Optimal Control and Inverse Optimal Control by Distribution Matching

Oleg Arenz[1], Hany Abdulsamad[2] and Gerhard Neumann[1]

Abstract— **Optimal control is a powerful approach to achieve optimal behavior. However, it typically requires a manual specification of a cost function which often contains several objectives, such as reaching goal positions at different time steps or energy efficiency. Manually trading-off these objectives is often difficult and requires a high engineering effort. In this paper, we present a new approach to specify optimal behavior. We directly specify the desired behavior by a distribution over future states or features of the states. For example, the experimenter could choose to reach certain mean positions with given accuracy/variance at specified time steps. Our approach also unifies optimal control and inverse optimal control in one framework. Given a desired state distribution, we estimate a cost function such that the optimal controller matches the desired distribution. If the desired distribution is estimated from expert demonstrations, our approach performs inverse optimal control. We evaluate our approach on several optimal and inverse optimal control tasks on non-linear systems using incremental linearizations similar to differential dynamic programming approaches.**

I. Introduction

Optimal control [1], [2] 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 [3] 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 [4] 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 [5], [6] 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 [7]. In order to ensure stability of the policy update, we follow the trajectory optimization approach from [8] 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

* This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No #645582 (RoMaNS).

[1] Computational Learning for Autonomous Systems, TU Darmstadt, 64287 Darmstadt, Germany, {oleg, geri}@robot-learning.de

[2] Intelligent Autonomous Systems, TU Darmstadt, 64287 Darmstadt, Germany, abdulsamad@ias.tu-darmstadt.de

© 2016 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

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 MaxEnt-IRL. Moreover, we compare our approach to the IRL approach given in [9] on a handwritten letter trajectory data set and to [10] 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.

A. Related Work

The first IRL approaches [3] 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 [3], the inverse RL problem is ill-defined and many reward functions exists that satisfy this criterion. Ziebart et al. [11] introduced a max-entropy formulation for inverse reinforcement learning that resolves the ill-posedness based on the principle of maximum entropy for estimating distributions [12]. We will discuss the MaxEnt-IRL algorithm in more details in the preliminaries. While the standard MaxEnt algorithms all require a model of the system dynamics to perform dynamic programming, [13] 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 in [5]. 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 for the trajectories is presented in [9]. 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 [9] in the experiment section.

There are many stochastic trajectory optimization techniques that rely on linearizations, including the incremental LQG algorithm [7], AICO [14], Robust Policy Updates [15] and the algorithm used for guided policy search [8]. To our experience, the approach presented in [8] is the most stable one as it uses a KL bound to the old policy to stabilize the policy update.

Englert et al. presented a model-based approach to imitation learning that shares a similar objective to our approach [10]. The authors modify the model-based policy search algorithm PILCO [16] 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 [10] obtain a highly non-linear policy by performing a computationally heavy, non-convex optimization.

II. 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.

A. Maximum Causal Entropy IRL

Maximum Causal Entropy IRL (MaxEnt-IRL) [4] aims at finding a reward function such that the resulting policy produces trajectories that are close to the expert's demonstrations. Following [17], 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. MaxEnt-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

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

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 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 [18] for further details on the derivation of the dual problem. The maximum entropy policy is then found to be

$$\pi_t(\mathbf{a}|\mathbf{s}) = \exp\left(Q^{\text{soft}}(\mathbf{s}, \mathbf{a}) - V^{\text{soft}}(\mathbf{s})\right). \tag{1}$$

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

$$V_t^{\text{soft}}(\mathbf{s}) = \log \int_{\mathbf{a}} \exp\left(Q_t^{\text{soft}}(\mathbf{s}, \mathbf{a})\right) d\mathbf{a}, \tag{2}$$

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

$$Q_t^{\text{soft}}(\mathbf{s}, \mathbf{a}) = r_t(\mathbf{s}, \mathbf{a}) + \int_{\mathbf{s}'} p_t(\mathbf{s}'|\mathbf{s}, \mathbf{a}) V_{t+1}^{\text{soft}}(\mathbf{s}') d\mathbf{s}'. \tag{3}$$

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

$$r_t(\mathbf{s}, \mathbf{a}) = \theta^{\top} \int_{\boldsymbol{y}} p_t(\boldsymbol{y}|\mathbf{s}, \mathbf{a}) \psi(\boldsymbol{y}) d\boldsymbol{y}, \tag{4}$$

where the weights $\boldsymbol{\theta}_t$ are the Lagrangian Multipliers of the feature matching constraint and learned by minimizing the dual function [18]

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

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 (1), i.e.

$$\frac{\partial \mathcal{G}}{\partial \boldsymbol{\theta}_t} = \tilde{\boldsymbol{\psi}}_t - \hat{\boldsymbol{\psi}}_t. \tag{5}$$

MaxEnt-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, MaxEnt-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.

III. (I)OC by Matching Distributions

In order to address the issues of Max-Ent 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

$$\max_{\pi_t(\mathbf{a}|\mathbf{s})} \sum_{t=1}^{T-1} H(\pi_t(\mathbf{a}|\mathbf{s})) - \sum_{t=2}^{T} \beta_t D_{\text{KL}}(p_t^{\pi}(\boldsymbol{y}) || q_t(\boldsymbol{y})), \tag{6}$$

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 MaxEnt-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 (6) is solved by minimizing the Lagrangian dual problem

$$\begin{aligned}\mathcal{G} =& \mathrm{E}_{p_1(\mathbf{s})}\left[V_1^{\text{soft}}(\mathbf{s})\right] \\ &+ \sum_t \beta_t \log \int_{\boldsymbol{y}} \exp\left(\log q_t(\boldsymbol{y}) - \frac{1}{\beta_t} \eta_t(\boldsymbol{y})\right) d\boldsymbol{y},\end{aligned} \tag{7}$$

where the policies $\pi(\mathbf{a}|\mathbf{s})$ and the softened state and state-action value functions $V^{\text{soft}}(\mathbf{s})$ and $Q^{\text{soft}}(\mathbf{s}, \mathbf{a})$ are the same as for MaxEnt-IRL, i.e., they are given by (1), (2) and (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

$$\eta_t(\boldsymbol{y}) = \beta_t \left(\log q_t(\boldsymbol{y}) - \log p_t^\pi(\boldsymbol{y}) \right) + \text{const.} \tag{8}$$

Note that (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 (1), (2) and (3). Furthermore, (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 (8) for estimating the task space reward function $\eta_t(\boldsymbol{y})$ based on an estimate of $p_t^\pi(\boldsymbol{y})$, (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.

$$\tilde{p}_t(\boldsymbol{y}) \propto \exp\left(\log q_t(\boldsymbol{y}) - \frac{1}{\beta_t} \eta_t(\boldsymbol{y}) \right), \tag{9}$$

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 (1).

The dual function (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

$$\frac{\partial \mathcal{G}}{\partial \boldsymbol{\theta}_t} = \mathrm{E}_{p_t^\pi(\boldsymbol{y})} \left[\boldsymbol{\psi}_t(\boldsymbol{y}) \right] - \mathrm{E}_{\tilde{p}_t(\boldsymbol{y})} \left[\boldsymbol{\psi}_t(\boldsymbol{y}) \right]. \tag{10}$$

However, while we use the gradient (10) for discussing the relative entropy based regularization as well as the relation between our approach and MaxEnt-IRL in Section III-A, we use a different procedure for optimizing the dual function (7) that is based on the recursive definition of the reward function (8) and discussed in Section III-B.

A. Relative Entropy Based Regularization

The gradient of the new formulation (10) only differs from the gradient of MaxEnt-IRL (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 (10) corresponds to the gradient of MaxEnt-IRL (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 (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 MaxEnt-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.

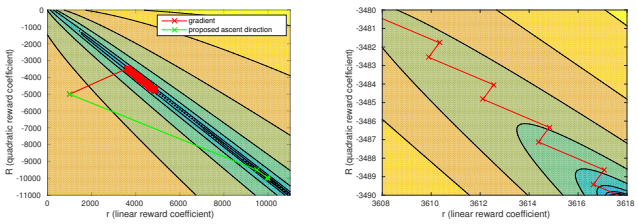
B. Alternative Descent Direction

In contrast to MaxEnt-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 (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 (1), (2) and (3), which can then be used for computing a new estimate $\boldsymbol{\theta}_{\eta,t}^{(i+1)}$ based on (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 (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 stepsize α.

The update direction $\boldsymbol{\delta}_{\boldsymbol{\theta}_{\eta,t}}$ is an ascent direction of the dual function (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 MaxEnt-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 MaxEnt-IRL, assuming that the parameters of the expert distribution $\boldsymbol{\theta}_{q_t}$ can be estimated.



(a) (b)

Fig. 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.

C. 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 as well as the resulting state distributions by means of 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 (1) are given by stochastic linear controllers

$$\pi_t(\mathbf{a}|\mathbf{s}) = \mathcal{N}\left(\mathbf{a}|\mathbf{K}_t\mathbf{s} + \mathbf{k}_t, \mathbf{\Sigma}_{\pi,t}\right). \tag{11}$$

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 (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.

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

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

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

the controller given by (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 (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.

1) 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

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

whereas the proposed ascent directions update along

$$\begin{aligned}\delta_{\mathbf{r}_{\boldsymbol{y},t}} &= \mathbf{\Sigma}_{p^\pi,t}^{-1}\boldsymbol{\mu}_{p^\pi,t} - \mathbf{\Sigma}_{\tilde{p},t}^{-1}\boldsymbol{\mu}_{\tilde{p},t},\\ \delta_{\mathbf{R}_{\boldsymbol{y},t}} &= -\frac{1}{2}\mathbf{\Sigma}_{p^\pi,t}^{-1} + \frac{1}{2}\mathbf{\Sigma}_{\tilde{p},t}^{-1}.\end{aligned} \tag{12}$$

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. 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. 1a shows three iterations of Inverse Reinforcement Learning when following the proposed search direction and thousand iterations when following the gradient. In both cases, the optimal stepsize 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. 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.

2) 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

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

Hence, the precision matrices of the expert distribution are interpolated with the reward matrices $\mathbf{R}_{\boldsymbol{y},t}$. When computing

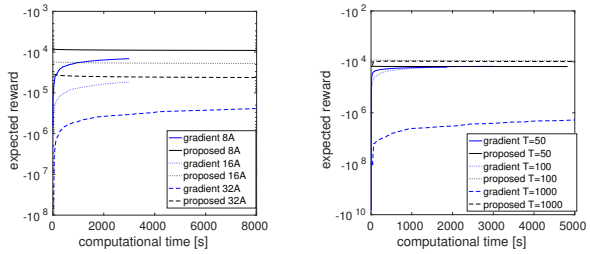


Fig. 2: (left) Expected reward for different number of actions and T=100. (right) Expected reward for different time horizons and eight actions. When using the proposed search direction, the algorithm converges significantly faster.

$\boldsymbol{\mu}_{\tilde{p},t}(\boldsymbol{y})$, 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.

D. 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. In the paper [7], the authors 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 become unstable if the trajectory distributions change too much. We follow [8] 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

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

IV. Experiments

We start our evaluation by comparing our method with MaxEnt-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 [9] where we want to learn a reward function for robotic handwriting and with [10] 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.

A. 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.

1) 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 MaxEnt-IRL, we use LBFGS [19] for optimization. For the proposed search direction such gradient based optimizers are not applicable. Therefore, we chose a simple stepsize adaption scheme that increases the stepsize by 1.2 if the dual function decreased after the last step and decreases the stepsize by 0.5 if the dual function increased. Furthermore, steps that led to an increase of the dual function are undone. Fig. 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, MaxEnt-IRL often failed to match the target distribution sufficiently well even after several hours of optimization.

2) 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. 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.

B. Robotic Handwriting

The problem of inferring a reward function for matching a target distribution was also tackled by [9] based on a variant [11] of MaxEnt-IRL that does not take causality

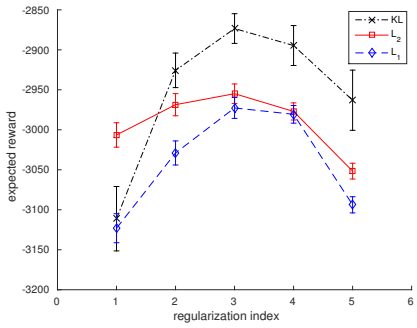


Fig. 3: Mean of the expected reward when presented with three demonstrations. Five different coefficients have been tested for each type of regularization. KL-based regularization can achieve more expected reward than regularization based on the ℓ_1 or ℓ_2 norm.

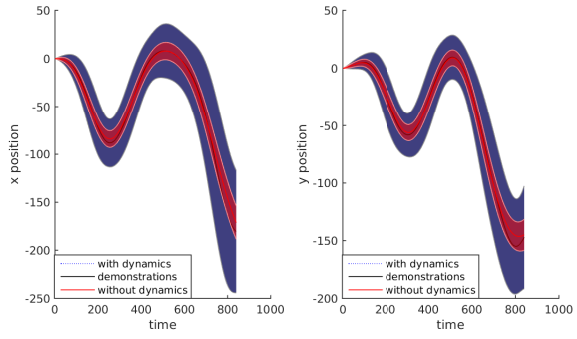


Fig. 4: Only when taking the system dynamics into account while learning the reward function, the resulting distribution (blue) matches the target distribution (black) accurately.

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 [20], [21] 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 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 to much reward for staying close to the mean trajectory and produces a stiff controller.

C. Pendulum Swing-Up

We compared our work to [10] on a simulated pendulum with a length of 0.6 meter that weighs 500 gram. A target

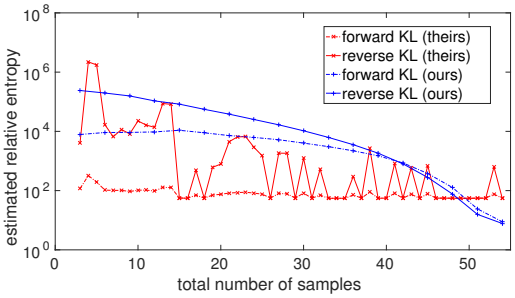


Fig. 5: Forward KL and reverse KL of our approach (blue) compared with [10] (red). The relative entropies have been estimated based on 1000 samples on the actual system. While our approach is less sample efficient, it converges to a better solution.

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 III-D. We learned the linear approximations using ridge regression based on five sample trajectory. 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 [8] 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 in [10] is very sample efficient by learning a Gaussian Process for approximating the system dynamics and, thus, uses only one sample per rollout.

Fig. 5 shows the relative entropy for both approaches plotted over the total number of samples. Since [10] 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 [10] provides good results already after few executions on the actual system, our approach converges to a better solution.

D. 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.

1) 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

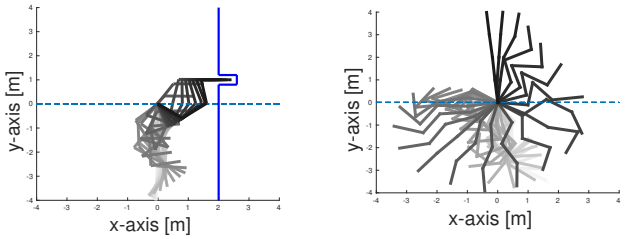


Fig. 6: (left) The peg-in-hole movement is performed with the specified accuracy. (right) A time-dependent reward function as well as the corresponding controller was learned from demonstrated swing-ups.

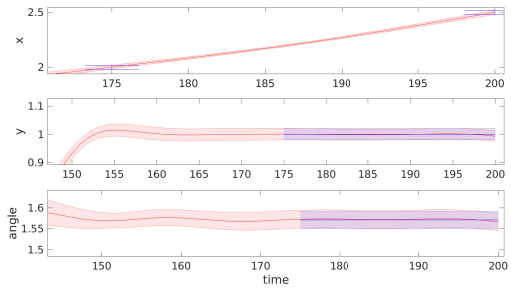


Fig. 7: The achieved distribution (red) was estimated based on 1000 samples and compared to the target distribution (blue) in the vicinity of specified time steps. Our approach accurately matches mean and variance for all three task variables.

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. 6 (left). Fig. 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.

2) 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 demonstrations were produced by a linear controller that was learned using MOTO [22]. Furthermore, only the joint positions were presented to our algorithm. Optimizing the learned reward function produced the desired behavior as shown in Fig. 6 (right).

V. 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 MaxEnt-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.

References

[1] R. Stengel, *Stochastic Optimal Control: Theory and Application*. John Wiley & Sons, Inc., 1986.

[2] H. J. Kappen, "An Introduction to Stochastic Control Theory, Path Integrals and Reinforcement Learning," in *Cooperative Behavior in Neural Systems*, Feb. 2007.

[3] A. Y. Ng and S. J. Russell, "Algorithms for Inverse Reinforcement Learning," in *Int. Conf. on Machine Learning (ICML)*, 2000.

[4] B. D. Ziebart, J. A. Bagnell, and A. K. Dey, "Modeling Interaction via the Principle of Maximum Causal Entropy," in *Int. Conf. on Machine Learning*, 2010.

[5] S. Levine, Z. Popovic, and V. Koltun, "Nonlinear Inverse Reinforcement Learning with Gaussian Processes," in *Neural Information Processing Systems (NIPS)*, 2011.

[6] S. Levine and V. Koltun, "Continuous Inverse Optimal Control with Locally Optimal Examples," in *Int. Conf. on Machine Learning*, 2012.

[7] W. Li and E. Todorov, "Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems," in *Int. Conf. on Informatics in Control, Automation and Robotics (ICRA)*, 2004.

[8] S. Levine and P. Abbeel, "Learning Neural Network Policies with Guided Policy Search under Unknown Dynamics," in *Neural Information Processing Systems*, 2014.

[9] H. Yin, A. Paiva, and A. Billard, "Learning Cost Function and Trajectory for Robotic Writing Motion," in *Int. Conf. on Humanoid Robots (HUMANOIDS)*, 2014.

[10] P. Englert, A. Paraschos, J. Peters, and M. P. Deisenroth, "Model-based Imitation Learning by Probabilistic Trajectory Matching," in *Int. Conf. on Robotics and Automation (ICRA)*, 2013.

[11] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum Entropy Inverse Reinforcement Learning," in *AAAI Conf. on Artificial Intelligence*, 2008.

[12] E. T. Jaynes, "Information Theory and Statistical Mechanics," *Physical review*, vol. 106, no. 4, p. 620, 1957.

[13] A. Boularias, J. Kober, and J. R. Peters, "Relative Entropy Inverse Reinforcement Learning," in *Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[14] M. Toussaint, "Robot Trajectory Optimization using Approximate Inference," in *Int. Conf. on Machine Learning (ICML)*, 2009.

[15] E. Rueckert, M. Mindt, J. Peters, and G. Neumann, "Robust Policy Updates for Stochastic Optimal Control," in *Int. Conf. on Humanoid Robots (HUMANOIDS)*, 2014.

[16] M. Deisenroth and C. Rasmussen, "PILCO: A Model-Based and Data-Efficient Approach to Policy Search," in *Int. Conf. on Machine Learning (ICML)*, 2011.

[17] P. Abbeel and A. Y. Ng, "Apprenticeship Learning via Inverse Reinforcement Learning," in *Int. Conf. on Machine Learning*, 2004.

[18] B. D. Ziebart, "Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy," Ph.D. dissertation, Machine Learning Department, Carnegie Mellon University, Dec 2010.

[19] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[20] M. Lichman, "UCI Machine Learning Repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[21] B. H. Williams, M. Toussaint, and A. J. Storkey, "Extracting Motion Primitives from Natural Handwriting data," in *Int. Conf. on Artificial Neural Networks (ICANN)*, 2006.

[22] R. Akrour, A. Abdolmaleki, H. Abdulsamad, and G. Neumann, "Model-Free Trajectory Optimization for Reinforcement Learning," in *Int. Conf. on Machine Learning (ICML)*, 2016.

Supplementary

S1. Derivations

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

Lagrangian:

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

Gradient of Lagrangian: ($t < T$)

$$\begin{aligned}
\frac{\mathcal{L}(p_t(\boldsymbol{s}), p_t(\boldsymbol{\psi}), \pi_t, \boldsymbol{\lambda}_t, V_t, \beta, \eta_t)}{\partial \pi_t(\boldsymbol{a}|\boldsymbol{s})} = &- \alpha_t p_t(\boldsymbol{s}) \log \pi_t(\boldsymbol{a}|\boldsymbol{s}) + \alpha_t p_t(\boldsymbol{s}) \log \pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s}) - p_t(\boldsymbol{s}) - \lambda_t(\boldsymbol{s}) - \mathbb{1}_{t>1} p_t(\boldsymbol{s}) V_t(\boldsymbol{s}) \\
&+ \mathbb{1}_{t>1} \int_{\boldsymbol{\psi}_2} \eta_t(\boldsymbol{\psi}) p_t(\boldsymbol{s}) p_t(\boldsymbol{\psi}|\boldsymbol{s}, \boldsymbol{a}) \, d\boldsymbol{\psi} + \int_{\boldsymbol{s'}} V_{t+1}(\boldsymbol{s'}) p_t(\boldsymbol{s}) p(\boldsymbol{s'}|\boldsymbol{s}, \boldsymbol{a}) \stackrel{!}{=} 0
\end{aligned}$$

$$\Rightarrow \pi_t^\star(\boldsymbol{a}|\boldsymbol{s}) = \exp\left(\frac{1}{\alpha_t}\left(-1+\alpha_t \log \pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s}) - \frac{\lambda_t(\boldsymbol{s})}{p_t(\boldsymbol{s})} - \mathbb{1}_{t>1}V_t(\boldsymbol{s}) + \int_{\boldsymbol{s'}} V_{t+1}(\boldsymbol{s'})p(\boldsymbol{s'}|\boldsymbol{s},\boldsymbol{a}) + \mathbb{1}_{t>1}\int_{\boldsymbol{\psi}} \eta_t(\boldsymbol{\psi})p_t(\boldsymbol{\psi}|\boldsymbol{s},\boldsymbol{a})\, d\boldsymbol{\psi}\right)\right) \tag{S1}$$

Elimination of λ:

$$\int_{\boldsymbol{a}} \exp\left(\frac{1}{\alpha_t}\left(-1+\alpha_t \log \pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s}) - \frac{\lambda_t(\boldsymbol{s})}{p_t(\boldsymbol{s})} - \mathbb{1}_{t>1}V_t(\boldsymbol{s}) + \int_{\boldsymbol{s'}} V_{t+1}(\boldsymbol{s'})p(\boldsymbol{s'}|\boldsymbol{s},\boldsymbol{a}) + \mathbb{1}_{t>1}\int_{\boldsymbol{\psi}} \eta_t(\boldsymbol{\psi})p_t(\boldsymbol{\psi}|\boldsymbol{s},\boldsymbol{a})\, d\boldsymbol{\psi}\right)\right) d\boldsymbol{a} \stackrel{!}{=} 1$$

$$\Rightarrow \exp\left(\frac{1}{\alpha_t}\left(1+\frac{\lambda_t(\boldsymbol{s})}{p_t(\boldsymbol{s})} + \mathbb{1}_{t>1}V_t(\boldsymbol{s})\right)\right) = \int_{\boldsymbol{a}} \exp\left(\frac{1}{\alpha_t}\left(\alpha_t \log \pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s}) + \int_{\boldsymbol{s'}} V_{t+1}(\boldsymbol{s'})p(\boldsymbol{s'}|\boldsymbol{s},\boldsymbol{a})d\boldsymbol{s'} + \mathbb{1}_{t>1}\int_{\boldsymbol{\psi}} \eta_t(\boldsymbol{\psi})p_t(\boldsymbol{\psi}|\boldsymbol{s},\boldsymbol{a})\, d\psi\right.\right.$$

$$\lambda_t(\boldsymbol{s}) = p_t(\boldsymbol{s})\left(-1-\mathbb{1}_{t>1}V_t(\boldsymbol{s}) + \alpha_t \log \int_{\boldsymbol{a}} \exp\left(\frac{1}{\alpha_t}\left(\alpha_t \log \pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s}) + \int_{\boldsymbol{s'}} V_{t+1}(\boldsymbol{s'})p(\boldsymbol{s'}|\boldsymbol{s},\boldsymbol{a})d\boldsymbol{s'} + \mathbb{1}_{t>1}\int_{\boldsymbol{\psi}} \eta_t(\boldsymbol{\psi})p_t(\boldsymbol{\psi}|\boldsymbol{s},\boldsymbol{a})\, d\boldsymbol{\psi}\right)\right)\right.$$

Inserting (S2) in (S1):

$$\Rightarrow \pi_t^\star(\boldsymbol{a}|\boldsymbol{s}) = \exp\left(\frac{1}{\alpha_t}\left(\alpha_t \log \pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s}) + \int_{\boldsymbol{s'}} V_{t+1}(\boldsymbol{s'})p(\boldsymbol{s'}|\boldsymbol{s},\boldsymbol{a})d\boldsymbol{s'} + \mathbb{1}_{t>1}\int_{\boldsymbol{\psi}} \eta_t(\boldsymbol{\psi})p_t(\boldsymbol{\psi}|\boldsymbol{s},\boldsymbol{a})\, d\boldsymbol{\psi}\right)\right. \tag{S2}$$

$$\left. - \log \int_a \exp\left(\frac{1}{\alpha_t}\left(\alpha_t \log \pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s}) + \int_{\boldsymbol{s'}} V_{t+1}(\boldsymbol{s'})p(\boldsymbol{s'}|\boldsymbol{s},\boldsymbol{a})d\boldsymbol{s'} + \mathbb{1}_{t>1}\int_{\boldsymbol{\psi}} \eta_t(\boldsymbol{\psi})p_t(\boldsymbol{\psi}|\boldsymbol{s},\boldsymbol{a})\, d\boldsymbol{\psi}\right)\right) d\boldsymbol{a}\right) \tag{S3}$$

Dual using (S3):

$$\begin{aligned}\mathcal{G}(p_t(\boldsymbol{s}), p_t(\boldsymbol{\psi}), V_t, \beta, \eta_t) = &-\sum_{t=2}^{T}\int_{\boldsymbol{\psi}} p_t(\boldsymbol{\psi}) \log \frac{p_t(\boldsymbol{\psi})}{q_t(\boldsymbol{\psi})} d\boldsymbol{\psi} \\ &+ \sum_{t=2}^{T} Z_{\psi,t}\left(\int_{\boldsymbol{\psi}} p_t(\boldsymbol{\psi}) - 1\right)\end{aligned}$$

$$
\begin{aligned}
&- \sum_{t=2}^{T} \int_{\psi} p_t(\boldsymbol{\psi}) \eta_t(\boldsymbol{\psi}) \\
&+ \eta_T(\boldsymbol{\psi}) \int_{\boldsymbol{s}} p_T(\boldsymbol{s}) p_T(\boldsymbol{\psi}|\boldsymbol{s}) \\
&+ \sum_{t=1}^{T-1} \alpha_t \epsilon_t \\
&+ \sum_{t=1}^{T-1} \int_{\boldsymbol{s}} p_t(\boldsymbol{s}) \alpha_t \log \int_{a} \exp\left(\frac{1}{\alpha_t} \left(\alpha_t \log \pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s}) + \int_{\boldsymbol{s}'} V_{t+1}(\boldsymbol{s}') p(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a}) d\boldsymbol{s}' + \mathbb{1}_{t>1} \int_{\boldsymbol{\psi}} \eta_t(\boldsymbol{\psi}) p_t(\boldsymbol{\psi}|\boldsymbol{s},\boldsymbol{a}) \right. \right. \\
&- \sum_{t=1}^{T} \int_{\boldsymbol{s}} p_t(\boldsymbol{s}) V_t(\boldsymbol{s}) \\
&+ \int_{\boldsymbol{s}} V_1(\boldsymbol{s}) \mu_1(\boldsymbol{s})
\end{aligned}
$$

Gradients of Dual:

$$
\frac{\partial \mathcal{G}(p_t(\boldsymbol{s}), V_t(\boldsymbol{s}))}{p_t(\boldsymbol{s})} = \begin{cases} -V_T(\boldsymbol{s}) + \int_{\boldsymbol{\psi}} p_T(\boldsymbol{\psi}|\boldsymbol{s}) \eta_T(\boldsymbol{\psi}) & t = \\ -V_1(\boldsymbol{s}) + \alpha_t \log \int_a \exp\left(\frac{1}{\alpha_t} \left(\alpha_t \log \pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s}) + \int_{\boldsymbol{s}'} V_{t+1}(\boldsymbol{s}') p(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a}) d\boldsymbol{s}' + \int_{\boldsymbol{\psi}} \eta_t(\boldsymbol{\psi}) p_t(\boldsymbol{\psi}|\boldsymbol{s},\boldsymbol{a}) \, d\boldsymbol{\psi} \right) \right) d\boldsymbol{a} & t < \\ -V_1(\boldsymbol{s}) + \alpha_t \log \int_a \exp\left(\frac{1}{\alpha_t} \left(\alpha_t \log \pi_{t,\text{last}}(\boldsymbol{a}|\boldsymbol{s}) + \int_{\boldsymbol{s}'} V_{t+1}(\boldsymbol{s}') p(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a}) d\boldsymbol{s}' \right) \right) d\boldsymbol{a} & t = \end{cases} \tag{S4}
$$

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

$$
\begin{aligned}
\frac{\partial \mathcal{G}(p_t(\boldsymbol{s}), V_t(\boldsymbol{s}))}{p_t(\boldsymbol{\psi})} &= -\eta_t(\boldsymbol{\psi}) - \log \frac{p_t(\boldsymbol{\psi})}{q_t(\boldsymbol{\psi})} + Z_{\boldsymbol{\psi},t} \\
\Rightarrow p_t^{\star}(\boldsymbol{\psi}) &= \exp\left(\log q_t(\boldsymbol{\psi}) - \eta_t(\boldsymbol{\psi}) - 1 + Z_{\boldsymbol{\psi},t} \right)
\end{aligned}
$$

$$= \exp\left(\log q_t(\boldsymbol{\psi}) - \eta_t(\boldsymbol{\psi}) - \log \int_{\boldsymbol{\psi}} \exp\left(\log q_t(\boldsymbol{\psi}) - \eta_t(\boldsymbol{\psi})\right)\right) \tag{S5}$$

setting (S4) and then (S5) back into the dual:

$$\begin{aligned}
\mathcal{G}(\beta, \eta_t) = &- \sum_{t=2}^{T} \int_{\boldsymbol{\psi}} p_t(\boldsymbol{\psi}) \log \frac{p_t(\boldsymbol{\psi})}{q_t(\boldsymbol{\psi})} d\boldsymbol{\psi} \\
&+ \sum_{t=2}^{T} Z_{\psi,t} \left(\int_{\boldsymbol{\psi}} p_t(\boldsymbol{\psi}) - 1 \right) \\
&- \sum_{t=2}^{T} \int_{\psi} p_t(\boldsymbol{\psi}) \eta_t(\boldsymbol{\psi}) \\
&+ \eta_T(\boldsymbol{\psi}) \int_{\boldsymbol{s}} p_T(\boldsymbol{s}) p_T(\boldsymbol{\psi}|\boldsymbol{s}) \\
&+ \sum_{t=1}^{T-1} \alpha_t \epsilon_t \\
&- \int_{\boldsymbol{s}} p_T(\boldsymbol{s}) \int_{\boldsymbol{\psi}} p_T(\boldsymbol{\psi}|\boldsymbol{s}) \eta_T(\boldsymbol{\psi}) \\
&+ \int_{\boldsymbol{s}} V_1(\boldsymbol{s}) \mu_1(\boldsymbol{s}) \\
= &\sum_{t=2}^{T} \log \int_{\boldsymbol{\psi}} \exp\left(\log q_t(\boldsymbol{\psi}) - \eta_t(\boldsymbol{\psi})\right) \\
&+ \sum_{t=1}^{T-1} \alpha_t \epsilon_t
\end{aligned}$$

$$+ \int_{\boldsymbol{s}} V_1(\boldsymbol{s}) \mu_1(\boldsymbol{s})$$