2011 IEEE/RSJ International Conference on
Intelligent Robots and Systems
September 25-30, 2011. San Francisco, CA, USA

Maximum Entropy Inverse Reinforcement Learning in Continuous State Spaces with Path Integrals

Navid Aghasadeghi and Timothy Bretl

Abstract— **In this paper, we consider the problem of inverse reinforcement learning for a particular class of continuous-time stochastic systems with continuous state and action spaces, under the assumption that both the cost function and the optimal control policy are parametric with known basis functions. Our goal is to produce a cost function for which a given policy, observed in experiment, is optimal. We proceed by enforcing a constraint on the relationship between input noise and input cost that produces a maximum entropy distribution over the space of all sample paths. We apply maximum likelihood estimation to approximate the parameters of this distribution (hence, of the cost function) given a finite set of sample paths. We iteratively improve our approximation by adding to this set the sample path that would be optimal given our current estimate of the cost function. Preliminary results in simulation provide empirical evidence that our algorithm converges.**

I. INTRODUCTION

Inverse reinforcement learning (IRL) is the problem of recovering a cost function that is consistent with observations of optimal or "expert" trajectories and with a given dynamic model [1]. In some cases, for example in the study of human motor control, it is precisely this cost function that we want to know. In other cases, imitating the behavior of an expert might be the goal. IRL problems are of interest in a wide range of applications, from basic science [2], [3] to optimal control of aircraft [4] and more recently aerobatic helicopter flight [5] within the robotics community.

In this paper, we propose an approach to IRL for a particular class of continuous-time stochastic systems with continuous state and action spaces. We begin by assuming that both the cost function and the optimal control policy are parametric with known basis functions, or in other words that these two functions are both weighted linear combinations of known features. Our IRL problem is then to recover the weights that describe the cost function given the weights that describe the policy.

To simplify this problem, we further assume that noise is additive in the input and that the cost function is quadratic in the input, with a weight that is inversely proportional to the variance of the noise. This assumption is reasonable in practice, since it assigns high cost to inputs with low variance, i.e., to inputs that are the most reliable. It has also been made before by [6]–[8], in order to solve the "forward" reinforcement learning problem more efficiently. In that case, this assumption transforms the Hamilton-Jacobi-Bellman (HJB) equation into a second order linear partial differential equation (PDE) and turns the problem of reinforcement learning into the problem of approximating a stochastic path integral. Similar assumptions have been made by others, for example requiring the cost function to have the form of a KL-divergence [9], and for much the same reason.

For IRL, our assumption about the relationship between input noise and inputs cost leads to a closed form maximum entropy distribution on all sample trajectories. We can proceed by applying a maximum-likelihood (ML) estimation to find the parameters of this distribution, which maximize the likelihood of observing the optimal policy, given a set of sampled trajectories. To find the parameters of the maximum entropy distribution, we utilize the *iterative scaling method* [10], [11]. This ML procedure leads to an estimated cost function, given a finite set of sampled trajectories. The problem of estimating a cost function given a finite set of trajectories, however, highly depends on the sampled trajectories. We thus introduce a cost updating algorithm which iteratively updates the estimate of the cost function. This algorithm iteratively adds to the set of all trajectories the sample trajectory which is optimal with respect to the current estimate of the cost function.

A similar approach to IRL was taken by [12], and indeed we drew direct inspiration from this prior work. The main difference between this approach and our own is that [12] does not consider updating the set of sampled trajectories. The algorithm in [12] should therefore be compared with the first component of our IRL method, namely the ML estimation. Moreover, as noted by the authors of [12], the optimization process used to estimate the cost function was not yet fully explored, and thus efficient implementations of the algorithm had not been proposed at the time of publication.

Our approach to IRL is also similar to the one of [13]. Both approaches rely on solving a sequence of reinforcement learning (RL) problems (equivalently, a sequence of optimal control problems) with candidate cost functions in order to estimate the unknown, true cost function. One difference between [13] and our own work is that we explicitly deal with continuous state and action spaces and use the path integral formulation to solve the forward RL problem, while [13] does not make use of a certain technique to solve this problem, and uses state discretization to solve RL problem in their experiments. The other recent work [5] also moves towards using a continuous state and action spaces. The more

N. Aghasadeghi is with the Department of Electrical and Computer Engineering and T. Bretl is with the Department of Aerospace Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, 61801, USA {aghasad1,tbretl}@illinois.edu

978-1-61284-456-5/11/$26.00 ©2011 IEEE

fundamental difference is that our approach also does not rely only on solving reinforcement learning problems in order to get information about the cost function—we use an initial set of sampled trajectories as well, from which we gain a lot of information. This same property is exhibited by [14], although in that case by making more restrictive assumptions about the cost function and about the nature of the control inputs.

There is a strong connection between what we propose and the MaxEnt approach [15], since this approach also considers parameter estimation on a maximum entropy distribution. However, [15] assumes discrete state/action spaces.

The rest of this paper proceeds as follows. Section II describes the system model we will consider, reviews an existing approach to reinforcement learning based on this model, and provides a formal statement of the IRL problem. Section III gives our solution to the IRL problem. Section IV evaluates the performance of our solution algorithm in simulation. We conclude in section V with a brief discussion of opportunities for future work.

II. Preliminaries and Problem Statement

In this section, we will first introduce our system model. Subsequently, we will talk about the problem of optimal control with PI^2 [8] as a stepping stone to introducing our proposed inverse reinforcement learning problem statement and algorithm.

A. System Model

We consider a dynamic system with the following continuous state-space representation:

$$\dot{\mathbf{x}}_t = f(\mathbf{x}_t) + G(\mathbf{x}_t)\Big(u_t + \epsilon_t\Big), \tag{1}$$

where x_t is the state of the trajectory at each time $t \geq 0$ and $\mathbf{x}_t \in \mathbb{R}^{n\times 1}$. The passive dynamics of the state are governed by $f(\mathbf{x}_t) \in \mathbb{R}^{n\times 1}$, and the control matrix is denoted by $G(\mathbf{x}_t) \in \mathbb{R}^{n\times p}$. The vector $u_t \in \mathbb{R}^{p\times 1}$ is the control vector, and ϵ_t is a gaussian noise with mean zero and variance Σ_ϵ.

Given the dynamics in equation (1), we can generate finite horizon trajectories starting at t_i and ending at t_N denoted by $\tau_{t_i} = (\mathbf{x}_t, t_i \leq t \leq t_N)$. Furthermore, we define the cost of a trajectory τ_{t_i} by $R(\tau_{t_i})$ as the following:

$$R(\tau_{t_i}) = \phi(\mathbf{x}_{t_N}) + \int_{t_i}^{t_N} \big(q_t(\mathbf{x}_t) + \frac{1}{2}u_t^T R u_t\big)dt, \tag{2}$$

where $\phi(x_N)$ represents the terminal cost, q_t represents an arbitrary state dependent cost, and $\frac{1}{2}u_t^T R u_t$ with a positive semi-definite matrix R, denotes the quadratic cost assigned to the inputs.

We talk about two problems associated with this system. First, in section II-B, we address optimal control, which is the problem of finding the set of inputs u_t, which minimizes the following value function:

$$V(x_{t_i}) = \min_{u_{t_i:t_N}} E_{\tau_i}\big[R(\tau_{t_i})\big], \tag{3}$$

where the expectation E_{τ_i} is taken over all of the trajectories starting at the state $\mathbf{x}_{t_i}$.

Solving the stochastic optimal control in its most general form is very difficult. Therefore, approximate methods are developed to solve this problem. In section II-B, we will talk about one such approach, which makes certain structural assumptions that are still reasonable in practice.

Additionally, we talk about the IRL problem in section II-C. Inverse reinforcement learning is the problem of recovering the cost function in (2), given the knowledge of the dynamics of the system (1), and the optimal policy, $\pi^* = \arg\min_\pi E_\tau\big[R(\tau)\big]$. Here we denote policies by π, and we define them as a mapping from the state of the system $\mathbf{x}_t$, to control inputs u_t, so we have: $\pi : \mathbb{R}^{n\times 1} \rightarrow \mathbb{R}^{p\times 1}$.

Similar to solving the stochastic optimal control problem, solving IRL problems is challenging as well. Additionally, IRL problems suffer from being ill-posed, i.e. different cost functions can result in the same optimal policy. In order to deal with these difficulties, we will make certain reasonable assumptions, and we will model the cost function as a weighted linear combinations of features, and we will introduce and address the IRL problem in section II-C.

B. Reinforcement Learning

In this section, we address the approach taken to solve (3), based on the PI^2 approach [8]. We begin by performing an Euler discretization of the dynamic system in (1) in the following way:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + f(\mathbf{x}_i)\Delta t + G(\mathbf{x}_i)\Big(u_i\Delta t + \epsilon_i\sqrt{\Delta t}\Big), \tag{4}$$

where Δt represents a constant time step. The symbol τ_i, now represents a trajectory discretized in time and starting at $\mathbf{x}_i$, $\tau_i = (\mathbf{x}_i, ..., \mathbf{x}_N)$. The optimal control is the problem of finding a sequence of inputs $u_i, ..., u_N$ such that:

$$V(x_i) = \min_{u_i,\ldots,u_N} E_{\tau_i}\big[R(\tau_i)\big], \tag{5}$$

where the cost $R(\tau_i)$ is the discrete counter part of the cost function (2). Authors in [8], based on previous work of [6] and [7], propose an approach to solve this problem given the assumption that $\gamma R^{-1} = \Sigma_\epsilon$, for a constant γ. This assumption imposes high costs on controls that are less noisy, and low costs on controls with high noise. This assumption makes intuitive sense since it is reasonable to assume a low cost for a very noisy and unreliable control input, and vice versa. Given this assumption, the paper [8] proposes an approach to solve the stochastic optimal control problem with dynamics (4), and value function defined by (5). The solution relies on changing the optimal control problem to an inference problem that involves computing a path integral.

Finding a solution to (5) requires finding optimal values for all input vectors $u_i, ..., u_N$, and thus could still be computationally expensive. In order to further simplify the problem, the paper [8] considers using parameterized policies, with a parameter vector θ. Finding the best parameterized policy then reduces to finding the optimal vector of parameters θ^*.

As a special case of parameterized policies, we consider using the Dynamic Movement Primitives (DMP) introduced in [16]. These policies are characterized by the following set of equations, which we have written in the form seen in equation (1):

$$\begin{pmatrix} \dot{x}_t \\ \dot{z}_t \\ \dot{y}_t \end{pmatrix} = \begin{pmatrix} -\alpha x_t \\ y_t \\ \alpha_z(\beta_z(g-y_t)-z_t) \end{pmatrix} + \begin{pmatrix} 0_{1\times p} \\ 0_{1\times p} \\ \mathbf{g}_t^{(c)T} \end{pmatrix} (\theta+\epsilon_t)$$

In the above DMP equations, we are assuming that the system in (1) can be partitioned into a directly and non-directly actuated parts (for more details see [8]). The DMP is then partitioned as $y_t = \mathbf{x}_t^{(c)}$, and $(x_t, z_t)^T = \mathbf{x}_t^{(m)}$, where $\mathbf{x}_t^{(c)}$ and $\mathbf{x}_t^{(m)}$ are the actuated and non-directly actuated parts of the system (1). Moreover, x and z are internal states of the DMP and α, α_z and β_z are time constants. These equations model a learnable point attractor starting at y_{t_0} and moving to the goal state g. The policy is parameterized by θ, which enters the equations linearly, and allows different shapes for trajectories. The basis functions for each time t are represented by $\mathbf{g}_t \in \mathbb{R}^{p\times 1}$ and defined as follows:

$$[\mathbf{g}_t]_j = \frac{w_j x_t}{\sum_{k=1}^{p} w_k}(g - y_{t_0}),$$
$$w_j = \exp(-\frac{1}{2}h_j(x_t - c_j)^2),$$

where h_j is the bandwidth and c_j is the center of the Gaussian kernels [16], [17].

The path integral formulation [8] is used to solve the optimal control problem. This formulation of optimal control, along with DMP equations, leads to the notion of a path cost $S(\tau)$ defined below:

$$\begin{aligned} S(\tau_{i,m}) =& \phi_N(\tau_{i,m}) + \sum_{j=i}^{N-1} q_j(\tau_{i,m}) \\ &+ \frac{1}{2}\sum_{j=i+1}^{N-1} (\theta + M_{j,m}\epsilon_{j,m})^T R(\theta + M_{j,m}\epsilon_{j,m}) \\ M_{j,m} =& \frac{R^{-1}\mathbf{g}_{j,m}\mathbf{g}_{j,m}^T}{\mathbf{g}_{j,m}^T R^{-1}\mathbf{g}_{j,m}}, \end{aligned} \tag{6}$$

where the index m in the equations above, is the index of the chosen trajectory. Moreover, $\mathbf{g}_{j,m}$ denotes the basis functions of the DMP for trajectory $\tau_{i,m}$ at time step j.

Additionally, the path integral framework provides a convenient probability distribution over trajectories, using the path costs defined in (6). The probability of a specific trajectory (or portion of a trajectory) $\tau_{i,m}$ from the set of sampled trajectories is denoted by $P(T = \tau_{i,m})$. The probability distribution over the given set of all K sampled trajectories, denoted by $\Omega = \{\tau_{0,1}, ..., \tau_{0,m}, ..., \tau_{0,K}\}$, has the following form:

$$P(T = \tau_{i,m}|\Omega) = \frac{e^{-\frac{1}{\lambda}S(\tau_{i,m})}}{\sum_{k=1}^{K} e^{-\frac{1}{\lambda}S(\tau_{i,k})}},$$

Lastly, the optimal control is solved by iteratively updating the parameter vector θ until convergence is reached using the following increments $\delta\theta_i$:

$$\delta\theta_i = \sum_{k=1}^{K} P(\tau_{i,k}) M_{i,k}\epsilon_{i,k} \tag{7}$$

In the next section, we introduce the statement of IRL problem. Note that in the remainder of the paper, we only deal with complete trajectories and we will replace the notation $\tau_{0,m}$ with τ_m.

C. Inverse Reinforcement Learning

The IRL problem addressed in this paper is recovering the cost function (2) given the dynamics of the system in equation (4) and the optimal policy parameter θ^*. To approach this problem, we model the path cost function as a weighted linear combination of a set of features (as we will describe in detail in the remaining of this section.) This leads to $S(\tau) = \boldsymbol{\beta}^{*T}\Phi(\tau)$, where $\Phi(\tau)$ is the value of the feature for trajectory τ and $\boldsymbol{\beta}^*$ are the weights of the features. Equivalently, this leads to a weighted linear form of $R(\tau)$ written as $R(\tau) = \boldsymbol{\beta}^{*T}\tilde{\Phi}(\tau)$.

Given this assumption, the inverse reinforcement learning problem is:

$$\hat{\boldsymbol{\beta}} = \arg\max_{\boldsymbol{\beta}} P(\theta^*|\boldsymbol{\beta}, \Omega) \tag{8}$$

The optimization (8) attempts to find the most likely set of weights $\hat{\boldsymbol{\beta}}$ which define the cost function, given observation of the optimal policy, parameterized by θ^*, and a set of sampled trajectories Ω. As we will see in Section III, the optimization in (8) is really composed of two problems. The first problem is finding the parameters $\hat{\boldsymbol{\beta}}$ given the optimal policy and given a set of sampled trajectories. The second problem is finding a set of sampled trajectories, which would lead to better estimates of the cost function parameters $\hat{\boldsymbol{\beta}}$.

To write the path cost as $S(\tau) = \boldsymbol{\beta}^T\Phi(\tau)$, we can consider the costs q_i and ϕ_N and the matrix R to be parameterized as $q_i = \boldsymbol{\beta}_q^T\boldsymbol{\psi}_i$, $\phi_N = \beta_{\phi_N}\psi_N$ and $R = \beta_R\hat{R}$, for some known features $\boldsymbol{\psi}_i, \psi_N$ and known matrix $\hat{R}$. (Note that in general and unless otherwise noted, if ϕ is a feature, and $\tau = (\mathbf{x}_0, ..., \mathbf{x}_N)$ is a trajectory, then $\phi(\tau) = \sum_{i=0}^{N}\phi(\mathbf{x}_i)$.)

The cost for a trajectory $\tau_m = (\mathbf{x}_0, ..., \mathbf{x}_N)$ can be written as $S(\tau_m) = \boldsymbol{\beta}^T\boldsymbol{\Phi}(\tau_m)$, where $\boldsymbol{\beta}^T = [\boldsymbol{\beta}_q^T, \beta_R, \beta_{\phi_N}]$, and:

$$\Phi(\tau_m) = \begin{pmatrix} \sum_{i=0}^{N-1}\boldsymbol{\psi}_i(\tau_m) \\ \frac{1}{2}\sum_{i=1}^{N-1}(\theta + M_{i,m}\epsilon_{i,m})^T\hat{R}(\theta + M_{i,m}\epsilon_{i,m}) \\ \psi_N(\mathbf{x}_N) \end{pmatrix}.$$

Equivalently, we can write $R(\tau_m) = \boldsymbol{\beta}^T\tilde{\Phi}(\tau_m)$ where $\boldsymbol{\beta}^T = [\boldsymbol{\beta}_q^T, \beta_R, \beta_{\phi_N}]$ and:

$$\tilde{\Phi}(\tau_m) = \begin{pmatrix} \sum_{i=0}^{N-1}\boldsymbol{\psi}_i(\tau_m) \\ \frac{1}{2}\sum_{i=1}^{N-1} u_i^T\hat{R}u_i, \\ \psi_N(\mathbf{x}_N) \end{pmatrix}.$$

In the next section, we introduce our method of approach to this problem. In particular, we discuss how to solve the ML estimation (8) for a given Ω, and also how to construct sets of sampled trajectories Ω.

III. METHOD OF APPROACH

In this section, we will describe our solution approach to the IRL problem described in II-C. As discussed, we propose finding the parameter $\boldsymbol{\beta}$ by performing the maximization in (8) given the nominal θ^*. We propose using a nominal trajectory τ^* instead of the parameter θ^*, as we will show that there is a one-to-one correspondence between the parameters θ of the DMP equations, and a generated nominal trajectory. The nominal trajectory τ^* is the resulting trajectory from the DMP equations where $\theta = \theta^*$ and the noise ϵ is set to zero. To show the one-to-one correspondence note the following. If two parameter vectors θ_1 and θ_2 lead to the same nominal trajectory τ, then we must have that $\mathbf{g}_i^T\theta_1 = \mathbf{g}_i^T\theta_2$. This does not occur if $\theta_1 \neq \theta_2$, since the vector $\mathbf{g}$ forms a basis function, and thus no two different parameters can lead to the same nominal trajectory. Therefore, we perform the following maximization instead of the maximization in (8):

$$\max_{\boldsymbol{\beta}} P(T = \tau^*|\boldsymbol{\beta}, \Omega) = \max_{\boldsymbol{\beta}} \frac{e^{-\frac{1}{\lambda}(\boldsymbol{\beta}^T\Phi(\tau^*))}}{\sum_{k=1}^{K} e^{-\frac{1}{\lambda}(\boldsymbol{\beta}^T\Phi(\tau_k))}}, \quad (9)$$

where τ^* is also included in the set of all trajectories, i.e. $\tau^* \in \Omega = \{\tau_1, ..., \tau_K\}$.

The IRL approach described by equation (9), generalizes to all parameterized policies that can be written in the form of the dynamic equations (4), and where the policies are defined by a parameterized combination of some basis functions, similar to what we see in the case of the DMPs.

The maximization in (9) is done using a set of K sampled trajectories. Note that in order to have an exact probability distribution over trajectories, we would need to sample all possible trajectories, and replace the sum in the denominator of equation (9) with an integral. Instead, to have an implementable version of this probability, we form an approximate probability by sampling trajectories and summing over all the sampled trajectories.

In order to solve for the maximizing $\boldsymbol{\beta}$ in equation (9), we are faced with two problems. First, given the nominal trajectory τ^*, and K overall sampled trajectories $\tau_1, ..., \tau_K$, we have to estimate the optimal parameters of the distribution. We will approach this problem using ML estimation. Secondly, we require K sample trajectories that provide us with an accurate final estimate of the parameters $\boldsymbol{\beta}$. The choice of the K sampled trajectories significantly affects the estimation of the $\boldsymbol{\beta}$ parameters. Moreover, choosing effective sampled trajectories could be very hard in different applications. We will approach this problem using the idea of cost updates which iteratively samples trajectories, and improves the estimates in each step of our simulation.

A. Iterative Scaling Algorithm

Solving the maximization (9) directly is not easy, due to the normalizing denominator. However, we can use the *Iterative Scaling Algorithm* [10], [11], which solves for parameters $\boldsymbol{\beta}$ iteratively. This algorithm solves for the parameters $\boldsymbol{\beta}$ to ensure that the empirical mean of the features equals the probabilistic mean of them. More precisely, given observed nominal trajectory τ^*, the algorithm updates the parameters of a Gibbs distribution according to:

$$\beta_{t+1,n} = \beta_{t,n} - ln\frac{\phi_n(\tau^*)}{E_{q_{\boldsymbol{\beta}_t}}\phi_n}, \quad (10)$$

$$E_{q_{\boldsymbol{\beta}_t}}\phi_n = \sum_{k=1}^{K} P(T = \tau_k|\beta_t)\phi_n(\tau_k). \quad (11)$$

The subscript n denotes the n-th element of the vector in the above equations. Note that equation (10) is updating the parameters $\boldsymbol{\beta}$ in order to produce a Gibbs distribution with a feature average value equal to the empirical feature average.

B. Cost Updating Algorithm

The idea of a cost updating algorithm stems from the fact that the sum over the set of sampled trajectories Ω is merely an approximation of the denominator of the probability distribution (9). Thus, the cost function recovered from the ML estimation will also be an approximate cost function. In other words, the optimal trajectory with respect to the estimated cost function is not necessarily equal to the nominal trajectory τ^*.

Intuitively, this effect is an outcome of poor sampling of the space of trajectories. When the sampled trajectories all have rather high costs with respect to the true cost function, then even a bad approximation of the cost function can promise the minimum cost for the observed trajectory. However, if the set of sampled trajectories, also consists of low cost trajectories with respect to the true cost function, inaccurate estimates of the cost function might cause the observed optimal trajectory not to have the smallest cost, thus a better estimate of the cost function is obtained. To overcome this issue, we will introduce an algorithm, which iteratively samples trajectories, and updates an estimate of the cost function. In this algorithm, we will use $\Omega_0 = \{\tau_1, ..., \tau_K\}$ to denote the set of all initial sampled trajectories, which also includes the nominal trajectory τ^*. Moreover, we will use $\hat{\tau}_t$ to denote a sampled trajectory at iteration t to be added to the set of all trajectories, and we will define $\Omega_{t+1} = \Omega_t \cup \{\hat{\tau}_{t+1}\}$. With these definitions and explanations, we will now introduce the following iterative algorithm:

Algorithm: Given nominal observed trajectory τ^*, the initial set of all trajectories $\Omega_0 = \{\tau_1, ..., \tau_K\}$, an initial value λ_0 for the λ temperature, and an unknown cost function $S(\tau) = \boldsymbol{\beta}^{*T}\Phi(\tau)$, do the following for each iteration t until termination:

1) Solve the ML $\hat{\boldsymbol{\beta}}_t = \arg\max_{\boldsymbol{\beta}} P(T = \tau^*|\boldsymbol{\beta}, \Omega_t)$.
2) Solve optimal control to find: $\hat{\tau}_{t+1} = \arg\min_{\tau} \hat{\boldsymbol{\beta}}_t^T \Phi(\tau)$.
3) Add trajectory to the set of all trajectories: $\Omega_{t+1} = \Omega_t \cup \{\hat{\tau}_{t+1}\}$.
4) Let $\Delta_t = |\hat{\boldsymbol{\beta}}_t^T \Phi(\tau^*) - \min_{\tau \in \Omega_t \setminus \{\tau^*\}} \hat{\boldsymbol{\beta}}_t^T \Phi(\tau)|$. If the change in the improvement is small, i.e. if $\left|\frac{\Delta_{t+1} - \Delta_t}{\Delta_{t+1}}\right| < \delta$, then $\lambda_{t+1} \leftarrow \lambda_t/(1 + \kappa)$, for $\kappa > 0$.
5) If $\lambda_{t+1} < \lambda_{thr}$ then **terminate**, otherwise $t \leftarrow t + 1$ and go back to step 1.

The above algorithm, explores the regions in the space where low cost trajectories exist under an estimated cost function. If these low cost trajectories are not close to the nominal trajectory τ^*, the addition of these trajectories $\hat{\tau}$ to the set of all sampled trajectories will improve the estimation of the cost function. Moreover, we are utilizing a simple annealing approach by reducing the temperature, which showed to improve the convergence rate in simulations.

The intuition behind this algorithm is the following. If the estimated cost function is far from the true cost function, there could be optimal trajectories derived from this estimated cost function that actually have a lower cost compared to the observed optimal trajectory. However, after another iteration, the algorithm makes sure that a cost function is recovered that results in the observed trajectory being the most likely. To do so, the algorithm attempts to find a better estimate of the cost function.

C. Comparison with Existing Methods

Several methods for solving the IRL problem for discrete MDP and continuous state space systems have been introduced in the literature. Here we will provide a review of some of these methods, and we will discuss the relationship between our proposed algorithm and these works, and the pros and cons of the different approaches.

One of the early approaches to IRL was by [1]. This approach later led to the apprenticeship learning algorithm developed in [13]. These works use MDPs and a cost function that is a weighted linear combination of predefined features. There are clear similarities between our proposed algorithm and the algorithm developed in [13]. First of all, both are based on the idea of matching the empirical averages of the demonstrated optimal trajectories. In our algorithm, this is done in the iterative scaling algorithm, where we aim to find a maximum entropy distribution which has an average equal to the empirical average. Authors in [13] also try to find a policy with averages close to the empirical average. Moreover, there is an intuitive similarity between the maximum entropy ML estimation in our algorithm and the max-margin in [13]. Both these methods try to estimate feature weights that will in some sense ensure the lowest cost for the optimal trajectory, and higher costs for the other sampled trajectories, thus introducing some margin. Additionally, both algorithms solve an optimal control problem in order to find new sample policies/trajectories to add to the set of all policies/trajectories.

The algorithm developed in [13], does not make use of a particular forward optimal control algorithm, and relies on discretization of the space to solve the forward optimal control problem. The discretization, for high-dimensional continuous problems, comes at a cost of higher computational complexity, since the algorithms to solve the optimal control problem are not very efficient. In contrast, our algorithm deals directly with continuous state spaces, and relies on the path integral formulation, which has shown promise to solve high dimensional optimal control problems with a reasonable computational complexity. Moreover, the iterative scaling algorithm, implemented in Matlab, was not computationally expensive. Thus we believe the proposed approach could be applicable to high dimensional problems that were thought to be infeasible before.

In addition to the discussed approaches, more recently an IRL approach [12] based on [8] has been developed for continuous state spaces. This approach, does not require solving the optimal control problem, however, it does suffer from the trajectory sampling problem that we have discussed in our paper. In order to approximate an integral over all trajectories, a summation over only a finite set of trajectories is used. Therefore, the selection of this set of trajectories highly affects the result of the IRL problem. Moreover, as the authors in [12] also point out, the optimization problem done in this algorithm is not fully studied, and more efficient algorithms have to be developed. Also, the performance of the algorithm depends on the heuristic selection of regularization parameters and objective function weighting, which would influence the result of the experiment.

IV. Initial Evaluation

We performed a simulation similar to the one in [12]. We generated six features, each feature being a sum of many Gaussians with random means and covariance matrices. We denote the features by ϕ_i and the wights by β_i. We used a 2-D point mass control system defined by the following DMP equations:

$$\ddot{x} = \frac{1}{m}(-b\dot{x} + u),$$
$$u = m\ddot{x}_d + b\dot{x} + k_P(x_d - x) + k_D(\dot{x}_d - \dot{x}),$$

where the desired signal to follow x_d is the output of the DMP y, i.e. $x_d = y, \dot{x}_d = \dot{y}$ and $\ddot{x}_d = \ddot{y}$. These equations are discretized and integrated forward starting from the origin in a 2-D space. The goal is to move the point mass to the final point $y = [1, 1]$, with minimum cost. We defined the cost function to be the following:

$$S(\tau) = \boldsymbol{\beta}^{\mathbf{T}}\Phi(\tau) + C_1(\mathbf{x}_{t_N} - [1,1]^T)^T(\mathbf{x}_{t_N} - [1,1]^T) + C_2(\dot{\mathbf{x}}_{t_N}^T\dot{\mathbf{x}}_{t_N}) + \frac{C_3}{2}\sum_{i=1}^{N-1}(\theta + M_{t_i,j}\epsilon_{t_i,j})^T\hat{R}(\theta + M_{t_i,j}\epsilon_{t_i,j}),$$

where the first term $\boldsymbol{\beta}^{\mathbf{T}}\Phi(\tau) = \sum_{i=0}^{N}\boldsymbol{\beta}^{\mathbf{T}}\Phi(\mathbf{x}_{\mathbf{t_i}})$ reflects the cost of the trajectory due to the sum of Gaussian features. The second and third term enforce costs on terminal position and velocity. Lastly, the fourth term enforces a cost on the inputs. In this simulation the constants C_1, C_2 and C_3 are known, and we will only be estimating the weights $\boldsymbol{\beta}$.

We applied our algorithm to an observed optimal policy. To apply our algorithm, we sampled only 10 sample trajectories, with an eye towards future high dimensional applications, where dense sampling of the space is not practical. Fig. 1 demonstrates the recovered cost function, and the best candidate trajectory with respect to the final estimated cost function. For this simulation, we

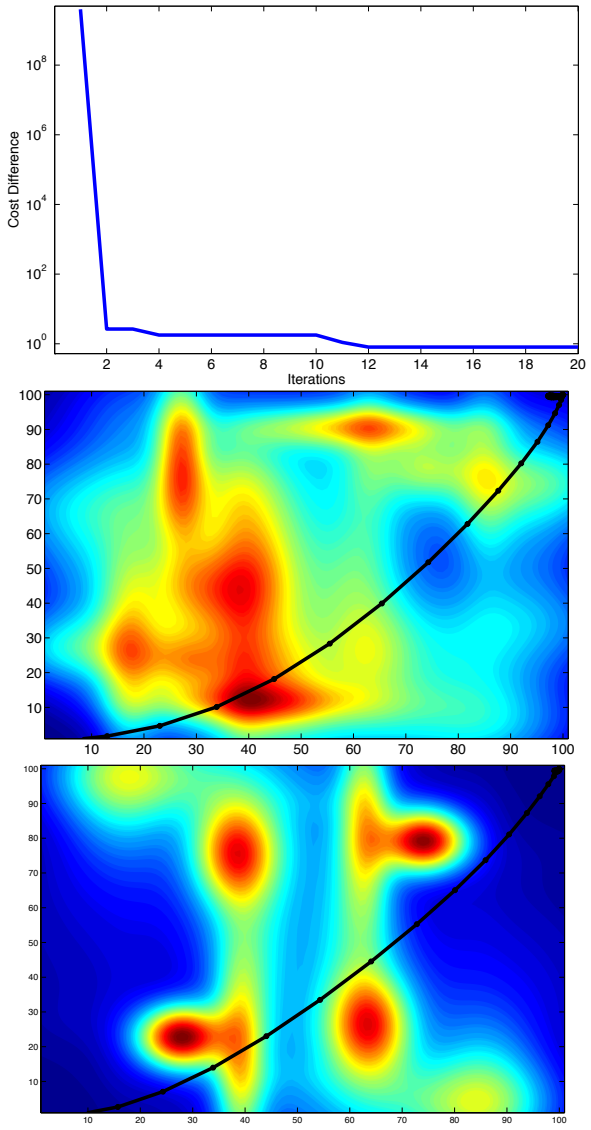

Fig. 1. **Top:** Plot of $\Delta_t^G = \boldsymbol{\beta}^{*T}\phi(\hat{\tau}_t^*) - \boldsymbol{\beta}^{*T}\phi(\tau^*)$ versus the number of iterations t, where $\hat{\tau}_t^* = \arg\min_{\tau \in \Omega_t \setminus \{\tau^*\}} \hat{\boldsymbol{\beta}}_t^T \phi(\tau)$. **Middle:** The true cost function, and the observed nominal trajectory, **Bottom:** Recovered cost function and best candidate sample trajectory

also utilized some of the code provided in *"http://www-clmc.usc.edu/Resources/Software"*.

A. Algorithm Performance

In order to evaluate the effectiveness of our algorithm, we evaluate the global convergence of our algorithm in simulations. To do so, first define the best candidate trajectory under the current cost estimate as $\hat{\tau}_t^* = \arg\min_{\tau \in \Omega_t \setminus \{\tau^*\}} \hat{\boldsymbol{\beta}}_t^T \Phi(\tau)$ at every iteration, and subsequently define $\Delta_t^G = \boldsymbol{\beta}^{*T}\Phi(\hat{\tau}_t^*) - \boldsymbol{\beta}^{*T}\Phi(\tau^*)$. Verifying $\lim_{t\to\infty} \Delta_t^G = 0$ ensures global convergence to a cost function which leads to a policy equal to the observed nominal policy. We will evaluate this expression in Fig. 1.

V. Discussion and Future Work

We proposed an algorithm for inverse reinforcement learning in a framework where the policy and the cost function were both a weighted linear combination of some known basis functions, and where the input cost was inversely proportional to the noise variance. We have shown using simulations that this approach improves the estimates of the cost function iteratively. These results should be considered preliminary. A formal comparison between our approach and existing IRL approaches, and a rigorous evaluation of the performance of the algorithm are topics of future work.

VI. Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant Nos. 0931871 and 0955088. The authors gratefully acknowledge Dr. Seth Hutchinson, Javad Ghaderi, Abdullah Akce, Miles Johnson and Samuel McCarthy for helpful comments.

References

[1] A. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000, pp. 663–670.

[2] E. Todorov, "Optimality principles in sensorimotor control," *Nature neuroscience*, vol. 7, no. 9, pp. 907–915, 2004.

[3] K. Kording and D. Wolpert, "The loss function of sensorimotor learning," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, no. 26, p. 9839, 2004.

[4] M. Krstic and P. Tsiotras, "Inverse optimal stabilization of a rigid spacecraft," *Automatic Control, IEEE Transactions on*, vol. 44, no. 5, pp. 1042–1049, 1999.

[5] P. Abbeel, A. Coates, and A. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, 2010.

[6] H. Kappen, "An introduction to stochastic control theory, path integrals and reinforcement learning," *Cooperative Behavior in Neural Systems*, vol. 887, pp. 149–181, 2007.

[7] B. van den Broek, W. Wiegerinck, and B. Kappen, "Graphical model inference in optimal control of stochastic multi-agent systems," *Journal of Artificial Intelligence Research*, vol. 32, no. 1, pp. 95–122, 2008.

[8] E. Theodorou, J. Buchli, and S. Schaal, "A Generalized Path Integral Control Approach to Reinforcement Learning," *Journal of Machine Learning Research*, vol. 11, pp. 3137–3181, 2010.

[9] E. Todorov, "Efficient computation of optimal actions," *Proceedings of the National Academy of Sciences*, vol. 106, no. 28, p. 11478, 2009.

[10] J. Darroch and D. Ratcliff, "Generalized iterative scaling for log-linear models," *The Annals of Mathematical Statistics*, vol. 43, no. 5, pp. 1470–1480, 1972.

[11] S. Chen and R. Rosenfeld, "A survey of smoothing techniques for ME models," *Speech and Audio Processing, IEEE Transactions on*, vol. 8, no. 1, pp. 37–50, 2002.

[12] M. Kalakrishnan, E. Theodorou, and S. Schaal, "Inverse Reinforcement Learning with PI 2," in *The Snowbird Workshop, submitted to*, 2010.

[13] P. Abbeel and A. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 1.

[14] K. Dvijotham and E. Todorov, "Inverse Optimal Control with Linearly-Solvable MDPs," in *Proceedings of the Interntional Conference on Machine Learning*. Citeseer, 2010.

[15] B. Ziebart, A. Maas, J. Bagnell, and A. Dey, "Maximum entropy inverse reinforcement learning," in *Proc. AAAI*, 2008, pp. 1433–1438.

[16] A. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," *Advances in neural information processing systems*, pp. 1547–1554, 2003.

[17] S. Schaal and C. Atkeson, "Constructive incremental learning from only local information," *Neural Computation*, vol. 10, no. 8, pp. 2047–2084, 1998.