradients or maxima. However, when performing only few Q-function-updates between optimizing the policy—which corresponds to a strong type of early-stopping—, such problems can be mitigated. Rather than relying on strong regularization that further disconnect theory and practice, we applied behavioral cloning to ensure overlapping support, which also seems reasonable and applicable in practice.

4.5. Discussion

Many modern methods in imitation learning and inverse reinforcement learning are based on an adversarial formulation. These methods frame the problem of distribution-matching as a minimax game between a policy and a discriminator, and rely on small policy updates for showing convergence to a Nash equilibrium. In contrast to these methods, we formulate distribution-matching as an iterative lower-bound optimization by alternating between

maximizing and tightening a bound on the reverse KL divergence. This non-adversarial formulation enables us to drop the requirement of "sufficiently small" policy updates for proving convergence. Algorithmically, our non-adversarial formulation is very similar to previous adversarial formulations and differs only due to an additional reward term that penalizes deviations from the previous policy.

4.5.1. Limitations and Future Work

As the resulting algorithms are very similar to their adversarial counterparts, it can be difficult to show significant differences in practice. Hence, in this work, we focused on the insights gained from the non-adversarial formulation. For example, we showed that adversarial inverse reinforcement learning, which was previously not well understood, can be straightforwardly derived from our non-adversarial formulation. However, eventually we would like to derive stronger practical advantages from our formulation. We demonstrated that the non-adversarial formulation can be used to derive novel algorithms by presenting O-NAIL, an actor-critic based offline imitation learning method and our comparisons with ValueDice suggest that the non-adversarial formulation may indeed be beneficial. However, we hope to further distinguish O-NAIL from prior work by building on the close connection between non-adversarial imitation learning and inverse reinforcement learning in order to learn generalizable reward functions offline.

Adversarial methods have been suggested for a variety of different divergences, including [Ghasemipour et al., 2020] but not limited to [Xiao et al., 2019] the family of f-divergences. The non-adversarial formulation is currently limited to the reverse KL divergence and penalizes deviations from the previous policy based on the reverse KL divergence. It is an open question, whether our lower bound can be generalized to other divergences, for example, when penalizing deviations based on different divergences.

5. Conclusion and Future Work

We investigated the I-projection in different problem settings that are shown in Table 5.1. For variational inference and density estimation we can evaluate the probability density of the variational model ("explicit model"). For reinforcement learning and imitation learning we act in an MDP and can only sample from the learned behavior using policy roll-outs but can not evaluate its probability density which depends on complex and often unknown interactions with the system dynamics ("implicit model"). For variational inference and reinforcement learning[1] the (unnormalized) target distribution is given ("explicit target"), whereas for density estimation and imitation learning, we only have access to samples ("implicit target"). For variational inference and density estimation we can also obtain samples from the model, and, thus these problem settings can be considered simpler than the respective formulations in a Markov decision process. However, this simplification in the problem formulation is put into perspective by the fact that we have higher demands on the model, namely, we aim to approximate all the modes in the target distribution. The unnormalized target distributions in variational inference and reinforcement learning can not be sampled straightforwardly and these problem formulation are, hence, not necessarily simpler than their counterparts with implicit targets.

[1] Here, by reinforcement learning we refer to the special case, where we have an additional entropy objective on the *behavior* (not just on the policy). As discussed in Section 1.2.2, the reward function then corresponds to an unnormalized target distribution.

	Explicit Model	Implicit Model (MDP)
Explicit Target	variational inference (Ch. 3)	reinforcement learning (Ch. 2)
Implicit Target	density estimation (Sec. 3.6.2)	imitation learning (Ch. 4)

Table 5.1.: The table shows the different problem settings discussed within this thesis and refers to the corresponding chapter or section. The problem settings differ depending on whether we can evaluate the (unnormalized) target distribution or the model or whether we only have access to samples.

The reinforcement learning setting was discussed in Chapter 2, where we assumed that a target distribution is given instead of a reward function. By also regularizing the entropy of the policy, we could show that the optimal policy can be found by maximizing a reward function that is given by the scaled log density-ratio between the target distribution and the distribution induced by the optimal policy. We exploited this circular dependency using a fixed-point iteration in order to efficiently learn policy and reward function. By using density estimation for learning the target distribution from demonstrations, this approach was also used for inverse reinforcement learning.

In Chapter 3, we discussed our method for variational inference, where we used an upper bound on the reverse KL for latent variable models in order to learn GMM approximations of the target distribution. By using insights from policy search and by dynamically adapting the number of components, we could show that our variational approximations can obtain similar sample quality as expensive Markov chain Monte-Carlo on a large number of test problems. In Section 3.6.2, we also briefly discussed the follow-up by Becker et al. [2020], which used the same upper bound for density estimation.

We built on our formulation for density estimation [Becker et al., 2020] and extended it in order to apply the upper bound also for imitation learning, which we discussed in Chapter 4. We showed that we can obtain a non-adversarial formulation for imitation learning and inverse reinforcement learning that resembles current adversarial methods. In particular, we showed that adversarial inverse reinforcement learning [Fu et al., 2018] is actually a non-adversarial method that can be derived based on our formulation. We also derived a novel non-adversarial method for offline imitation learning.

5.1. Future Work

Depending on the respective problem formulation, we developed different algorithms that can be improved and extended individually, which we already discussed in the corresponding chapters. Here, we will mainly discuss potential synergies between the different methods.

We successfully applied the same upper bound to three of the four problem settings shown in Table 5.1, namely for variational inference [Arenz et al., 2018, 2020], density estimation [Becker et al., 2020] and imitation learning [Arenz and Neumann, 2020]. It is therefore natural to also consider the bound for the remaining problem setting, reinforcement learning. It could be possible to apply our decomposition to simplify the training of latent-variable policies—for example GMMs—in order to learn multimodal or hierarchical [Sutton et al., 1999] policies. We already performed first steps towards this goal by learning GMM policies for robot teleoperation [Ewerton et al., 2020], however,

here, we only considered episodic policy search.

The motivation for using the upper bound in variational inference and in imitation learning are slightly different; in variational inference it allowed us to reduce the problem of optimizing a Gaussian mixture model to the problem of optimizing individual Gaussian components, whereas in imitation learning it allowed us to reduce the I-projection problem—which includes the intractable entropy of the behavior—to a more standard reinforcement learning problem that only includes the entropy of the policy. It would be interesting to apply the upper bound in both ways in order to perform both hierarchical and non-adversarial imitation learning. As discussed in Chapter 4, the advantage function and thus the log-density of the policy can also be treated as a maximally entangled reward function, which can be learned by using our non-adversarial formulation. We could exploit this connection between the policy and reward function, by treating the state transitions in the Semi-MDP as non-Markovian reward function which we would expect to be less entangled with the system dynamics than the lower-level policies.

6. Contribution Statements

Although this thesis is a summary of my scientific work, it would not have been possible without the involvement of my collaborators. In this Section, we will disentangle the individual contributions based on the publications that underlie Chapters 2, 3 and 4.

6.1. Contributions for Chapter 2

I derived the concrete algorithm and implemented most parts of it. I also performed the evaluation and wrote most parts of the resulting article [Arenz et al., 2016]. Hany Abdulsamad suggested alternate options for optimizing the dual problem, contributed small parts to the implementation and provided feedback on the draft. Gerhard Neumann suggested the initial problem formulation and also sketched the derivations. He also assisted in identifying problems in the implementation, suggested experiments and wrote small parts of the article.

6.2. Contributions for Chapter 3 (disregarding Section 3.6)

I proposed most algorithmic choices, for example those discussed in Section 3.2.2, 3.2.3 and 3.2.4. I also implemented the algorithm (except for code related to the Goodwin oscillator), performed the evaluation and wrote the article [Arenz et al., 2020]. Mingjun Zhong implemented the Goodwin oscillator, suggested MCMC competitors for the evaluation and provided feedback on the draft. Gerhard Neumann (re-)discovered the lower bound and suggested using it in combination with MORE for learning GMM approximations. He also advised me during the development of the concrete algorithm and provided feedback on the draft.

6.3. Contributions for Section 3.6.1

Marco Ewerton led the research and proposed several algorithmic choices, implemented the parts related to the simulation environments and probabilistic movement primitives (ProMPs), conducted the experiments and user studies and wrote most parts of the article [Ewerton et al., 2020]. I suggested to use VIPS for learning a mixture of ProMPs, proposed some algorithmic choices, implemented parts related to VIPS and robot control, assisted during the robot experiments and wrote small parts of the paper. Jan Peters provided advice during the development of the approach and provided feedback on the draft.

6.4. Contributions for Section 3.6.2

Philipp Becker proposed several algorithmic choices, performed the derivations, implemented the algorithm, conducted the experiments and wrote the article [Becker et al., 2020]. I gave advice for the implementation, made few suggestions regarding experiments and provided feedback on the draft. Gerhard Neumann suggested applying the lower bound for density estimation and to extend VIPS to conditional latent variable models and sketched the derivations. He also provided advice during the development of the approach and revised the draft.

6.5. Contributions for Chapter 4

I performed the derivations for the adversarial and non-adversarial formulation in the observation-based, trajectory-centric setting and showed that Markovian reward functions can be obtained under additional assumptions. I also identified the relation to AIRL. I proposed to apply the lower bound for offline imitation learning and derived the resulting algorithm (O-NAIL). I also performed the implementation and evaluation and wrote the article [Arenz and Neumann, 2020]. Gerhard Neumann suggested applying the lower bound for imitation learning and sketched the derivation. He also provided advice during the development of the approach and provided feedback on the draft.

A. Appendix for Chapter 2

A.1. Full Specification of the Optimization Problem and its Derivations

$$
\begin{aligned}
\underset{\pi_t(\boldsymbol{a}|\boldsymbol{s})}{\text{maximize}} \quad & -\sum_{t=1}^{T-1} \int_{\boldsymbol{s}} p_t(\boldsymbol{s}) \int_{\boldsymbol{a}} \pi_t(\boldsymbol{a}|\boldsymbol{s}) \log \pi_t(\boldsymbol{a}|\boldsymbol{s}) d\boldsymbol{a} d\boldsymbol{s} - \sum_{t=2}^{T} \beta_t \int_{\boldsymbol{y}} p_t(\boldsymbol{y}) \log \frac{p_t(\boldsymbol{y})}{q_t(\boldsymbol{y})} d\boldsymbol{y} \\
\text{subject to} \quad & \forall_{t<T} \int_{\boldsymbol{s}} p_t(\boldsymbol{s}) \int_{\boldsymbol{a}} \pi_t(\boldsymbol{a}|\boldsymbol{s}) \log \frac{\pi_t(\boldsymbol{a}|\boldsymbol{s})}{\pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s})} d\boldsymbol{a} \leq \epsilon_t, \\
& \forall_{\boldsymbol{s}}\ p_1(\boldsymbol{s}) = \boldsymbol{\mu}_1(\boldsymbol{s}) \\
& \forall_{t>1} \forall_{\boldsymbol{s}'}\ p_t(\boldsymbol{s}') = \int_{\boldsymbol{s},\boldsymbol{a}} p_{t-1}(\boldsymbol{s}) \pi_{t-1}(\boldsymbol{a}|\boldsymbol{s}) p_{t-1}(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a}) d\boldsymbol{s} d\boldsymbol{a}, \\
& \forall_{2<t<T} \forall_{\boldsymbol{y}}\ p_t(\boldsymbol{y}) = \int_{\boldsymbol{s}} p_t(\boldsymbol{s}) \int_{\boldsymbol{a}} \pi_t(\boldsymbol{a}|\boldsymbol{s}) p_t(\boldsymbol{y}|\boldsymbol{s},\boldsymbol{a}) d\boldsymbol{a} d\boldsymbol{s}, \\
& \forall_{\boldsymbol{y}}\ p_T(\boldsymbol{y}) = \int_{\boldsymbol{s}} p_T(\boldsymbol{s}) p_T(\boldsymbol{y}|\boldsymbol{s}) d\boldsymbol{s}, \\
& \forall_{t<T} \forall_{\boldsymbol{s}} \int_{a} \pi_t(\boldsymbol{a}|\boldsymbol{s}) d\boldsymbol{a} = 1, \\
& \forall_{t>2} \int_{\boldsymbol{y}} p_t(\boldsymbol{y}) d\boldsymbol{y} = 1.
\end{aligned}
$$

Lagrangian:

$$
\begin{aligned}
\mathcal{L} = &- \sum_{t=1}^{T-1} \int_{\boldsymbol{s}} p_t(\boldsymbol{s}) \int_{\boldsymbol{a}} \pi_t(\boldsymbol{a}|\boldsymbol{s}) \log \pi_t(\boldsymbol{a}|\boldsymbol{s}) d\boldsymbol{a} d\boldsymbol{s} - \sum_{t=2}^{T} \beta_t \int_{\boldsymbol{y}} p_t(\boldsymbol{y}) \log \frac{p_t(\boldsymbol{y})}{q_t(\boldsymbol{y})} d\boldsymbol{y} \\
&+ \sum_{t=1}^{T-1} \alpha_t \left(\epsilon_t - \int_{\boldsymbol{s}} p_t(\boldsymbol{s}) \int_{a} \pi_t(\boldsymbol{a}|\boldsymbol{s}) \log \frac{\pi_t(\boldsymbol{a}|\boldsymbol{s})}{\pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s})} d\boldsymbol{a} d\boldsymbol{s} \right) \\
&+ \int_{\boldsymbol{s}} V_1(\boldsymbol{s}) \left(\mu_1(\boldsymbol{s}) - p_1(\boldsymbol{s}) \right) d\boldsymbol{s} \\
&+ \sum_{t=2}^{T} \int_{\boldsymbol{s}'} V_t(\boldsymbol{s}') \left(\int_{\boldsymbol{s},\boldsymbol{a}} p_{t-1}(\boldsymbol{s}) \pi_{t-1}(\boldsymbol{a}|\boldsymbol{s}) p_{t-1}(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a}) d\boldsymbol{a} d\boldsymbol{s} - p_t(\boldsymbol{s}') \right) d\boldsymbol{s}' \\
&+ \sum_{t=2}^{T-1} \int_{y} \eta_t(\boldsymbol{y}) \left(\int_{\boldsymbol{s}} p_t(\boldsymbol{s}) \int_{\boldsymbol{a}} \pi_t(\boldsymbol{a}|\boldsymbol{s}) p_t(\boldsymbol{y}|\boldsymbol{s},\boldsymbol{a}) d\boldsymbol{a} d\boldsymbol{s} - p_t(\boldsymbol{y}) \right) d\boldsymbol{y} \\
&+ \int_{\boldsymbol{y}} \eta_T(\boldsymbol{y}) \left(\int_{\boldsymbol{s}} p_T(\boldsymbol{s}) p_T(\boldsymbol{y}|\boldsymbol{s}) d\boldsymbol{s} - p_T(\boldsymbol{y}) \right) d\boldsymbol{y} \\
&+ \sum_{t=1}^{T-1} \int_{\boldsymbol{s}} \lambda_{\pi,t}(\boldsymbol{s}) \left(1 - \int_{\boldsymbol{a}} \pi_t(\boldsymbol{a}|\boldsymbol{s}) d\boldsymbol{a} \right) d\boldsymbol{s} + \sum_{t=2}^{T} \lambda_{y,t} \left(\int_{\boldsymbol{y}} p_t(\boldsymbol{y}) d\boldsymbol{y} - 1 \right)
\end{aligned}
$$

The partial derivative of the Lagrangian w.r.t. the policy is given by

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \pi_t(\boldsymbol{a}|\boldsymbol{s})} = &-p_t(\boldsymbol{s}) \left(1 + \log \pi_t(\boldsymbol{a}|\boldsymbol{s}) \right) - \alpha_t p_t(\boldsymbol{s}) \left(1 + \log \pi_t(\boldsymbol{a}|\boldsymbol{s}) - \log \pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s}) \right) \\
&+ \int_{\boldsymbol{s}'} V_{t+1}(\boldsymbol{s}') p_t(\boldsymbol{s}) p_t(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a}) d\boldsymbol{s}' + \mathbb{1}_{t>1} \int_{\boldsymbol{y}} \eta_t(\boldsymbol{y}) p_t(\boldsymbol{s}) p_t(\boldsymbol{y}|\boldsymbol{s},\boldsymbol{a}) \, d\boldsymbol{y} - \lambda_{\pi,t}(\boldsymbol{s}).
\end{aligned} \tag{A.1}
$$

Setting the partial derivative to zero yields

$$
\begin{aligned}
\pi_t^\star(\boldsymbol{a}|\boldsymbol{s}) = &\exp\Big(\frac{1}{1+\alpha_t} \Big(-1 - \alpha_t + \alpha_t \log \pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s}) + \int_{\boldsymbol{s}'} V_{t+1}(\boldsymbol{s}') p_t(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a}) \, d\boldsymbol{s}' \\
&\qquad + \mathbb{1}_{t>1} \int_{\boldsymbol{y}} \eta_t(\boldsymbol{y}) p_t(\boldsymbol{y}|\boldsymbol{s},\boldsymbol{a}) d\boldsymbol{y} - \frac{\lambda_{\pi,t}(\boldsymbol{s})}{p_t(\boldsymbol{s})} \Big) \Big) \\
= &\frac{1}{Z_{\pi,t}} \exp\Big(\frac{1}{1+\alpha_t} \Big(\alpha_t \log \pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s}) \\
&\qquad + \int_{\boldsymbol{s}'} V_{t+1}(\boldsymbol{s}') p_t(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a}) \, d\boldsymbol{s}' + \mathbb{1}_{t>1} \int_{\boldsymbol{y}} \eta_t(\boldsymbol{y}) p_t(\boldsymbol{y}|\boldsymbol{s},\boldsymbol{a}) d\boldsymbol{y} \Big) \Big),
\end{aligned} \tag{A.2}
$$

with partition function

$$Z_{\pi,t} = \exp\left(\frac{1}{1+\alpha_t}\left(-1-\alpha_t-\frac{\lambda_{\pi,t}(\boldsymbol{s})}{p_t(\boldsymbol{s})}\right)\right).$$

By setting the partial derivative $\frac{\partial \mathcal{L}}{\partial \lambda_{\pi,t}} = 1 - \int_{\boldsymbol{a}} \pi_t(\boldsymbol{a}|\boldsymbol{s}) d\boldsymbol{a}$ to zero, we find that inserting the optimal Lagrangian multiplier $\lambda^{\star}_{\pi,t}$ normalizes the policy $\pi_t(\boldsymbol{a}|\boldsymbol{s})$ and hence

$$\begin{aligned} Z_{\pi,t} = \int_{\boldsymbol{a}} \exp\Big(\frac{1}{1+\alpha_t}\Big(\alpha_t \log \pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s}) + \int_{\boldsymbol{s}'} V_{t+1}(\boldsymbol{s}')p_t(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})\, d\boldsymbol{s}' \\ + \mathbb{1}_{t>1} \int_{\boldsymbol{y}} \eta_t(\boldsymbol{y}) p_t(\boldsymbol{y}|\boldsymbol{s},\boldsymbol{a}) d\boldsymbol{y}\Big)\Big) d\boldsymbol{a}. \end{aligned} \tag{A.3}$$

By inserting (A.3) in (A.2) we find that

$$\begin{aligned} \Rightarrow \pi^{\star}_t(\boldsymbol{a}|\boldsymbol{s}) = \exp\Big(\frac{1}{1+\alpha_t}\Big(\alpha_t \log \pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s}) + \int_{\boldsymbol{s}'} V_{t+1}(\boldsymbol{s}')p_t(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})\, d\boldsymbol{s}' \\ + \mathbb{1}_{t>1} \int_{\boldsymbol{y}} \eta_t(\boldsymbol{y}) p_t(\boldsymbol{y}|\boldsymbol{s},\boldsymbol{a}) d\boldsymbol{y}\Big) \\ - \log \int_{\boldsymbol{a}} \exp\Big(\frac{1}{1+\alpha_t}\Big(\alpha_t \log \pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s}) + \int_{\boldsymbol{s}'} V_{t+1}(\boldsymbol{s}')p_t(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})\, d\boldsymbol{s}' \\ + \mathbb{1}_{t>1} \int_{\boldsymbol{y}} \eta_t(\boldsymbol{y}) p_t(\boldsymbol{y}|\boldsymbol{s},\boldsymbol{a}) d\boldsymbol{y}\Big)\Big) d\boldsymbol{a}\Big) \end{aligned} \tag{A.4}$$

Inserting (A.4) into the Lagrangian yields the dual

$$
\begin{aligned}
\mathcal{G} = &- \sum_{t=1}^{T-1} \int_{\boldsymbol{s}} p_t(\boldsymbol{s}) \int_{\boldsymbol{a}} \pi_t(\boldsymbol{a}|\boldsymbol{s}) \Big(\alpha_t \log \pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s}) + \int_{\boldsymbol{s}'} V_{t+1}(\boldsymbol{s}') p_t(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})\, d\boldsymbol{s}' \\
&+ \mathbb{1}_{t>1} \int_{\boldsymbol{y}} \eta_t(\boldsymbol{y}) p_t(\boldsymbol{y}|\boldsymbol{s},\boldsymbol{a}) d\boldsymbol{y} \Big) d\boldsymbol{a} d\boldsymbol{s} \\
&+ \sum_{t=1}^{T-1} (1+\alpha_t) \int_{\boldsymbol{s}} p_t(\boldsymbol{s}) \log \int_{\boldsymbol{a}} \exp\Big(\frac{1}{1+\alpha_t} \Big(\alpha_t \log \pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s}) \\
&+ \int_{\boldsymbol{s}'} V_{t+1}(\boldsymbol{s}') p_t(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})\, d\boldsymbol{s}' + \mathbb{1}_{t>1} \int_{\boldsymbol{y}} \eta_t(\boldsymbol{y}) p_t(\boldsymbol{y}|\boldsymbol{s},\boldsymbol{a}) d\boldsymbol{y} \Big) \Big) d\boldsymbol{a} d\boldsymbol{s} \\
&- \sum_{t=2}^{T} \beta_t \int_{\boldsymbol{y}} p_t(\boldsymbol{y}) \log \frac{p_t(\boldsymbol{y})}{q_t(\boldsymbol{y})} d\boldsymbol{y} + \sum_{t=1}^{T-1} \alpha_t \left(\epsilon_t + \int_{\boldsymbol{s}} p_t(\boldsymbol{s}) \int_{\boldsymbol{a}} \pi_t(\boldsymbol{a}|\boldsymbol{s}) \log \pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s}) d\boldsymbol{a} d\boldsymbol{s} \right) \\
&+ \int_{\boldsymbol{s}} V_1(\boldsymbol{s}) \left(\mu_1(\boldsymbol{s}) - p_1(\boldsymbol{s}) \right) d\boldsymbol{s} \\
&+ \sum_{t=2}^{T} \int_{\boldsymbol{s}'} V_t(\boldsymbol{s}') \left(\int_{\boldsymbol{s},\boldsymbol{a}} p_{t-1}(\boldsymbol{s}) \pi_{t-1}(\boldsymbol{a}|\boldsymbol{s}) p_{t-1}(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a}) d\boldsymbol{a} d\boldsymbol{s} - p_t(\boldsymbol{s}') \right) d\boldsymbol{s}' \\
&+ \sum_{t=2}^{T-1} \int_{y} \eta_t(\boldsymbol{y}) \left(\int_{\boldsymbol{s}} p_t(\boldsymbol{s}) \int_{\boldsymbol{a}} \pi_t(\boldsymbol{a}|\boldsymbol{s}) p_t(\boldsymbol{y}|\boldsymbol{s},\boldsymbol{a}) d\boldsymbol{a} d\boldsymbol{s} - p_t(\boldsymbol{y}) \right) d\boldsymbol{y} \\
&+ \int_{\boldsymbol{y}} \eta_T(\boldsymbol{y}) \left(\int_{\boldsymbol{s}} p_T(\boldsymbol{s}) p_T(\boldsymbol{y}|\boldsymbol{s}) d\boldsymbol{s} - p_T(\boldsymbol{y}) \right) d\boldsymbol{y} \\
&+ \sum_{t=1}^{T-1} \int_{\boldsymbol{s}} \lambda_{\pi,t}(\boldsymbol{s}) \left(1 - \int_{\boldsymbol{a}} \pi_t(\boldsymbol{a}|\boldsymbol{s}) d\boldsymbol{a} \right) d\boldsymbol{s} + \sum_{t=2}^{T} \lambda_{y,t} \left(\int_{\boldsymbol{y}} p_t(\boldsymbol{y}) d\boldsymbol{y} - 1 \right) \\
= &\sum_{t=1}^{T-1} (1+\alpha_t) \int_{\boldsymbol{s}} p_t(\boldsymbol{s}) \log \int_{\boldsymbol{a}} \exp\Big(\frac{1}{1+\alpha_t} \Big(\alpha_t \log \pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s}) \\
&+ \int_{\boldsymbol{s}'} V_{t+1}(\boldsymbol{s}') p_t(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})\, d\boldsymbol{s}' + \mathbb{1}_{t>1} \int_{\boldsymbol{y}} \eta_t(\boldsymbol{y}) p_t(\boldsymbol{y}|\boldsymbol{s},\boldsymbol{a}) d\boldsymbol{y} \Big) \Big) d\boldsymbol{a} d\boldsymbol{s} \\
&- \sum_{t=2}^{T} \beta_t \int_{\boldsymbol{y}} p_t(\boldsymbol{y}) \log \frac{p_t(\boldsymbol{y})}{q_t(\boldsymbol{y})} d\boldsymbol{y} + \sum_{t=1}^{T-1} \alpha_t \epsilon_t + \int_{\boldsymbol{s}} \mu_1(\boldsymbol{s}) V_1(\boldsymbol{s}) d\boldsymbol{s} - \sum_{t=1}^{T} \int_{\boldsymbol{s}} V_t(\boldsymbol{s}) p_t(\boldsymbol{s})\, d\boldsymbol{s} \\
&- \sum_{t=2}^{T} \int_{y} \eta_t(\boldsymbol{y}) p_t(\boldsymbol{y}) d\boldsymbol{y} + \int_{\boldsymbol{y}} \eta_T(\boldsymbol{y}) \int_{\boldsymbol{s}} p_T(\boldsymbol{s}) p_T(\boldsymbol{y}|\boldsymbol{s}) d\boldsymbol{s} d\boldsymbol{y} + \sum_{t=2}^{T} \lambda_{y,t} \left(\int_{\boldsymbol{y}} p_t(\boldsymbol{y}) d\boldsymbol{y} - 1 \right)
\end{aligned}
$$

Gradients of Dual:
We first define

$$r_t(\boldsymbol{s},\boldsymbol{a}) = \begin{cases} \int_{\boldsymbol{y}} p_T(\boldsymbol{y}|\boldsymbol{s})\eta_T(\boldsymbol{y})d\boldsymbol{y} & \text{, if } t = T \\ \int_{\boldsymbol{y}} \eta_t(\boldsymbol{y})p_t(\boldsymbol{y}|\boldsymbol{s},\boldsymbol{a})d\boldsymbol{y} + \alpha_t \log \pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s}) & \text{, if } 1 < t < T \\ \alpha_t \log \pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s}) & \text{, if } t = 1 \end{cases}$$

$$Q_t(\boldsymbol{s},\boldsymbol{a}) = \begin{cases} r_t(\boldsymbol{s},\boldsymbol{a}) & \text{, if } t = T \\ r_t(\boldsymbol{s},\boldsymbol{a}) + \int_{\boldsymbol{s'}} p_t(\boldsymbol{s'}|\boldsymbol{s},\boldsymbol{a})V_{t+1}(\boldsymbol{s'})d\boldsymbol{s'} & \text{, if } t < T \end{cases}$$

The partial derivatives of the dual are then given by:

$$\frac{\partial \mathcal{G}}{p_t(\boldsymbol{s})} = -V_t(\boldsymbol{s}) + (1+\alpha_t)\log \int_a \exp\left(\frac{1}{1+\alpha_t} Q_t(\boldsymbol{s},\boldsymbol{a})\right), \tag{A.5}$$

Setting the derivative w.r.t. the state distribution to zero yields Bellman's Equation (backward pass).

$$\frac{\partial \mathcal{G}}{\partial V_t(\boldsymbol{s})} = \begin{cases} -p_1(\boldsymbol{s}) + \mu_1(\boldsymbol{s}) & \text{, if } t = 1 \\ -p_t(\boldsymbol{s}) + \int_{\boldsymbol{s},\boldsymbol{a}} \pi_{t-1}(\boldsymbol{a}|\boldsymbol{s})p_{t-1}(\boldsymbol{s})p_{t-1}(\boldsymbol{s'}|\boldsymbol{s},\boldsymbol{a}) & \text{, if } t > 1 \end{cases}$$

Setting the partial derivative w.r.t. the value function to zero yields the dynamics equation (forward pass).

$$\frac{\partial \mathcal{G}}{\partial p_t(\boldsymbol{y})} = -\eta_t(\boldsymbol{y}) - \beta_t(1 + \log \frac{p_t(\boldsymbol{y})}{q_t(\boldsymbol{y})}) + \lambda_{\boldsymbol{y},t}$$

Setting the partial derivative w.r.t. the distribution over task variables to zero yields the regularized target distribution $\tilde{p}_t(\boldsymbol{y})$,

$$\begin{aligned} \tilde{p}_t(\boldsymbol{y}) &= \exp\left(\log q_t(\boldsymbol{y}) - \frac{\eta_t(\boldsymbol{y})}{\beta_t} - 1 + \frac{\lambda_{\boldsymbol{y},t}}{\beta_t}\right) \\ &= \exp\left(\log q_t(\boldsymbol{y}) - \frac{1}{\beta_t}\eta_t(\boldsymbol{y}) - \log \int_{\boldsymbol{y}} \exp\left(\log q_t(\boldsymbol{y}) - \frac{1}{\beta_t}\eta_t(\boldsymbol{y})\right) d\boldsymbol{y}\right). \end{aligned} \tag{A.6}$$

Alternatively, we can solve for the task space reward function, yielding

$$\tilde{\eta}_t(\boldsymbol{y}) = -\beta_t \log \frac{p_t(\boldsymbol{y})}{q_t(\boldsymbol{y})}. \tag{A.7}$$

Setting (A.5) and (A.6) back into the dual yields

$$\mathcal{G}(\beta, \eta_t) = \sum_{t=2}^{T} \beta_t \log \int_{\boldsymbol{y}} \exp\left(\log q_t(\boldsymbol{y}) - \frac{1}{\beta_t}\eta_t(\boldsymbol{y})\right) d\boldsymbol{y} + \sum_{t=1}^{T-1} \alpha_t \epsilon_t + \int_{\boldsymbol{s}} V_1(\boldsymbol{s})\mu_1(\boldsymbol{s}).$$

A.2. Proof that the Alternate Update Direction is a Descent Direction

We will now prove, that interpolating the current estimate of the task-space reward function with the estimate computed by $(A.7)$ corresponds to an update along a descent direction of $\mathcal{G}$. To do so, we assume that the target distribution over task-variables, $q_t(\boldsymbol{y})$, and the induced distribution over task-variables, $p_t(\boldsymbol{y})$, are Gibbs distributions with potential function that are–without loss of generality–linear in arbitrary features $\boldsymbol{\psi}(\boldsymbol{y})$. The reward function estimate according to Equation (A.7) and the regularized target distribution according to Equation (A.6) are then given by

$$\tilde{\eta}_t(\boldsymbol{y}) = \beta_t \left(\boldsymbol{\theta}_{q,t} - \boldsymbol{\theta}_{p^{(i)},t}\right)^\top \boldsymbol{\psi}(\boldsymbol{y}) + \text{const} = \tilde{\boldsymbol{\theta}}_{\eta,t}^{(i)\top} \boldsymbol{\psi}(\boldsymbol{y}) + \text{const},$$
$$\tilde{p}_t(\boldsymbol{y}) \propto \exp\left((\boldsymbol{\theta}_{q,t} - \frac{1}{\beta_t}\boldsymbol{\theta}_{\eta,t}^{(i)})^\top \boldsymbol{\psi}_t(\boldsymbol{y})\right).$$

Interpolating the current estimate of the reward function weights $\theta_{\eta,t}^{(i+1)}$ with the new estimate $\tilde{\theta}_{\eta,t}^{(i)}$ with step size α corresponds to the weight update

$$\begin{aligned}\boldsymbol{\theta}_{\eta,t}^{(i+1)} &= (1-\alpha)\boldsymbol{\theta}_{\eta,t}^{(i)} + \alpha\tilde{\boldsymbol{\theta}}_{\eta,t}^{(i)} \\ &= \boldsymbol{\theta}_{\eta,t}^{(i)} - \alpha\left(\boldsymbol{\theta}_{\eta,t}^{(i)} - \tilde{\boldsymbol{\theta}}_{\eta,t}^{(i)}\right) = \boldsymbol{\theta}_{\eta,t}^{(i)} - \alpha\boldsymbol{\delta}_{\theta_\eta,t}.\end{aligned}$$

Lemma 4. *Let $\tilde{p}_t(\boldsymbol{y})$ and $p_t(\boldsymbol{y})$ be Gibbs distributions, i.e. $\tilde{p}_t(\boldsymbol{y}) = Z_{\tilde{p}_t}^{-1} \exp\left(\boldsymbol{\theta}_{\tilde{p}_t}^\top \boldsymbol{\psi}(\boldsymbol{y})\right)$ and $p_t(\boldsymbol{y}) = Z_{p_t}^{-1} \exp\left(\boldsymbol{\theta}_{p_t}^\top \boldsymbol{\psi}(\boldsymbol{y})\right)$. Further, let $\boldsymbol{\delta}'_{\theta_\eta,t} = \beta_t^{-1}\boldsymbol{\delta}_{\theta_\eta,t}$. The scaled update direction $\boldsymbol{\delta}'_{\theta_\eta,t}$ then corresponds to the difference between the weights of the potential functions of $p_t(\boldsymbol{y})$ and $\tilde{p}_t(\boldsymbol{y})$,*

$$\boldsymbol{\delta}'_{\theta_\eta,t} = \boldsymbol{\theta}_{p_t} - \boldsymbol{\theta}_{\tilde{p}_t}.$$

Proof.

$$\begin{aligned}\boldsymbol{\delta}'_{\theta_\eta,t} &= \beta_t^{-1}\left(\boldsymbol{\theta}_{\eta,t}^{(i)} - \tilde{\boldsymbol{\theta}}_{\eta,t}^{(i)}\right) = \beta_t^{-1}\boldsymbol{\theta}_{\eta,t}^{(i)} - \left(\boldsymbol{\theta}_{q_t} - \boldsymbol{\theta}_{p_t^{(i)}}\right) \\ &= \boldsymbol{\theta}_{p_t} - \boldsymbol{\theta}_{\tilde{p}_t}\end{aligned}$$

□

Theorem 3. *Let $\tilde{p}_t(\boldsymbol{y})$ and $p_t(\boldsymbol{y})$ be defined as in Lemma 4. The update direction $\boldsymbol{\delta}'_{\theta_\eta,t}$ is then an ascent direction of $\mathcal{G}$.*

Proof.

$$\begin{aligned}
\left\langle \boldsymbol{\delta}'_{\theta_\eta,t}, \frac{\partial \mathcal{G}}{\partial \boldsymbol{\theta}_{\eta,t}} \right\rangle &= \left\langle \boldsymbol{\theta}_{p_t} - \boldsymbol{\theta}_{\tilde{p}_t}, \mathrm{E}_{p_t(\boldsymbol{y})}\left[\boldsymbol{\psi}(\boldsymbol{y})\right] - \mathrm{E}_{\tilde{p}_t(\boldsymbol{y})}\left[\boldsymbol{\psi}(\boldsymbol{y})\right] \right\rangle \\
&= \mathrm{E}_{p_t(\boldsymbol{y})}\left[(\boldsymbol{\theta}_{p_t} - \boldsymbol{\theta}_{\tilde{p}_t})^\top \boldsymbol{\psi}(\boldsymbol{y}) \right] \\
&\quad - \mathrm{E}_{\tilde{p}_t(\boldsymbol{y})}\left[(\boldsymbol{\theta}_{p_t} - \boldsymbol{\theta}_{\tilde{p}_t})^\top \boldsymbol{\psi}(\boldsymbol{y}) \right] \\
&= \quad \mathrm{E}_{p_t(\boldsymbol{y})}\left[\log \frac{p_t(\boldsymbol{y})}{\tilde{p}_t(\boldsymbol{y})} \right] + \mathrm{E}_{\tilde{p}_t(\boldsymbol{y})}\left[\log \frac{\tilde{p}_t(\boldsymbol{y})}{p_t(\boldsymbol{y})} \right] \\
&= D_{\mathrm{KL}}(p_t(\boldsymbol{y}) || \tilde{p}_t(\boldsymbol{y})) + D_{\mathrm{KL}}(\tilde{p}_t(\boldsymbol{y}) || p_t(\boldsymbol{y})) \\
&\geq 0,
\end{aligned}$$

where strict inequality holds if $\tilde{p}_t(\boldsymbol{y}) \neq p_t(\boldsymbol{y})$. □

By replacing $\tilde{p}_t$ by the empirical expert distribution q_t, convergence can also be shown for the special case of MaxEnt-IRL.

B. Appendix for Chapter 3

B.1. VIPS1 Derivations

For each update we wish to solve the optimization problem

$$\begin{aligned} \max_{q(\mathbf{x})} \quad & \int_{\mathbf{x}} q(\mathbf{x})\tilde{R}(\mathbf{x})d\mathbf{x} + \mathrm{H}(q(\mathbf{x})), \\ \text{subject to} \quad & \mathrm{KL}\Big(q(\mathbf{x})||q^{(i)}(\mathbf{x})\Big) \leq \epsilon, \\ & \int_{\mathbf{x}} q(\mathbf{x})d\mathbf{x} = 1, \end{aligned}$$

where we recall that the the reward surrogate $\tilde{R}(\mathbf{x})$ is a quadratic function and the variational approximation of the previous iteration, $q^{(i)}(\mathbf{x})$, is Gaussian. We formulate the optimization for general distributions $q(\mathbf{x})$, but we will see that the optimal solution is also Gaussian. Using the definition of the Shannon entropy and Kullback-Leibler divergence and introducing the Lagrangian multipliers η and λ, the Lagrangian function is given by

$$\begin{aligned} \mathcal{L}(q,\eta,\lambda) &= \int_{\mathbf{x}} q(\mathbf{x})\big(\tilde{R}(\mathbf{x}) - \log q(\mathbf{x})\big)d\mathbf{x} + \eta\left(\epsilon - \int_{\mathbf{x}} q(\mathbf{x})\big(\log q(\mathbf{x}) - \log q^{(i)}(\mathbf{x})\big)d\mathbf{x}\right) \\ &\quad + \lambda\big(1 - \int_{\mathbf{x}} q(\mathbf{x})d\mathbf{x}\big) \\ &= \int_{\mathbf{x}} q(\mathbf{x})\big(\tilde{R}(\mathbf{x}) - (1+\eta)\log q(\mathbf{x}) + \eta \log q^{(i)}(\mathbf{x}) - \lambda\big)d\mathbf{x} + \eta\epsilon + \lambda. \end{aligned}$$

The optimum $q^\star(\mathbf{x})$ occurs where the partial derivative $\frac{\partial\mathcal{L}(q,\eta,\lambda)}{\partial q(\mathbf{x})}$ is equal to zero, that is,

$$\begin{aligned} \frac{\partial\mathcal{L}(q^\star,\eta,\lambda)}{\partial q(\mathbf{x})} &= \tilde{R}(\mathbf{x}) - (1+\eta)\log q^\star(\mathbf{x};\eta,\lambda) - (1+\eta) + \eta\log q^{(i)}(\mathbf{x}) - \lambda \stackrel{!}{=} 0 \\ &\Rightarrow q^\star(\mathbf{x};\eta,\lambda) = \exp\Big(-\frac{\lambda+1+\eta}{1+\eta}\Big)\exp\Big(\frac{\tilde{R}(\mathbf{x}) + \eta\log q^{(i)}(\mathbf{x})}{1+\eta}\Big). \end{aligned} \tag{B.1}$$

The Lagrange dual function is, thus, given by

$$\begin{aligned}\mathcal{G}(\eta,\lambda) =&\mathcal{L}(q^\star,\eta,\lambda)\\ =&\int_{\mathbf{x}} q^\star(\mathbf{x};\eta,\lambda)\big(\tilde{R}(\mathbf{x}) - \big(-\lambda - 1 - \eta + \tilde{R}(\mathbf{x}) + \eta\log q^{(i)}\big) + \eta\log q^{(i)}(\mathbf{x}) - \lambda\big)d\mathbf{x}\\ &+\eta\epsilon+\lambda\\ =&(1+\eta)\int_{\mathbf{x}} q^\star(\mathbf{x};\eta,\lambda)d\mathbf{x} + \eta\epsilon + \lambda.\end{aligned}$$

As strong duality holds due to Slater's condition [Boyd and Vandenberghe, 2004], we can find the optimal distribution $q^\star(\mathbf{x};\eta,\lambda)$ by minimizing the dual function with respect to η and λ and then using the optimal step size $\eta^\star$ and Lagrangian multiplier $\lambda^\star$ to compute $q^\star(\mathbf{x};\eta^\star,\lambda^\star)$ according to Equation B.1. The partial derivatives are given by

$$\begin{aligned}\frac{\partial\mathcal{G}(\eta,\lambda)}{\partial\eta} =&\epsilon + \int_{\mathbf{x}} q^\star(\mathbf{x};\eta,\lambda)d\mathbf{x}\\ &+(1+\eta)\int_{\mathbf{x}} q^\star(\mathbf{x};\eta,\lambda)\Big(\frac{\log q^{(i)}(\mathbf{x})-1}{1+\eta} - \frac{\log q^\star(\mathbf{x};\eta,\lambda)}{(1+\eta)}\Big)d\mathbf{x}\\ =&\epsilon - \int_{\mathbf{x}} q^\star(\mathbf{x};\eta,\lambda)\big(\log q^\star(\mathbf{x};\eta,\lambda) - \log q^{(i)}\big)d\mathbf{x}\end{aligned}$$

and

$$\frac{\partial\mathcal{G}(\eta,\lambda)}{\partial\lambda} = -\int_{\mathbf{x}} q^\star(\mathbf{x};\eta,\lambda)d\mathbf{x} + 1,$$

where the optimal Lagrangian multiplier $\lambda^\star(\eta)$ for a given η normalizes $q^\star(\mathbf{x};\eta,\lambda^\star)$, that is,

$$\frac{\partial\mathcal{G}(\eta,\lambda^\star)}{\partial\lambda} = 0 \Leftrightarrow \int_{\mathbf{x}} q^\star(\mathbf{x};\eta,\lambda^\star)d\mathbf{x} = 1.$$

Hence, we can perform coordinate descent by alternately updating η along its partial derivative and computing the optimal λ. Such procedure corresponds to optimizing the dual

$$\mathcal{G}(\eta) = (1+\eta)\int_{\mathbf{x}} q^\star(\mathbf{x};\eta,\lambda^\star)d\mathbf{x} + \eta\epsilon + \lambda^\star(\eta) = 1 + \eta + \eta\epsilon + \lambda^\star(\eta) \tag{B.2}$$

based on the gradient

$$\frac{\partial\mathcal{G}(\eta)}{\partial\eta} = \epsilon - \mathrm{KL}(q^\star(\mathbf{x};\eta,\lambda^\star)||q^{(i)}(\mathbf{x})).$$

We will now express the approximation of the previous iteration $q^{(i)}(\mathbf{x})$ in terms of its natural parameters $\mathbf{Q}^{(i)}$ and $\mathbf{q}^{(i)}$ and the reward surrogate as

$$\tilde{R}(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^\top \mathbf{R}^{(i)}\mathbf{x} + \mathbf{x}^\top \mathbf{r}^{(i)}.$$

Then, according to Equation B.1, the optimal distribution

$$\begin{aligned} q^\star(\mathbf{x};\eta) = &\exp\big(\frac{\eta \log Z(\mathbf{Q}^{(i)},\mathbf{q}^{(i)}) - \lambda^\star(\eta) - 1 - \eta}{1+\eta}\big) \\ &\cdot \exp\big(-\frac{1}{2}\mathbf{x}^\top \frac{\mathbf{R}^{(i)} + \eta\mathbf{Q}^{(i)})}{1+\eta}\mathbf{x} + \mathbf{x}^\top \frac{\mathbf{r}^{(i)} + \eta\mathbf{q}^{(i)})}{1+\eta}\big) \end{aligned} \tag{B.3}$$

is Gaussian with natural parameters

$$\mathbf{Q}(\eta) = \frac{\eta}{\eta+1}\mathbf{Q}^{(i)} + \frac{1}{\eta+1}\mathbf{R}^{(i)}, \qquad \mathbf{q}(\eta) = \frac{\eta}{\eta+1}\mathbf{q}^{(i)} + \frac{1}{\eta+1}\mathbf{r}^{(i)}.$$

Further, we can see from Equation B.3 and the optimality condition $\int_{\mathbf{x}} q(\mathbf{x};\eta,\lambda^\star) = 1$, that

$$\begin{aligned} \lambda^\star(\eta) = &-(1+\eta)\log \int_{\mathbf{x}} \exp\big(-\frac{1}{2}\mathbf{x}^\top \frac{\mathbf{R}^{(i)} + \eta\mathbf{Q}^{(i)})}{1+\eta}\mathbf{x} + \mathbf{x}^\top \frac{\mathbf{r}^{(i)} + \eta\mathbf{q}^{(i)})}{1+\eta}\big) d\mathbf{x} - 1 - \eta \\ &+ \eta \log Z(\mathbf{Q}^{(i)},\mathbf{q}^{(i)}) \\ =&\eta \log Z(\mathbf{Q}^{(i)},\mathbf{q}^{(i)}) - (1+\eta)\log Z(\mathbf{Q}(\eta),\mathbf{q}(\eta)) - 1 - \eta. \end{aligned} \tag{B.4}$$

Using Equation B.4 and Equation B.2, the dual function can be expressed as

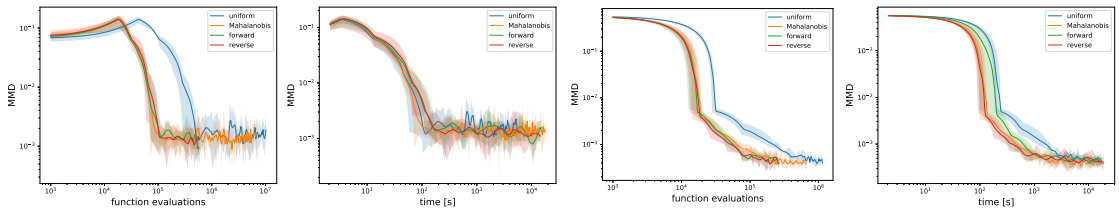
$$\mathcal{G}(\eta) = \eta\epsilon + \eta \log Z(\mathbf{Q}^{(i)},\mathbf{q}^{(i)}) - (1+\eta)\log Z(\mathbf{Q}(\eta),\mathbf{q}(\eta)). \tag{B.5}$$

B.2. Effects of Different Dissimilarity Measures for Sample Selection

VIPS++ uses the Mahalanobis distance to the mean of the distributions in the sample database as dissimilarity measure for sample selection according to Equation 3.19. We compared this choice to different dissimilarity measures, namely $KL\left(q(\mathbf{x}|o)||\mathcal{N}_{\mathbf{x}_i}(\mathbf{x})\right)$ (denoted as reverse KL) and $KL\left(\mathcal{N}_{\mathbf{x}_i}(\mathbf{x})||q(\mathbf{x}|o)\right)$ (denoted as forward) and against using a uniform distribution instead of Equation 3.19. The results are shown in Figure B.1.

B.3. Pseudo-Code for Sample Selection

The procedure for selecting relevant samples from the database is shown in Algorithm 6.



(a) GMM (20D) (b) GMM (20D) (c) Goodwin (d) Goodwin

Figure B.1.: Using the Mahalanobis distance as dissimilarity measure results in similar sample efficiency compared to using the KL divergence while adding less computational overhead.

Algorithm 6 Identifying relevant samples in the database

Require: database $\mathcal{S} = \{(\mathbf{x}_0, \log \tilde{p}(\mathbf{x_0}), \mathcal{N}_{\mathbf{x_0}}), \ldots, (\mathbf{x}_N, \log \tilde{p}(\mathbf{x_N}), \mathcal{N}_{\mathbf{x_N}})\}$
Require: number of components in the approximation, N_o
Require: desired number of samples that should be reused per component, n_{reuse}

function SELECT_SAMPLES
  $\boldsymbol{\mathcal{X}}_{\subset} \leftarrow \{\}$
  for $o = 1 \ldots N_o$ **do**
    $n_{\text{added}} \leftarrow 0$
    $h(\cdot, o) \leftarrow$ compute for each distinct component in the database according to (3.19)
    while $n_{\text{added}} < n_{\text{reuse}}$ **do**
      $i \sim h(\cdot, o)$ ▷ choose a distribution by sampling $h(i, o)$
      $h(\cdot, o) \leftarrow$ remove element i and normalize
      for each sample $\mathbf{x_j}$ of component $\mathcal{N}_i$ **do**
        if $\mathbf{x}_j \notin \boldsymbol{\mathcal{X}}_{\subset}$ **then**
          $\boldsymbol{\mathcal{X}}_{\subset} \leftarrow \boldsymbol{\mathcal{X}}_{\subset} \cup \mathbf{x}_j$
        end if
        $n_{\text{added}} \leftarrow n_{\text{added}} + 1$ ▷ also count $\mathbf{x}_j$ if it was already added
        if $n_{\text{added}} == n_{\text{reuse}}$ **then**
          break
        end if
      end for
    end while
  end for
  return $\boldsymbol{\mathcal{X}}_{\subset}$
end function

B.4. Approximating the Initial Reward and Sensitivity Regarding its Hyper-parameter

We approximate the initial reward of a new component based on the approximation given by Equation 3.23 because it is simpler and more efficient and unlikely to affect the selected candidate. Please note that the difference between the log-sum-exp and the maximum is numerically zero unless for candidates where the density of the current mixture model is close to the threshold. In such case the log-sum-exp can be larger by at most $\log(2)$.

Furthermore, during our experiments we did not exploit that the initial entropy of the new component can already be computed before deciding on the mean of the new component. Hence, we estimated the density at its mean as

$$q_{\mathbf{x}_s}(\mathbf{x}_s|o_n) \approx \max_{\mathbf{x}_i \in \mathcal{X}_{\text{total}}} \log q(\mathbf{x}_i), \tag{B.6}$$

since the current approximation needs to be evaluated anyway on each candidate for the first operand of the maximum operator in Equation 3.23. In the main document we presented the more principled, exact computation of $q_{\mathbf{x}_s}(\mathbf{x}_s|o_n)$ to improve clarity. However, in practice the difference between the described heuristic and the implemented heuristic is negligible because the errors that are introduced by the approximation are small compared to the variations of the assumed log weight $\log q(o_n)$.

For varying the (negated) assumed initial weights we specify several different values in an array $\boldsymbol{\Delta} = [1000, 500, 200, 100, 50]$ and pick one of these values $\boldsymbol{\Delta}_j$ by cycling through this array. The adding heuristic is thus computed as

$$\tilde{R}_{\mathbf{x}_s}(o_{\text{n}}) = R(\mathbf{x}_s) - \max\left(\log q(\mathbf{x}_s), \max_{\mathbf{x}_i \in \mathcal{X}_{\text{total}}} \log q(\mathbf{x}_i) - \boldsymbol{\Delta}_j\right). \tag{B.7}$$

Instead of pre-specifying the possible values for the assumed initial weights, it would also be possible to sample continuous values from a given distribution. However, for small changes in the assumed initial weight the heuristic would typically select qualitatively similar candidates and, thus, it is simpler to specify a few values that relate to different levels of exploration than to specify a distribution. It would also be possible to specify a single value Δ, however, this would add a hyper-parameter that has to be tuned depending on the experiment. Furthermore, switching between different levels of exploration can be more efficient because we do not only want to add components close to missing modes, but also close to modes that are already covered in order to approximate them better. Figure B.2 shows learning curves for different values of Δ on the *planar robot* experiment with four goals, which features several disconnected non-Gaussian modes. Here, varying the values performed better than any fixed assumed value for the initial weight. Figure B.3

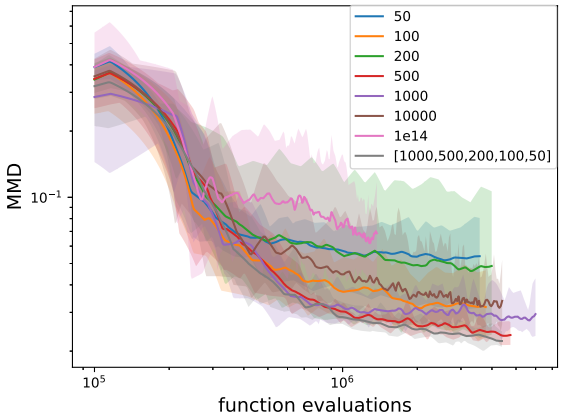


Figure B.2.: We evaluated the MMD for the *planar robot* experiment with four goals for different fixed values Δ of the assumed initial log-weight (negated) as well as for varying values. Varying the value (VIPS++) performed better than any fixed value that we tested. However, the experiment with $\Delta = 500$ indicates that tuning a fixed value may also perform well.

shows the different initial means that would have been chosen depending on the assumed initial weight. The selected candidates are sensible for a large range of Δ.

B.5. Scaling a Gaussian to Obtain a Desired Entropy

We want to find the scaling factor c to obtain a desired entropy H_{init} for a Gaussian distribution with given covariance matrix $\boldsymbol{\Sigma}$ of order $n \times n$.

$$\begin{aligned}\mathrm{H}(\boldsymbol{\Sigma}; c) &= \frac{1}{2}\log|2\pi e c\boldsymbol{\Sigma}| \\ &= \frac{1}{2}n\log(c) + \frac{1}{2}\log|2\pi e\boldsymbol{\Sigma}| \overset{!}{=} \mathrm{H}_{\text{init}} \Rightarrow c = \exp\left(\frac{1}{n}\left(2\mathrm{H}_{\text{init}} - \log|2e\pi\boldsymbol{\Sigma}|\right)\right)\end{aligned}$$

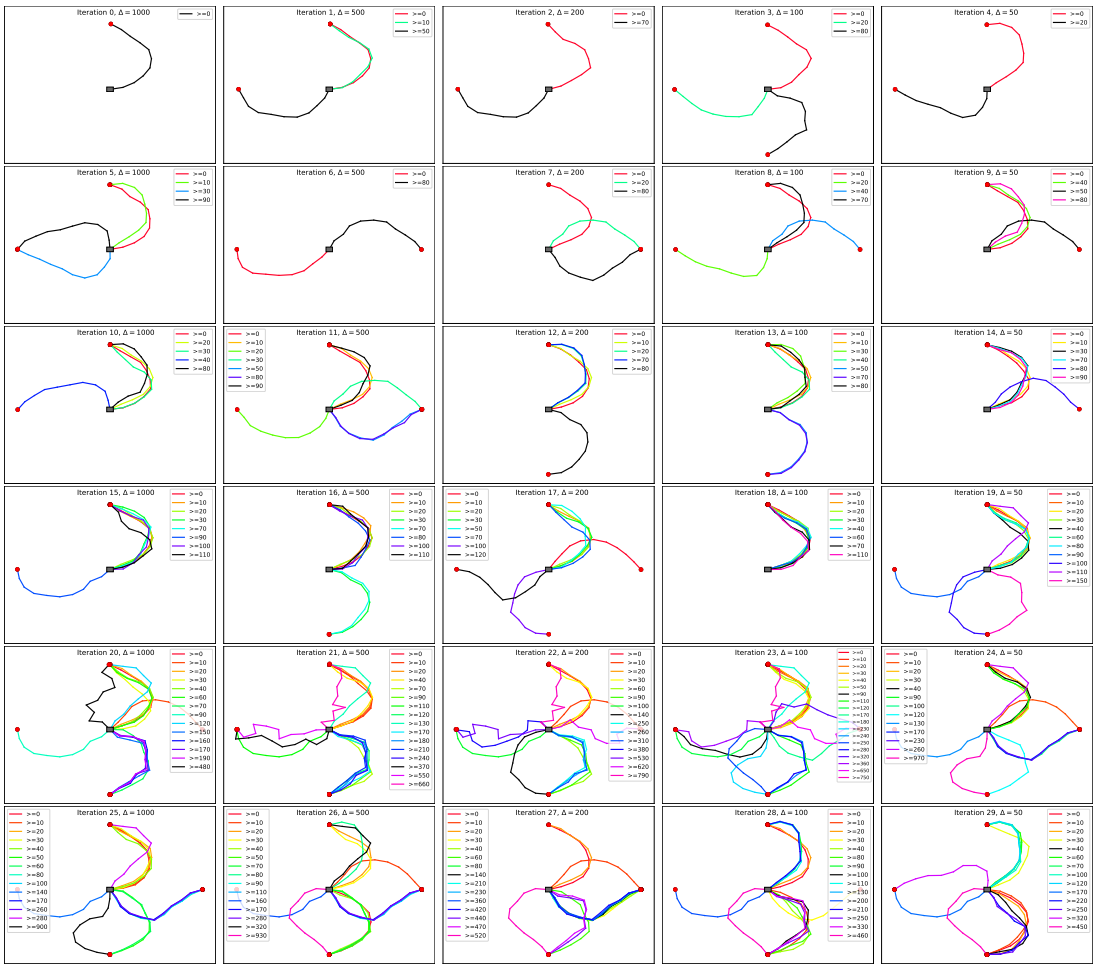


Figure B.3.: The plots show the candidates selected by the heuristic (Equation B.7) for values of Δ from 0 to 1000 in steps of 10 on the *planar robot* experiment with 4 goal positions for the first 30 iterations. At each iteration a new component is added based on the value shown in the title. These components are colored in black. Often the same candidate is selected for large ranges of Δ. All selected candidates seem reasonable. However, although all candidates reach one of the desired goal positions, the configurations can be less smooth (resulting in low likelihood due to the prior) for large values of Δ, which can be seen especially at iterations 20-23. While optimizing such components may require more iterations and samples, they are also more likely to discover a new mode. For example, the component added at iteration 20 is the first component that reaches the top goal position from the left side.

B.6. Goodwin Model

The Goodwin model is defined as

$$\begin{aligned} \frac{dx_1}{dt} &= \frac{a_1}{1 + a_2 x_g^{\rho}} - \alpha x_1 \\ \frac{dx_2}{dt} &= k_1 x_1 - \alpha x_2 \\ &\vdots \\ \frac{dx_g}{dt} &= k_{g-1} x_{g-1} - \alpha x_g, \end{aligned} \tag{B.8}$$

where x_1 represents the concentration of mRNA for a target gene, x_2 represents the corresponding protein product of the gene, and x_3 to x_g are intermediate protein species that ultimately lead to a negative feedback, via x_g, on the rate at which mRNA is transcribed. We consider $g = 9$ intermediate species and assume that the parameters $\rho = 10$ and $\alpha = 0.53$ are known. We put a Gamma prior with shape 2 and rate 1 on the remaining 10 parameters a_1, a_2 and $\kappa_1 \dots \kappa_8$ that need to be inferred. We use the prior also to randomly choose their true values. For an initial condition $\mathbf{x}_0 = \mathbf{0}$, we create 81 noisy observations $\mathbf{o}_{1\dots81}$ of x_1 and x_2 using steps of $dt = 1$. We assume Gaussian observation noise with zero mean and variance $\sigma^2 = 0.2$ and discard the first 40 observations. The posterior distribution is given by

$$p(a_1, a_2, \kappa_1, \dots, \kappa_8 | \mathbf{o}_{40\dots81}) = \frac{1}{Z} p(a_1) p(a_2) \prod_{i=1}^{8} p(\kappa_i) \prod_{t=40}^{81} p_t(\mathbf{o}_t | a_1, a_2, \kappa_1, \dots, \kappa_8), \tag{B.9}$$

where $p_t(\mathbf{o}_t | a_1, a_2, \kappa_1, \dots, \kappa_8)$ is a Gaussian distribution with variance $\sigma^2 = 0.2$ and a mean which is computed by numerically integrating the ODE (Equation B.8).

B.7. Planar Robot Experiment

The x and y coordinate of the end-effector are given by

$$x(\boldsymbol{\theta}) = \sum_{i=1}^{10} \cos\left(\sum_{j=1}^{i} \theta_j\right), \qquad y(\boldsymbol{\theta}) = \sum_{i=1}^{10} \sin\left(\sum_{j=1}^{i} \theta_j\right).$$

The target distribution is given as the product of two distributions,

$$p(\boldsymbol{\theta}) = \frac{1}{Z} p_{\text{conf}}(\boldsymbol{\theta}) p_{\text{cart}}(\boldsymbol{\theta}),$$

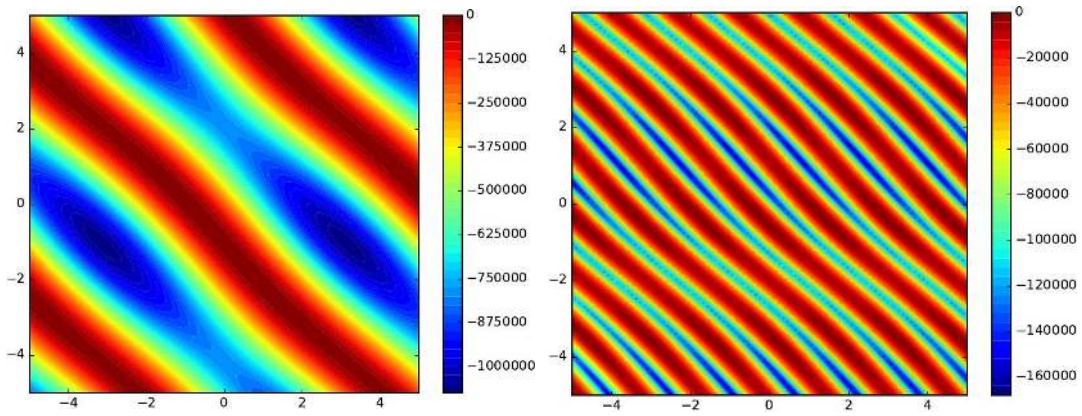


Figure B.4.: The plots show the target densities for the *planar robot* experiments with one goal position (left) and four goal positions (right), when varying the first two dimensions of a ground truth sample.

where $p_{\text{conf}}(\boldsymbol{\theta})$ enforces smooth configurations and $p_{\text{cart}}(\boldsymbol{\theta})$ penalizes deviations from the goal position. We model $p_{\text{conf}}(\boldsymbol{\theta})$ as zero mean Gaussian distribution with diagonal covariance matrix, where the angle of the first joint has a variance of 1 and the remaining joints have a variance of 4×10^{-2}. We consider two experiments that differ in the choice of goal positions. For the first experiment we specify a single goal position at position $(7, 0)$ modeled by a Gaussian distribution in Cartesian space with variance 1×10^{-4} in both directions, namely

$$p_{\text{cart},1}(\boldsymbol{\theta}) = \mathcal{N}\left(\begin{bmatrix} x(\boldsymbol{\theta}) \\ y(\boldsymbol{\theta}) \end{bmatrix} \mid \begin{bmatrix} 7 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \times 10^{-4} & 0 \\ 0 & 1 \times 10^{-4} \end{bmatrix}\right).$$

For the second experiment we specify four goal positions at positions $(7, 0)$, $(0, 7)$, $(-7, 0)$ and $(0, -7)$. The likelihood $p_{\text{cart},2}$ is given by the maximum over the four respective Gaussian distributions. Figure B.4 visualizes the target densities for both variants, when varying the first dimension of the respective first ground-truth sample.

B.8. Number of Components

The average number of components learned by VIPS++ is shown in Figure B.5.

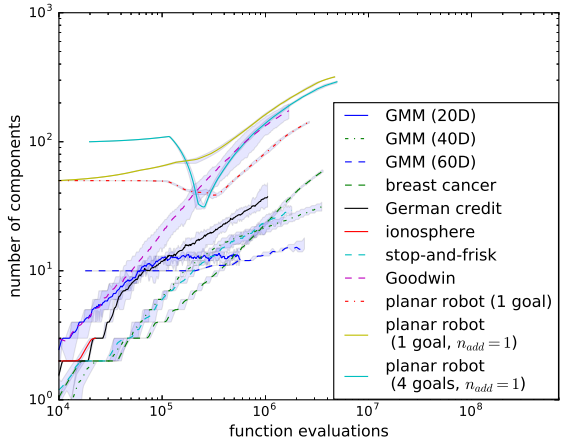


Figure B.5.: The average number of components learned by VIPS++ is plotted over function evaluations for all experiments in the test bed. When using the faster adding rate, $n_{\text{add}} = 1$, VIPS++ learns GMMs with approximately 350 components.

B.9. Implementations

For our comparisons we relied on open-source implementations, preferably by the original authors.

- For PTMCMC, we use an implementation by Ellis and van Haasteren [2017] that uses adaptive proposal distributions for the individual chains. We roughly tuned the number of chains for each experiment. As we could not run this implementation on our cluster, we ran the experiments on a fast quad-core laptop and made use of multi-threading. We therefore report four times the actual wall-clock time.
- For ESS, we use a Python implementation by Bovy [2013] that is based on the Matlab implementation by Iain Murray. If the target distribution decomposes into a product of a Gaussian prior and an arbitrary likelihood term, we directly provide this decomposition to the algorithm. If the target distribution does not use a Gaussian prior, we choose an appropriate Gaussian distribution $p_{\text{prior}}(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \alpha\mathbf{I})$ as prior and provide it along with the resulting likelihood $\log p_{\text{likelihood}}(\mathbf{x}) = \log \tilde{p}(\mathbf{x}) - \log p_{\text{prior}}(\mathbf{x})$, as described by Nishihara et al. [2014].
- Our comparisons with HMC are based on PYHMC [Nabney et al., 2018]. We tuned the step size and trajectory length for each experiment based on preliminary experiments.

We also performed some experiments with NUTS [Hoffman and Gelman, 2014], however, HMC with tuned parameters always outperformed the automatically tuned parameters of NUTS.

- For slice sampling, we use a Python adaptation [Slavitt, 2013] of a Matlab implementation by Iain Murray and tuned the step size based on preliminary experiments.

- For SVGD, we use the implementation of the original authors [Liu and Wang, 2016] and tune the step size based on preliminary experiments.

- For Variational Boosting, we use the implementation of the original authors [Miller et al., 2017]. However, this implementation is not optimized with respect to the number of function evaluations and often uses an unnecessary large number of samples. We therefore modified the implementation slightly. We also use their implementation of NPVI for our experiments.

- For black-box variational inference and inverse autoregressive flows we used our own implementation based on tensorflow [Abadi et al., 2015]. The code for conducting these experiments is available online[1]. For black-box variational inference, we tuned the learning rate as well as the number of samples per iteration (batch size). For inverse autoregressive flows, we tuned the learning rate, the batch size, the number of flows and the (common) width of the two hidden layers of the autoregressive networks for each flow.

B.10. Considered Algorithms and Experiments

Table B.1 provides an overview about which algorithms have been evaluated on which experiments.

- Our implementations of IAF and BBVI use a different code base (based on Tensorflow [Abadi et al., 2015]) for which we only implemented a subset of the experiments. However, we ensured that the test bed includes simple, unimodal experiments (*German credit* and *breast cancer*) as well as the most challenging, multi-modal experiments that we considered (*planar robot* and *GMM*).

- We did not evaluate PTMCMC on the simple test problems where parallel Markov chains would be wasteful.

[1] The implementation can be found at `https://github.com/OlegArenz/tensorflow_VI`.

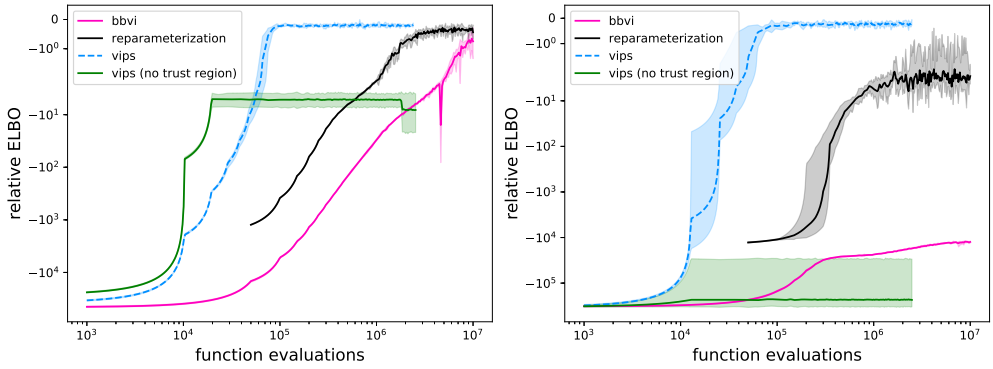
	VIPS	SVGD	ESS	HMC	NPVI	VBOOST	PTMCMC	SLICE	IAF	BBVI
German Credit	X	X	X	X	X	X	-	**X**	**X**	**X**
Breast Cancer	X	X	X	**X**	X	-	-	**X**	**X**	**X**
Frisk	X	X	**X**	X	X	X	-	**X**	-	-
GMM	X	X	X	-	-	-	X	**X**	**X**	**X**
Planar (1 goal)	X	X	X	-	X	X	X	X	**X**	**X**
Ionosphere	**X**	**X**	**X**	**X**	-	-	-	**X**	-	-
Goodwin	**X**	**X**	**X**	**X**	-	-	**X**	**X**	-	-
Planar (4 goal)	**X**	**X**	**X**	-	-	-	**X**	**X**	**X**	**X**
GMM (Higher Dim.)	**X**	-	-	-	-	-	-	-	-	-

Table B.1.: The table shows which algorithms were applied to each test problem. New experiment compared to our previous work [Arenz et al., 2018] are marked in bold.

- We did not evaluate HMC on the experiments with disconnected modes because we do not expect it to mix efficiently on such problems.
- We tried to evaluate VBOOST and NPVI on all test problems. However, we could not always obtain reliable results due to numerical problems that we could not fix without major changes to the implementation.
- We only evaluated VIPS++ on the higher-dimensional GMM experiments because it was the only method to solve the twenty-dimensional variant.

B.11. Alternatives for Learning Gaussian Variational Approximations

VIPS++ uses a variant of MORE (which we denote as VIPS1) for learning Gaussian variational approximations. However, it would also be possible to update the individual components using black-box variational inference [Ranganath et al., 2014] or the reparameterization trick, which assumes that the target distribution is differentiable. We



(a) breast cancer (b) planar robot (1 goal)

Figure B.6.: The proposed variant of MORE is significantly more efficient at optimizing Gaussian variational approximations compared to stochastic gradients using the reparameterization trick or black-box variational inference. By using locale surrogate objectives, we require trust regions to ensure stable optimization.

compared against these alternatives on *breast cancer* experiment as well as on the *planar robot* experiment with a single goal position. The learning curves of the ELBO are shown in Figure B.6. For each experiment, we subtracted a constant offset from the ELBO such that the highest (approximated) ELBO on each plot equals zero. Such relative ELBO ensures high resolution in the vicinity of the best ELBO on each of the plots. Please note that we use the symmetric logarithm to scale the y-Axis. Remarkably, VIPS1 is significantly more efficient than the reparameterization trick even though we do not require the gradient of the target distribution. We also compared against a variant of VIPS1 that does not constrain the KL divergence between updates. Such optimization is unstable as it exploits model errors caused by the local surrogate.

B.12. VIPS++ Hyper-Parameters

The hyper-parameters used for all experiments are given in Table B.2.

DESCRIPTION	VALUE
KL BOUND FOR COMPONENTS	$1 \times 10^{-2} \leq \epsilon(o) \leq 5$
NUMBER OF DESIRED SAMPLES (PER DIM. AND COMPONENT)	20
NUMBER OF REUSED SAMPLES (PER DIM. AND COMPONENT)	40
ADDING RATE FOR COMPONENTS	30 OR 1
DELETION RATE FOR COMPONENTS	10
MINIMUM WEIGHT	1×10^{-6}
INITIAL WEIGHT	1×10^{-29}
$\boldsymbol{\Delta}$ FOR ADDING-HEURISTIC	[1000, 500, 200, 100, 50]
ℓ_2-REGULARIZATION FOR WLS	$1 \times 10^{-14} \leq \kappa \leq 1 \times 10^{-6}$

Table B.2.: The table shows the hyper-parameters of VIPS++ as well as their values used during the experiments. The bound on the KL-divergence and the coefficient for ℓ_2-regularization when fitting the surrogates are automatically adapted within in the provided ranges.

B.13. Computing the Maximum Mean Discrepancy

We approximate the MMD between two sample sets $\mathbf{X}$ and $\mathbf{Y}$ as

$$\begin{aligned}\mathrm{MMD}(\mathbf{X}, \mathbf{Y}) = \frac{1}{m^2} \sum_{i,j}^{m} k(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{n^2} \sum_{i,j}^{n} k(\mathbf{y}_i, \mathbf{y}_j) \\ - \frac{2}{mn} \sum_{i}^{m} \sum_{j}^{n} k(\mathbf{x}_i, \mathbf{y}_i).\end{aligned}$$

We use a squared exponential kernel given by

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{1}{\alpha}(\mathbf{x} - \mathbf{y})^\top \boldsymbol{\Sigma} (\mathbf{x} - \mathbf{y})\right),$$

where $\boldsymbol{\Sigma}$ is a diagonal matrix where each entry is set to the median of squared distances within the ground-truth set and the bandwidth α is chosen depending on the problem. As true ground-truth samples are only available for the GMM experiment, we apply generalized elliptical slice sampling [Nishihara et al., 2014] with large values for burn-in, thinning and chain lengths to produce baseline samples that are regarded as ground-truth for the remaining experiments. Note that obtaining these ground-truth samples is computationally very expensive, taking up to two days of computation time on 128

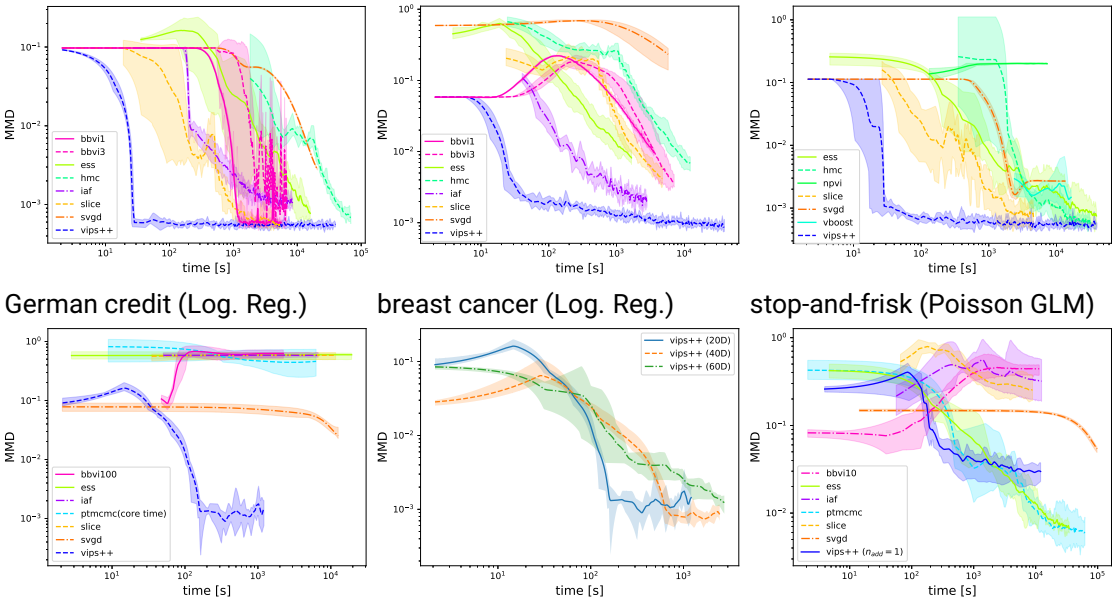


Figure B.7.: The maximum mean discrepancy with respect to baseline samples is plotted over computational time on log-log plots for the different sampling problems in the test bed.

CPU cores. We estimate the MMD based on ten thousand ground-truth samples and two thousand samples from the given sampling method. For MCMC methods, we choose the two thousand most promising samples by applying a sufficient amount of burn-in and using the largest thinning that keeps at least two thousand samples in the set.

B.14. Evaluations with Respect to Computational Time

Figure B.7 shows the achieved MMDs with respect to time for the experiments that have been omitted in the main document.

B.15. Evaluations with Respect to ELBO

We also compared the achieved ELBO $L(\theta)$ between VIPS++, inverse autoregressive flows (IAF) and black-box variational inference (BBVI). We approximate the ELBO based on

2000 samples from the learned approximation. The respective learning curves are shown in Figure B.8 where we subtracted a constant offset as described in Appendix B.11.

B.16. Visualization of Samples for Planar Robot Experiments

Samples obtained by BBVI, IAF, PTMCMC and VIPS++ for the planar robot experiment with one goal and four goals are shown in Figure B.9 and Figure B.10, respectively.

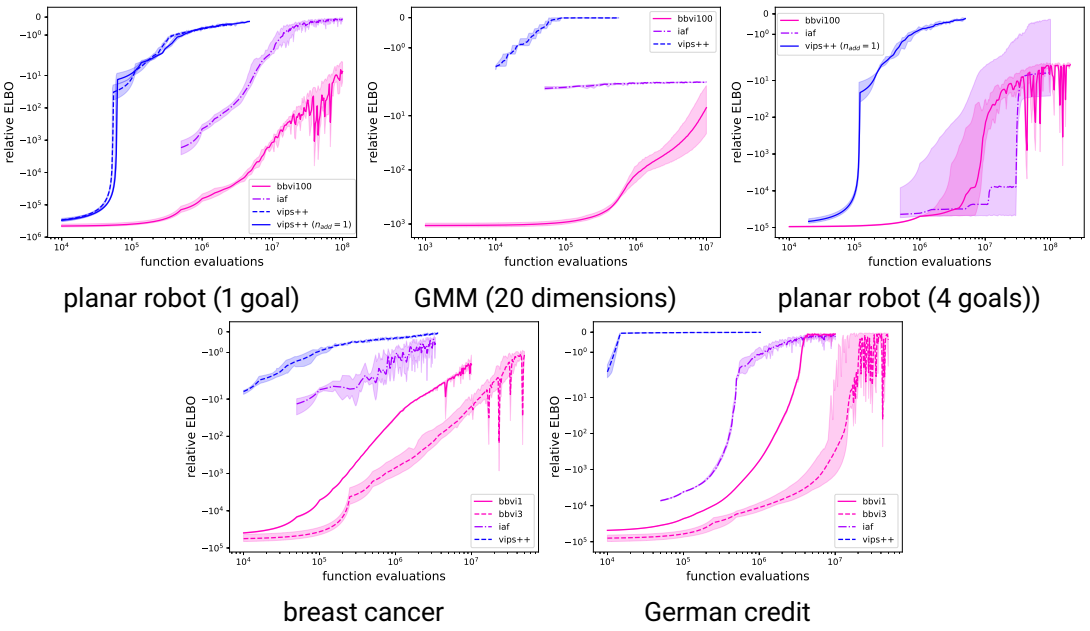


Figure B.8.: In contrast to the evaluation with respect to the MMD, all methods improve on the ELBO during learning, which is expected as the respective optimization problems aim to maximize the ELBO. Interestingly, IAF achieves a similar ELBO on the simpler planar robot experiment as VIPS++, although it performed significantly worse on the MMD. We verified that IAF achieves a similar approximated entropy as VIPS++, which is surprising since the learned approximation only sampled from one of the two main configurations (see Figure B.9). We hypothesize that even the large GMMs learned by VIPS++ are not able to cover the modes as well as the normalizing flows.

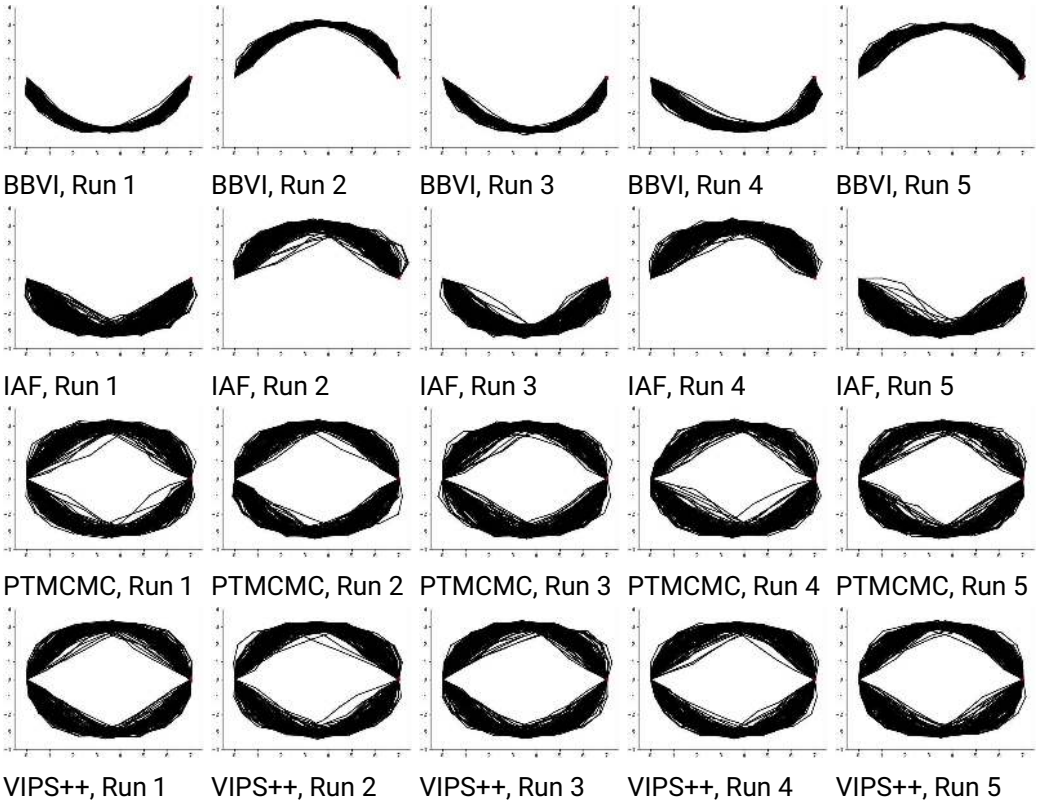


Figure B.9.: 200 sampled configurations are shown for five different training runs for the planar robot experiment with a single goal. For the variational inference methods BBVI, IAF and VIPS++, the plots show samples of the final learned model. For PTMCMC, the plots show the 200 most promising samples, which are obtained by applying a sufficient amount of burn-in and using the largest thinning that keeps at least 200 samples in the set.

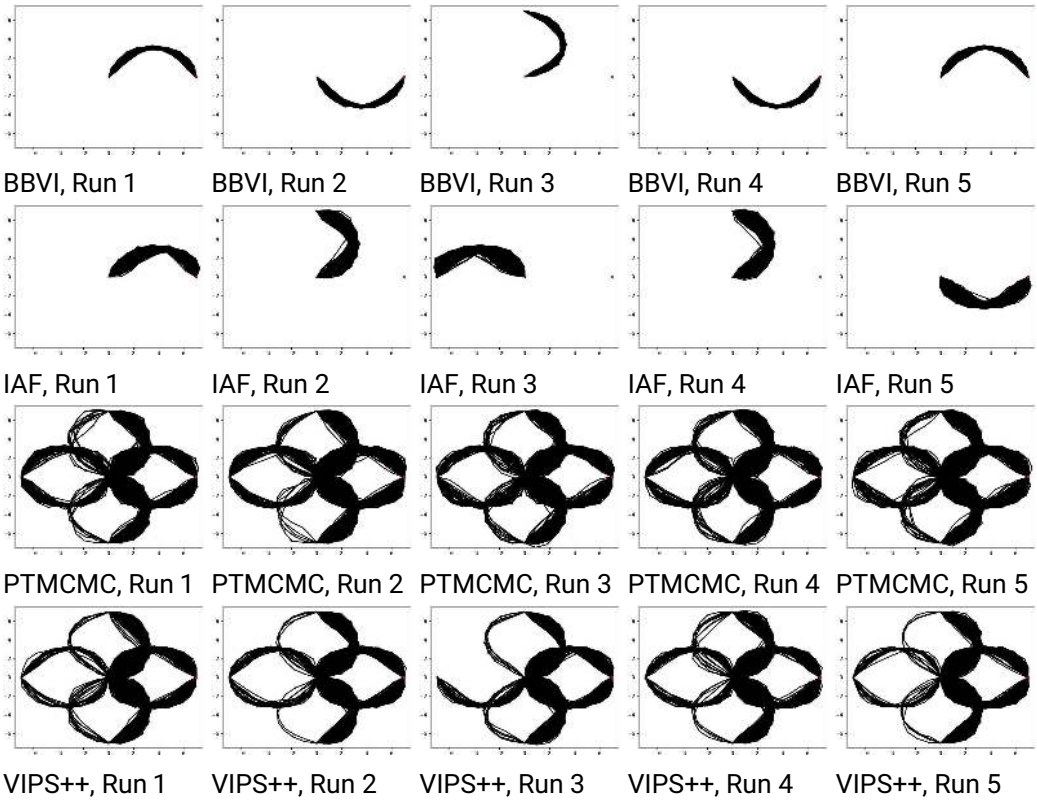


Figure B.10.: A thousand sampled configurations are shown for five different training runs for the planar robot experiment with four goals. For the variational inference methods BBVI, IAF and VIPS++, the plots show samples of the final learned model. For PTMCMC, the plots show the thousand most promising samples, which are obtained by applying a sufficient amount of burn-in and using the largest thinning that keeps at least thousand samples in the set.

C. Appendix for Chapter 4

C.1. BCE-loss of the AIRL Discriminator

The binary cross entropy loss for the AIRL discriminator is given by

$$\begin{aligned}
J_{\text{BCE}}(\boldsymbol{\theta}) =& E_{\mathbf{s},\mathbf{a}\sim q(\mathbf{s},\mathbf{a})}\left[\log\left(\frac{\exp\left(\bar{\nu}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})\right)}{\pi(\mathbf{a}|\mathbf{s})+\exp\left(\bar{\nu}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})\right)}\right)\right] \\
&+ E_{\mathbf{s},\mathbf{a}\sim p^{\pi}(\mathbf{s},\mathbf{a})}\left[\log\left(\frac{\pi(\mathbf{a}|\mathbf{s})}{\pi(\mathbf{a}|\mathbf{s})+\exp\left(\bar{\nu}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})\right)}\right)\right] \\
=& E_{\mathbf{s},\mathbf{a}\sim q(\mathbf{s},\mathbf{a})}\left[\bar{\nu}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})\right] + E_{\mathbf{s},\mathbf{a}\sim p^{\pi}(\mathbf{s},\mathbf{a})}\left[\log\pi(\mathbf{a}|\mathbf{s})\right] \\
&- 2E_{\mathbf{s},\mathbf{a}\sim\mu(\mathbf{s},\mathbf{a})}\left[\log\left(\pi(\mathbf{a}|\mathbf{s})+\exp\left(\bar{\nu}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})\right)\right)\right],
\end{aligned}$$

where $\mu(\mathbf{s},\mathbf{a}) = \frac{1}{2}(q(\mathbf{s},\mathbf{a}) + p^{\pi}(\mathbf{s},\mathbf{a}))$ is a mixture of the distributions induced by the expert and the agent. The gradient with respect to the discriminator parameters is given by

$$\begin{aligned}
\frac{dJ_{\text{BCE}}(\boldsymbol{\theta})}{d\boldsymbol{\theta}} =& E_{\mathbf{s},\mathbf{a}\sim q(\mathbf{s},\mathbf{a})}\left[\frac{d\bar{\nu}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})}{d\boldsymbol{\theta}}\right] - 2E_{\mathbf{s},\mathbf{a}\sim\mu(\mathbf{s},\mathbf{a})}\left[\frac{\exp\left(\bar{\nu}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})\right)}{\pi(\mathbf{a}|\mathbf{s})+\exp\left(\bar{\nu}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})\right)}\frac{d\bar{\nu}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})}{d\boldsymbol{\theta}}\right] \\
=& E_{\mathbf{s},\mathbf{a}\sim q(\mathbf{s},\mathbf{a})}\left[\frac{d\bar{\nu}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})}{d\boldsymbol{\theta}}\right] - E_{\mathbf{s},\mathbf{a}\sim\mu(\mathbf{s},\mathbf{a})}\left[\frac{\bar{p}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})}{\frac{1}{2}\left(p^{\pi}(\mathbf{s},\mathbf{a})+\bar{p}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})\right)}\frac{d\bar{\nu}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})}{d\boldsymbol{\theta}}\right] \\
=& E_{\mathbf{s},\mathbf{a}\sim q(\mathbf{s},\mathbf{a})}\left[\frac{d\bar{\nu}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})}{d\boldsymbol{\theta}}\right] - E_{\mathbf{s},\mathbf{a}\sim\mu(\mathbf{s},\mathbf{a})}\left[\frac{\bar{p}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})}{\bar{\mu}(\mathbf{s},\mathbf{a})}\frac{d\bar{\nu}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})}{d\boldsymbol{\theta}}\right],
\end{aligned} \tag{C.1}$$

where we introduced $\bar{p}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a}) = p^{\pi}(\mathbf{s})\exp\left(\bar{\nu}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})\right)$ and $\bar{\mu}(\mathbf{s},\mathbf{a}) = \frac{1}{2}\left(p^{\pi}(\mathbf{s},\mathbf{a}) + \bar{p}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})\right)$.

Fu et al. [2018] argue that, when assuming that the policy π maximizes the policy objective, we would have $\bar{p}_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a}) = p_{\boldsymbol{\theta}}(\mathbf{s},\mathbf{a})$ and that the gradient (Eq. C.1) would then correspond to an importance-sampling based estimate of the maximum-likelihood gradient

(Eq. 4.15). However, for obtaining the correct importance weights, we would need to further assume that $\bar{p}_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) = q(\mathbf{s}, \mathbf{a})$ such that $\bar{\mu}(\mathbf{s}, \mathbf{a}) = \mu(\mathbf{s}, \mathbf{a})$, that is, we would need to assume that the distribution induced by the current policy $p^{\pi}(\mathbf{s}, \mathbf{a}) = \bar{p}_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})$ matches the expert distribution. Furthermore, we would additionally need to assume that the discriminator is optimal such that $\bar{\nu}_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) = A^{\text{soft},\pi}(\mathbf{s}, \mathbf{a}) = \log \pi(\mathbf{a}|\mathbf{s})$ in order to ensure $\bar{p}_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a}) = p_{\boldsymbol{\theta}}(\mathbf{s}, \mathbf{a})$. While it is reassuring that both methods share a stationary point when the expert distribution is perfectly matched and the policy and discriminator are optimal, the connection between AIRL and MaxCausalEnt-IRL presented by Fu et al. [2018] seems rather weak.

C.2. Proof for Proposition 1

Let

$$p(t|\boldsymbol{\tau}) = \begin{cases} \frac{1}{T(\boldsymbol{\tau})} & t < T(\boldsymbol{\tau}) \\ 0 & otherwise \end{cases}$$

denote the probability of observing the time step t when the trajectory $\boldsymbol{\tau}$ of length $T(\boldsymbol{\tau})$ is given. Based on the assumption $p(\mathbf{o}|\boldsymbol{\tau}, t) = p(\mathbf{o}|\mathbf{s}_t^{\boldsymbol{\tau}}, \mathbf{a}_t^{\boldsymbol{\tau}})$, we can express $p(\mathbf{o}|\boldsymbol{\tau})$ as follows:

$$p(\mathbf{o}|\boldsymbol{\tau}) = \sum_{t=0}^{\infty} p(t|\boldsymbol{\tau}) p(\mathbf{o}|\boldsymbol{\tau}, t) = \sum_{t=0}^{\infty} p(t|\boldsymbol{\tau}) p(\mathbf{o}|\mathbf{s}_t^{\boldsymbol{\tau}}, \mathbf{a}_t^{\boldsymbol{\tau}}). \tag{C.2}$$

Based on equation C.2, we can express the objective of the episodic reinforcement learning problem given by Eq. 4.7 as

$$\begin{aligned}
J_{\text{rl,ep}}(\pi) &= \mathrm{E}_{\boldsymbol{\tau} \sim p^{\pi}(\boldsymbol{\tau})} \left[\int_{\mathbf{o}} p(\mathbf{o}|\boldsymbol{\tau}) f^*(D(\mathbf{o})) d\mathbf{o} \right] \\
&= \int_{\boldsymbol{\tau}} p^{\pi}(\boldsymbol{\tau}) \int_{\mathbf{o}} \sum_{t=0}^{\infty} p(t|\boldsymbol{\tau}) p(\mathbf{o}|\mathbf{s}_t^{\boldsymbol{\tau}}, \mathbf{a}_t^{\boldsymbol{\tau}}) f^*(D(\mathbf{o})) d\mathbf{o} d\boldsymbol{\tau} \\
&= \sum_{t=0}^{\infty} p(t) \int_{\boldsymbol{\tau}} p^{\pi}(\boldsymbol{\tau}|t) \int_{\mathbf{o}} p(\mathbf{o}|\mathbf{s}_t^{\boldsymbol{\tau}}, \mathbf{a}_t^{\boldsymbol{\tau}}) f^*(D(\mathbf{o})) d\mathbf{o} d\boldsymbol{\tau} \\
&= \sum_{t=0}^{\infty} p(t) \int_{\mathbf{s},\mathbf{a}} p_t^{\pi}(\mathbf{s}, \mathbf{a}) \int_{\mathbf{o}} p(\mathbf{o}|\mathbf{s}, \mathbf{a}) f^*(D(\mathbf{o})) d\mathbf{o} d\mathbf{s} d\mathbf{a} \\
&= (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \int_{\mathbf{s},\mathbf{a}} p_t^{\pi}(\mathbf{s}, \mathbf{a}) \int_{\mathbf{o}} p(\mathbf{o}|\mathbf{s}, \mathbf{a}) f^*(D(\mathbf{o})) d\mathbf{o} d\mathbf{s} d\mathbf{a}
\end{aligned}$$

$$= (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \int_{\mathbf{s},\mathbf{a}} p_t^{\pi}(\mathbf{s},\mathbf{a}) r_{\text{adv}}(\mathbf{s},\mathbf{a}) d\mathbf{s} d\mathbf{a}$$
$$= \int_{\mathbf{s},\mathbf{a}} p^{\pi}(\mathbf{s},\mathbf{a}) r_{\text{adv}}(\mathbf{s},\mathbf{a}) d\mathbf{s} d\mathbf{a}. \tag{C.3}$$

Hence, we can solve the episodic reinforcement learning problem (Eq. 4.7) also by maximizing the expected Markovian reward $r_{\text{adv}}(\mathbf{s},\mathbf{a}) = \int_{\mathbf{o}} p(\mathbf{o}|\mathbf{s},\mathbf{a}) f^*(D(\mathbf{o})) d\mathbf{o}$. When defining $p(\mathbf{o}|\mathbf{s},\mathbf{a})$ as a delta distribution at $\mathbf{o}(\mathbf{s},\mathbf{a}) = [\mathbf{s}^\top, \mathbf{a}^\top]^\top$ or at $\mathbf{o}(\mathbf{s},\mathbf{a}) = \mathbf{s}$ we can also recover the common objective of matching the expert's state-action distribution or its state marginal. However, such restrictions are not necessary to obtain Markovian rewards. □

C.3. Proof for Lemma 1

Based on Equation 4.22, we can express the reverse KL divergence $D_{\text{RKL}}(\tilde{p}(\mathbf{s},\mathbf{a})||q(\mathbf{s},\mathbf{a}))$ in terms of the lower bound $J_{\text{NAIL},\tilde{\pi}}(\tilde{\pi})$ (Eq.4.24) as

$$D_{\text{RKL}}(\tilde{p}(\mathbf{s},\mathbf{a})||q(\mathbf{s},\mathbf{a})) = -J_{\text{NAIL},\tilde{\pi}}(\tilde{\pi}) - \mathrm{E}_{\tilde{p}(\mathbf{o})}\left[\mathrm{KL}\left(\tilde{p}(\boldsymbol{\tau}|\mathbf{o})||\tilde{p}(\boldsymbol{\tau}|\mathbf{o})\right)\right] = -J_{\text{NAIL},\tilde{\pi}}(\tilde{\pi}),$$

and the reverse KL for a given policy π as

$$D_{\text{RKL}}(p^{\pi}(\mathbf{s},\mathbf{a})||q(\mathbf{s},\mathbf{a})) = -J_{\text{NAIL},\tilde{\pi}}(\pi) - \mathrm{E}_{p^{\pi}(\mathbf{o})}\left[\mathrm{KL}\left(p^{\pi}(\boldsymbol{\tau}|\mathbf{o})||\tilde{p}(\boldsymbol{\tau}|\mathbf{o})\right)\right].$$

Hence, for any policy π that satisfies $J_{\text{NAIL},\tilde{\pi}}(\pi) > J_{\text{NAIL},\tilde{\pi}}(\tilde{\pi})$, we have

$$\begin{aligned} &D_{\text{RKL}}(\tilde{p}(\mathbf{s},\mathbf{a})||q(\mathbf{s},\mathbf{a})) - D_{\text{RKL}}(p^{\pi}(\mathbf{s},\mathbf{a})||q(\mathbf{s},\mathbf{a})) \\ &\quad = \underbrace{J_{\text{NAIL},\tilde{\pi}}(\pi) - J_{\text{NAIL},\tilde{\pi}}(\tilde{\pi})}_{>0} + \underbrace{\mathrm{E}_{p^{\pi}(\mathbf{o})}\left[\mathrm{KL}\left(p^{\pi}(\boldsymbol{\tau}|\mathbf{o})||\tilde{p}(\boldsymbol{\tau}|\mathbf{o})\right)\right]}_{>0} > 0. \end{aligned}$$

□

C.4. Proof for Theorem 1

The sequence $\left\{D_{\text{RKL}}(p^{\pi^{(i)}}(\mathbf{s},\mathbf{a})||q(\mathbf{s},\mathbf{a}))\right\}_{i=0}^{\infty}$ is monotonously decreasing (see Lemma 1) and bounded below (due to the non-negativity of the KL) and thus convergent [Bibby, 1974]. At convergence $\pi^{(i)}$ must be a stationary point of the lower bound objective

(otherwise we could improve using gradient descent), that is

$$0 = \frac{d}{d\pi} J_{\text{NAIL},\pi^{(i)}}(\pi)\Big|_{\pi=\pi^{(i)}} = \frac{d}{d\pi} D_{\text{RKL}}(p^{\pi}(\mathbf{s},\mathbf{a})||q(\mathbf{s},\mathbf{a}))\Big|_{\pi=\pi^{(i)}} + \underbrace{\frac{d}{d\pi} \mathrm{E}_{p^{\pi}(\mathbf{o})}\left[\mathrm{KL}\left(p^{\pi}(\boldsymbol{\tau}|\mathbf{o})||p^{\pi^{(i)}}(\boldsymbol{\tau}|\mathbf{o})\right)\right]\Big|_{\pi=\pi^{(i)}}}_{=0}.$$

Hence, $\pi^{(i)}$ is a stationary point of the KL objective, that is,

$$\frac{d}{d\pi} D_{\text{RKL}}(p^{\pi}(\mathbf{s},\mathbf{a})||q(\mathbf{s},\mathbf{a}))\Big|_{\pi=\pi^{(i)}} = 0.$$

□

C.5. Proof for Lemma 2

Haarnoja et al. [2018] showed that the soft Q-function for policy π and reward r can be learned by repeatably applying the modified Bellman backup operator

$$\mathcal{T}^{\pi} Q^{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t) \triangleq r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathrm{E}_{\mathbf{s}_{t+1}\sim p}\left[V^{\text{soft}}(\mathbf{s}_{t+1})\right] \tag{C.4}$$

where

$$V^{\text{soft}}(\mathbf{s}) = \mathrm{E}_{\mathbf{a}\sim\pi}\left[Q^{\text{soft}}(\mathbf{s},\mathbf{a}) - \log \pi(\mathbf{a}|\mathbf{s})\right]. \tag{C.5}$$

We will now prove that $\hat{Q}(\mathbf{s},\mathbf{a}) \triangleq Q_r^{\tilde{\pi}}(\mathbf{s},\mathbf{a}) + \log \tilde{\pi}(\mathbf{a}|\mathbf{s})$ is the soft Q-function for policy $\tilde{\pi}$ and lower bound reward $r_{\text{lb}}(\mathbf{s},\mathbf{a}) = r(\mathbf{s},\mathbf{a}) + \log \tilde{\pi}(\mathbf{a}|\mathbf{s})$, that is $\hat{Q} = Q_{r_{\text{lb}}}^{\text{soft},\tilde{\pi}}$, by showing that it is a fixed point of the modified Bellman backup operator (Eq. C.4).

Applying the modified Bellman update to $\hat{Q}(\mathbf{s},\mathbf{a})$ yields

$$\begin{aligned}
\mathcal{T}^{\tilde{\pi}} \hat{Q}(\mathbf{s}_t,\mathbf{a}_t) &= r_{\text{lb}}(\mathbf{s}_t,\mathbf{a}_t) + \gamma \mathrm{E}_{\mathbf{s}_{t+1}\sim p, \mathbf{a}_{t+1}\sim\tilde{\pi}}\left[\hat{Q}(\mathbf{s},\mathbf{a}) - \log \tilde{\pi}(\mathbf{a}_{t+1}|\mathbf{s}_{t+1})\right] \\
&= r(\mathbf{s}_t,\mathbf{a}_t) + \log \tilde{\pi}(\mathbf{a}_t|\mathbf{s}_t) \\
&\quad + \gamma \mathrm{E}_{\mathbf{s}_{t+1}\sim p, \mathbf{a}_{t+1}\sim\tilde{\pi}}\left[Q_r^{\tilde{\pi}}(\mathbf{s}_{t+1},\mathbf{a}_{t+1}) + \log \tilde{\pi}(\mathbf{a}_{t+1}|\mathbf{s}_{t+1}) - \log \tilde{\pi}(\mathbf{a}_{t+1}|\mathbf{s}_{t+1})\right] \\
&= r(\mathbf{s}_t,\mathbf{a}_t) + \log \tilde{\pi}(\mathbf{a}_t|\mathbf{s}_t) + \gamma \mathrm{E}_{\mathbf{s}_{t+1}\sim p, \mathbf{a}_{t+1}\sim\tilde{\pi}}\left[Q_r^{\tilde{\pi}}(\mathbf{s}_{t+1},\mathbf{a}_{t+1})\right] \\
&= Q_r^{\tilde{\pi}}(\mathbf{s}_t,\mathbf{a}_t) + \log \tilde{\pi}(\mathbf{a}_t|\mathbf{s}_t) = \hat{Q}(\mathbf{s},\mathbf{a}).
\end{aligned}$$

□

C.6. Proof of Lemma 3

The first part of the proof closely follows the proof given by Haarnoja et al. [2018], which itself closely follows the proof of the policy improvement theorem given by Sutton and Barto [1998]. However, Haarnoja et al. [2018] only considered policies that are optimal with respect to the I-projection-loss (Eq. 4.30), although the proof is also valid for the less strict assumption of policy improvement, as shown below.

Based on our assumption we have for any state $\mathbf{s}$

$$\begin{aligned} \mathrm{E}_{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s})}\left[Q_{r_{\mathrm{lb}}}^{\mathrm{soft},\pi^{(i)}}(\mathbf{s},\mathbf{a}) - \log\pi(\mathbf{a}|\mathbf{s})\right] \geq & \mathrm{E}_{\mathbf{a}\sim\pi^{(i)}(\mathbf{a}|\mathbf{s})}\left[Q_{r_{\mathrm{lb}}}^{\mathrm{soft},\pi^{(i)}}(\mathbf{s},\mathbf{a}) - \log\pi^{(i)}(\mathbf{a}|\mathbf{s})\right] \\ = & V_{r_{\mathrm{lb}}}^{\mathrm{soft},\pi^{(i)}}(\mathbf{s}). \end{aligned} \tag{C.6}$$

Based on Equation C.6 we can repeatedly apply the soft Bellman equation to bound the soft Q-function for the old policy by the soft Q-function for the new policy, as follows:

$$\begin{aligned} Q_{r_{\mathrm{lb}}}^{\mathrm{soft},\pi^{(i)}}(\mathbf{s}_t,\mathbf{a}_t) &= r_{\mathrm{lb}}(\mathbf{s}_t,\mathbf{a}_t) + \gamma\mathrm{E}_{\mathbf{s}_{t+1}\sim p}\left[V^{\mathrm{soft}}(\mathbf{s}_{t+1})\right] \\ &\leq r_{\mathrm{lb}}(\mathbf{s}_t,\mathbf{a}_t) + \gamma\mathrm{E}_{\mathbf{s}_{t+1}\sim p,\mathbf{a}_{t+1}\sim\pi}\left[Q_{r_{\mathrm{lb}}}^{\mathrm{soft},\pi^{(i)}}(\mathbf{s}_{t+1},\mathbf{a}_{t+1}) - \log\pi(\mathbf{a}_{t+1}|\mathbf{s}_{t+1})\right] \\ &\vdots \\ &\leq Q_{r_{\mathrm{lb}}}^{\mathrm{soft},\pi}(\mathbf{s}_t,\mathbf{a}_t). \end{aligned} \tag{C.7}$$

We now express the lower bound objective $J_{\mathrm{NAIL},\pi^{(i)}}(\pi)$ in terms of its Q-function $Q_{r_{\mathrm{lb}}}^{\mathrm{soft},\pi}$ and relate it to the lower bound objective of the last policy $J_{\mathrm{NAIL},\pi^{(i)}}(\pi^{(i)})$ using the inequalities C.7 and C.6, namely,

$$\begin{aligned} J_{\mathrm{NAIL},\pi^{(i)}}(\pi) &= \int_{\mathbf{s},\mathbf{a}} p^{\pi}(\mathbf{s},\mathbf{a})\left(r_{\mathrm{lb}}^{\pi^{(i)}}(\mathbf{s},\mathbf{a}) - \log\pi(\mathbf{a}|\mathbf{s})\right) d\mathbf{s}d\mathbf{a} \\ &= (1-\gamma)\mathrm{E}_{\mathbf{s}\sim p_0(\mathbf{s}),\mathbf{a}\sim\pi}\left[Q_{r_{\mathrm{lb}}^{\pi^{(i)}}}^{\mathrm{soft},\pi} - \log\pi(\mathbf{a}|\mathbf{s})\right] \\ &\geq (1-\gamma)\mathrm{E}_{\mathbf{s}\sim p_0(\mathbf{s}),\mathbf{a}\sim\pi}\left[Q_{r_{\mathrm{lb}}^{\pi^{(i)}}}^{\mathrm{soft},\pi^{(i)}} - \log\pi(\mathbf{a}|\mathbf{s})\right] \\ &\geq (1-\gamma)\mathrm{E}_{\mathbf{s}\sim p_0(\mathbf{s}),\mathbf{a}\sim\pi^{(i)}}\left[Q_{r_{\mathrm{lb}}^{\pi^{(i)}}}^{\mathrm{soft},\pi^{(i)}} - \log\pi^{(i)}(\mathbf{a}|\mathbf{s})\right] = J_{\mathrm{NAIL},\pi^{(i)}}(\pi^{(i)}). \end{aligned}$$

□

Bibliography

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

P. Abbeel and A. Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2004.

P. Abbeel, A. Coates, and A. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *International Journal of Robotic Research (IJRR)*, 29:1608–1639, 2010.

A. Abdolmaleki, R. Lioutikov, N. Lua, L. P. Reis, J. Peters, and G. Neumann. Model-based relative entropy stochastic search. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 153–154, 2015.

A. Abdolmaleki, B. Price, N. Lau, L. P. Reis, and G. Neumann. Deriving and improving cma-es with information geometric trust regions. In *The Genetic and Evolutionary Computation Conference (GECCO 2017)*, July 2017.

H. Abdulsamad, O. Arenz, J. Peters, and G. Neumann. State-regularized policy search for linearized dynamical systems. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2017.

F. Abi-Farraj, C. Pacchierotti, O. Arenz, G. Neumann, and P. R. Giordano. A haptic shared-control architecture for guided multi-target robotic grasping. *IEEE Transactions on Haptics*, 13(2):270–285, 2019.

F. V. Agakov and D. Barber. An auxiliary variational method. In *International Conference on Neural Information Processing*, pages 561–566. Springer, 2004.

R. Akrour, A. Abdolmaleki, H. Abdulsamad, and G. Neumann. Model-free trajectory optimization for reinforcement learning. In *International Conference on Machine Learning (ICML)*, volume 6, pages 4342–4352, June 2016. URL http://eprints.lincoln.ac.uk/25747/.

R. Akrour, A. Abdolmaleki, H. Abdulsamad, J. Peters, and G. Neumann. Model-free trajectory-based policy optimization with monotonic improvement. *Journal of Machine Learning Research*, 19(14):1–25, 2018.

J. Altosaar, R. Ranganath, and D. M. Blei. Proximity variational inference. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

O. Arenz and G. Neumann. Non-adversarial imitation learning and its connections to adversarial methods. *arXiv preprint arXiv:2008.03525*, 2020.

O. Arenz, H. Abdulsamad, and G. Neumann. Optimal control and inverse optimal control by distribution matching. In *International Conference on Intelligent Robots and Systems (IROS)*, 2016.

O. Arenz, M. Zhong, and G. Neumann. Efficient gradient-free variational inference using policy search. In *International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 234–243. PMLR, 2018. URL http://proceedings.mlr.press/v80/arenz18a.html.

O. Arenz, M. Zhong, and G. Neumann. Trust-region variational inference with Gaussian mixture models. *Journal of Machine Learning Research*, 21(163):1–60, 2020. URL http://jmlr.org/papers/v21/19-524.html.

B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469 – 483, 2009. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2008.10.024. URL http://www.sciencedirect.com/science/article/pii/S0921889008001772.

M. Arjovsky and L. Bottou. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2017.

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning (ICML)*, 2017.

P. Becker, O. Arenz, and G. Neumann. Expected information maximization: Using the i-projection for mixture density estimation. In *International Conference on Learning Representations (ICLR)*, 2020.

J. Bibby. Axiomatisations of the average and a further generalisation of monotonic sequences. *Glasgow Mathematical Journal*, 15(1):63–65, 1974.

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, 2006.

C. M. Bishop, N. D. Lawrence, T. Jaakkola, and M. I. Jordan. Approximating posterior distributions in belief networks using mixtures. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 416–422, 1998.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 2017.

A. Boularias, J. Kober, and J. Peters. Relative entropy inverse reinforcement learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

J. Bovy. Python implementation of elliptical slice sampling, 2013. URL `https://github.com/jobovy/bovy_mcmc`.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

R. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

B. Calderhead. A general construction for parallelizing Metropolis-Hastings algorithms. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)*, Nov 2014.

B. Calderhead and M. Girolami. Estimating Bayes factors via thermodynamic integration and population mcmc. *Computational Statistics & Data Analysis*, 53(12):4028–4045, 2009.

X. Chen, M. Monfort, A. Liu, and B. D. Ziebart. Robust covariate shift regression. In *International Conference on Artificial Intelligence and Statistics*, pages 1270–1279, 2016.

R. Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

C. Daniel, G. Neumann, and J. Peters. Hierarchical relative entropy policy search. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

C. Daniel, G. Neumann, O. Kroemer, and J. Peters. Hierarchical relative entropy policy search. *Journal of Machine Learning Research (JMLR)*, 17:1–50, June 2016. URL http://eprints.lincoln.ac.uk/25743/.

M. P. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning (ICML)*, pages 465–472, 2011.

M. P. Deisenroth, G. Neumann, and J. Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, pages 388–403, 2013.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

L. Dinh, D. Krueger, and Y. Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

P. M. Djuric, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Miguez. Particle filtering. *IEEE signal processing magazine*, 20(5):19–38, 2003.

M. Donsker and S. Varadhan. Asymptotic evaluation of certain markov process expectations for large time. iv. *Communications on Pure and Applied Mathematics*, 36(2):183–212, Mar. 1983. ISSN 0010-3640. doi: 10.1002/cpa.3160360204.

A. Doucet, N. De Freitas, and N. Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001.

S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.

D. J. Earl and M. W. Deem. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910–3916, 2005.

J. Ellis and R. van Haasteren. jellis18/ptmcmcsampler: Official release, 2017. URL https://doi.org/10.5281/zenodo.1037579.

F. End, R. Akrour, J. Peters, and G. Neumann. Layered direct policy search for learning hierarchical skills. In *International Conference on Robotics and Automation (ICRA)*, May 2017. URL http://eprints.lincoln.ac.uk/26737/.

P. Englert, A. Paraschos, J. Peters, and M. P. Deisenroth. Model-based imitation learning by probabilistic trajectory matching. In *International Conference on Robotics and Automation (ICRA)*, 2013.

M. Ewerton, G. Maeda, G. Neumann, V. Kisner, G. Kollegger, J. Wiemeyer, and J. Peters. Movement primitives with multiple phase parameters. In *International Conference on Robotics and Automation (ICRA)*, pages 201–206. IEEE, 2016.

M. Ewerton, O. Arenz, G. Maeda, D. Koert, Z. Kolev, M. Takahashi, and J. Peters. Learning trajectory distributions for assisted teleoperation and path planning. *Frontiers in Robotics and AI*, 6:89, 2019. ISSN 2296-9144. doi: 10.3389/frobt.2019.00089. URL `https://www.frontiersin.org/article/10.3389/frobt.2019.00089`.

M. Ewerton, O. Arenz, and J. Peters. Assisted teleoperation in changing environments with a mixture of virtual guides. *Advanced Robotics*, 2020. doi: 10.1080/01691864.2020.1785326.

K. Fan, Z. Wang, J. Beck, J. T. Kwok, and K. Heller. Fast second-order stochastic backpropagation for variational inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1387–1395, 2015.

C. Finn, P. Christiano, P. Abbeel, and S. Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiV:1611.03852*, 2016a.

C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning, (ICML)*, pages 49–58, 2016b.

J. Fu, K. Luo, and S. Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2018.

M. Germain, K. Gregor, I. Murray, and H. Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning (ICML)*, pages 881–889, 2015.

S. J. Gershman, M. D. Hoffman, and D. M. Blei. Nonparametric variational inference. In *International Conference on Machine Learning (ICML)*, 235–242, 2012.

S. K. S. Ghasemipour, R. Zemel, and S. Gu. A divergence minimization perspective on imitation learning methods. In *Conference on Robot Learning (CoRL)*, pages 1259–1277, 2020.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2672–2680, 2014. URL `http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf`.

B. Goodwin. Oscillatory behavior in enzymatic control processes. *Advances in Enzyme Regulation*, 3:425–437, 1965.

W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud. FFJORD: free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations (ICLR)*, 2019.

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research (JMLR)*, 13:723–773, Mar. 2012. ISSN 1532-4435.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems (NeurIPS)*, pages 5767–5777, 2017.

F. Guo, X. Wang, K. Fan, T. Broderick, and D. B. Dunson. Boosting variational inference. *arXiv:1611.05559v2 [stat.ML]*, 2016.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, pages 1861–1870, 2018.

N. Hansen, S. Muller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.

N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver. Memory-based control with recurrent neural networks. *arXiv preprint arXiv:1512.04455*, 2015.

T. C. Hesterberg. *Advances in importance sampling*. PhD thesis, Stanford University, 1988.

J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4565–4573, 2016.

M. D. Hoffman and A. Gelman. The no-u-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research (JMLR)*, 15(1): 1593–1623, 2014.

M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research (JMLR)*, 14(4):1303–1347, 2013.

C.-W. Huang, D. Krueger, A. Lacoste, and A. Courville. Neural autoregressive flows. In *International Conference on Machine Learning (ICML)*, pages 2078–2087, 2018.

A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):21, 2017.

T. S. Jaakkola and M. I. Jordan. Improving the mean field approximation via the use of mixture distributions. *Learning in Graphical Models*, 89:163–174, 1998.

E. T. Jaynes. Information Theory and Statistical Mechanics. *Physical review*, 106(4):620, 1957.

S. Kakade. Optimizing average reward using discounted rewards. In *International Conference on Computational Learning Theory (COLT)*, pages 605–615. Springer, 2001.

H. J. Kappen. An introduction to stochastic control theory, path integrals and reinforcement learning. In *Cooperative Behavior in Neural Systems*, volume 887 of *American Institute of Physics Conference Series*, pages 149–181, Feb. 2007.

L. Ke, M. Barnes, W. Sun, G. Lee, S. Choudhury, and S. Srinivasa. Imitation learning as f-divergence minimization. *arXiv preprint arXiv:1905.12888*, 2019.

M. E. Khan, P. Baqué, F. Fleuret, and P. Fua. Kullback-Leibler proximal variational inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3402–3410, 2015.

M. E. Khan, R. Babanezhad, W. Lin, M. Schmidt, and M. Sugiyama. Faster stochastic variational inference using proximal-gradient methods with general divergence functions. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 319–328, 2016.

H. Kimura and S. Kobayashi. Reinforcement learning for locomotion of a two-linked robot arm. In *European Workshop on Learning Robots (EWRL)*, pages 144–153, 1997.

D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10215–10224. Curran Associates, Inc., 2018.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2013.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4743–4751, 2016.

D. Koert, G. Maeda, R. Lioutikov, G. Neumann, and J. Peters. Demonstration based trajectory optimization for generalizable robot motions. In *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*, pages 515–522. IEEE, 2016.

V. R. Konda and J. N. Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems (NeurIPS)*, pages 1008–1014, 2000.

A. Kong, J. S. Liu, and W. H. Wong. Sequential imputations and bayesian missing data problems. *Journal of the American statistical association*, 89(425):278–288, 1994.

P. Kormushev, S. Calinon, and D. G. Caldwell. Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input. *Advanced Robotics*, 25 (5):581–603, 2011. URL `http://dblp.uni-trier.de/db/journals/ar/ar25.html#KormushevCC11`.

I. Kostrikov, K. K. Agrawal, D. Dwibedi, S. Levine, and J. Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *International Conference on Learning Representations (ICLR)*, 2018.

I. Kostrikov, O. Nachum, and J. Tompson. Imitation learning via off-policy distribution matching. In *International Conference on Learning Representations (ICLR)*, 2020.

D. Kulić, C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura. Incremental learning of full body motion primitives and their sequencing through human motion observation. *The International Journal of Robotics Research (IJRR)*, 31(3):330–345, 2012.

S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Y. Kuniyoshi, M. Inaba, and H. Inoue. Learning by watching: Extracting reusable task knowledge from visual observation of human performance. *IEEE transactions on robotics and automation*, 10(6):799–822, 1994.

S. Levine and P. Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1071–1079, 2014.

S. Levine and V. Koltun. Continuous inverse optimal control with locally optimal examples. In *International Conference on Machine Learning (ICML)*, 2012.

S. Levine and V. Koltun. Guided policy search. In *International Conference on Machine Learning (ICML)*, pages 1–9, 2013.

S. Levine, Z. Popovic, and V. Koltun. Nonlinear inverse reinforcement learning with gaussian processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 19–27, 2011.

W. Li and E. Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *International Conference on Informatics in Control, Automation and Robotics*, 2004.

M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Q. Liu and D. Wang. Stein variational gradient descent: A general purpose Bayesian inference algorithm. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2378–2386. Curran Associates, Inc., 2016.

L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary deep generative models. In *International Conference on Machine Learning (ICML)*, pages 1445–1454, 2016.

G. Maeda, M. Ewerton, D. Koert, and J. Peters. Acquiring and generalizing the embodiment mapping from human observations to robot skills. *IEEE Robotics and Automation Letters*, 1(2):784–791, 2016.

H. Mania, A. Guy, and B. Recht. Simple random search of static linear policies is competitive for reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1803–1812. Curran Associates, Inc., 2018.

A. Menon and C. S. Ong. Linking losses for density ratio and class-probability estimation. In *International Conference on Machine Learning (ICML)*, pages 304–313, 2016.

A. C. Miller, N. J. Foti, A. D'Amour, and R. P. Adams. Variational boosting: Iteratively refining posterior approximations. In *International Conference on Machine Learning (ICML)*, 2017.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2016.

M. Monfort, A. Liu, and B. Ziebart. Intent prediction and trajectory forecasting via predictive inverse linear-quadratic regulation. In *AAAI Conference on Artificial Intelligence*, 2015.

I. Murray, R. Adams, and D. MacKay. Elliptical slice sampling. In *International Conference on Artificial Intelligence and Statistics*, pages 541–548, 2010.

I. T. Nabney, A. Vehtari, K. K., and M. R. T. pyhmc: Hamiltonian Monte Carlo in python, 2018. URL `https://github.com/rmcgibbo/pyhmc`.

O. Nachum, Y. Chow, B. Dai, and L. Li. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2318–2328, 2019.

R. Neal and G. E. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

R. M. Neal. Sampling from multimodal distributions using tempered transitions. *Statistics and Computing*, 6(4):353–366, Dec 1996.

R. M. Neal. Slice sampling. *The Annals of Statistics*, 31(3):705–767, 06 2003. doi: 10.1214/aos/1056562461.

G. Neu, A. Jonsson, and V. Gómez. A unified view of entropy-regularized Markov decision processes. *arXiv preprint arXiv: 1705.07798*, 2017. URL `http://arxiv.org/abs/1705.07798`.

A. Ng and S. Russell. Algorithms for Inverse Reinforcement Learning. In *in Proceceedings of the 17th International Conference on Machine Learning (ICML)*, 2000.

R. Nishihara, I. Murray, and R. P. Adams. Parallel mcmc with generalized elliptical slice sampling. *Journal of Machine Learning Research (JMLR)*, 15(1):2087–2112, Jan. 2014.

S. Nowozin, B. Cseke, and R. Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 271–279, 2016.

B. O'Donoghue, R. Munos, K. Kavukcuoglu, and V. Mnih. Pgq: Combining policy gradient and q-learning. *arXiv preprint arXiv:1611.01626*, 2016.

T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018.

G. Papamakarios, T. Pavlakou, and I. Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2338–2347, 2017.

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

E. Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

J. Peters, K. Muelling, and Y. Altun. Relative entropy policy search. In *AAAI Conference on Artificial Intelligence*, 2010.

C. Peterson and E. Hartman. Explorations of the mean field theory learning algorithm. *Neural Networks*, 2(6):457–494, 1989.

D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems (NeurIPS)*, pages 305–313, 1989.

T. Rainforth, Y. Zhou, X. Lu, Y. W. Teh, F. Wood, H. Yang, and J.-W. van de Meent. Inference trees: Adaptive inference with exploration. *arXiv preprint arXiv:1806.09550*, 2018.

R. Ranganath, S. Gerrish, and D. Blei. Black box variational inference. *Artificial Intelligence and Statistics*, pages 814–822, 2014.

R. Ranganath, D. Tran, and D. Blei. Hierarchical variational models. In *International Conference on Machine Learning (ICML)*, pages 324–333, 2016.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

N. Ratliff, A. Bagnell, and M. Zinkevich. Maximum margin planning. In *International Conference on Machine Learning (ICML)*, 2006.

J. Regier, M. I. Jordan, and J. McAuliffe. Fast black-box variational inference through stochastic trust-region optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2399–2408, 2017.

D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning (ICML)*, pages 1530–1538, 2015.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning (ICML)*, pages 1278–1286, 2014.

G. O. Roberts and O. Stramer. Langevin diffusions and Metropolis-Hastings algorithms. *Methodology and Computing in Applied Probability*, 4(4):337–357, 2002.

M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, pages 832–837, 1956.

S. Ross and D. Bagnell. Efficient reductions for imitation learning. In *International Conference on Artificial Intelligence and Statistics*, pages 661–668. PMLR, 2010.

E. Rueckert, M. Mindt, J. Peters, and G. Neumann. Robust policy updates for stochastic optimal control. In *International Conference on Humanoid Robots (HUMANOIDS)*, 2014.

S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Malaysia; Pearson Education Limited, 2016.

T. Salimans and D. A. Knowles. Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4):837–882, 2013.

T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

L. K. Saul, T. Jaakkola, and M. I. Jordan. Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, 4:61–76, 1996.

S. Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6):233–242, 1999. URL `http://courses.media.mit.edu/2003spring/mas963/schaal-TICS1999.pdf`.

J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, 2015.

C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423,623–656, 1948.

M. Shimosaka, K. Nishi, J. Sato, and H. Kataoka. Predicting driving behavior using inverse reinforcement learning with multiple reward functions towards environmental diversity. In *Intelligent Vehicles Symposium (IV), 2015 IEEE*, pages 567–572. IEEE, 2015.

S. Shirakawa, Y. Akimoto, K. Ouchi, and K. Ohara. Sample reuse in the covariance matrix adaptation evolution strategy based on importance sampling. In *Annual Conference on Genetic and Evolutionary Computation*, pages 305–312, 2015.

J. Singh, A. R. Srinivasan, G. Neumann, and A. Kucukyilmaz. Haptic-guided teleoperation of a 7-dof collaborative robot arm with an identical twin master. *IEEE Transactions on Haptics*, 13(1):246–252, 2020.

I. Slavitt. Python implementation of slice sampling, 2013. URL `https://isaacslavitt.com/2013/12/30/metropolis-hastings-and-slice-sampling/`.

D. C. Sorensen. Newton's method with a model trust region modification. *SIAM Journal on Numerical Analysis*, 19(2):409–426, 1982.

R. Stengel. *Stochastic Optimal Control: Theory and Application*. John Wiley & Sons, Inc., 1986.

M. Sugiyama, T. Suzuki, S. Nakajima, H. Kashima, P. von Bünau, and M. Kawanabe. Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60(4):699–746, 2008.

M. Sugiyama, T. Suzuki, and T. Kanamori. Density-ratio matching under the bregman divergence: a unified framework of density-ratio estimation. *Annals of the Institute of Statistical Mathematics*, 64(5):1009–1044, 2012.

R. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Neural Information Processing Systems (NIPS)*, 1999. URL `citeseer.ist.psu.edu/article/sutton00policy.html`.

R. S. Sutton and A. G. Barto. *Reinforcement Learning*. MIT Press, Boston, MA, 1998.

L. Theis and M. Hoffman. A trust-region method for stochastic variational inference with applications to streaming data. In *International Conference on Machine Learning (ICML)*, pages 2503–2511. PMLR, 2015.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 5026–5033, 2012.

F. Torabi, G. Warnell, and P. Stone. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018.

M. Toussaint. Robot trajectory optimization using approximate inference. In *International Conference on Machine Learning (ICML)*, 2009.

D. Tran, R. Ranganath, and D. M. Blei. The variational gaussian process. In *International Conference on Learning Representations (ICLR)*, 2016.

E. Uchibe. Efficient sample reuse in policy search by multiple importance sampling. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 545–552, 2018.

H. van Hoof, G. Neumann, and J. Peters. Non-parametric policy search with limited information loss. *Journal of Machine Learning Research (JMLR)*, 18(73):1–46, 2017. URL `http://jmlr.org/papers/v18/16-142.html`.

T. Weber, N. Heess, A. Eslami, J. Schulman, D. Wingate, and D. Silver. Reinforced variational inference. In *Advances in Neural Information Processing Systems (NeurIPS) Workshops*, 2015.

B. H. Williams, M. Toussaint, and A. J. Storkey. Extracting motion primitives from natural handwriting data. In *International Conference on Artificial Neural Networks (ICANN)*, 2006.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

R. J. Williams and J. Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

H. Xiao, M. Herman, J. Wagner, S. Ziesche, J. Etesami, and T. H. Linh. Wasserstein adversarial imitation learning. *arXiv preprint arXiv:1906.08113*, 2019.

H. Yin, A. Paiva, and A. Billard. Learning cost function and trajectory for robotic writing motion. In *International Conference on Humanoid Robots (HUMANOIDS)*, 2014.

Y.-x. Yuan. Recent advances in trust region algorithms. *Mathematical Programming*, 151 (1):249–281, 2015.

B. Ziebart, A. Maas, A. Bagnell, and A. Dey. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2008.

B. Ziebart, A. Bagnell, and A. Dey. Modeling interaction via the principle of maximum causal entropy. In *International Conference on Machine Learning (ICML)*, 2010.

B. D. Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, Machine Learning Department, Carnegie Mellon University, Dec 2010.

B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa. Planning-based prediction for pedestrians. In *Intelligent Robots and Systems (IROS)*, pages 3931–3936. IEEE, 2009.

O. Zobay. Variational Bayesian inference with Gaussian-mixture approximations. *Electronic Journal of Statistics*, 8(1):355–389, 2014. doi: 10.1214/14-EJS887.