On the Minimization of the Policy Gradient in Inverse Reinforcement Learning

Matteo Pirotta matteo.pirotta@polimi.it
Marcello Restelli marcello.restelli@polimi.it
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy

Abstract

Inverse Reinforcement Learning (IRL) deals with the problem of recovering the reward function optimized by an expert given a set of demonstrations of the expert's policy. Most IRL algorithms need to repeatedly compute the optimal policy for different reward functions. This paper proposes a new IRL approach that allows to recover the reward function without the need of solving any "direct" RL problem. The idea is to find the reward function that minimizes the gradient of a parameterized representation of the expert's policy. In particular, when the reward function can be represented as a linear combination of some basis functions, we will show that the aforementioned optimization problem can be efficiently solved. We present a preliminary empirical evaluation of the proposed approach on a multidimensional version of the Linear-Quadratic Regulator (LQR) both in the case where the parameters of the expert's policy are known and in the (more realistic) case where the parameters of the expert's policy need to be inferred from the expert's demonstrations.

1. Introduction

Markov Decision Processes (MDPs) are an effective mathematical tool in modeling decision making in uncertain dynamic environments, where tasks are simply defined by providing a reward function. However, in many real-world problems, even the specification of the reward function can be problematic and it is easier to provide demonstrations from a desired policy. Inverse Reinforcement Learning (IRL) aims to find the reward function that is (implicitly or explicitly) optimized by the demonstrated policy. Once the reward function has been recovered, it can be used either to generalize the expert's behavior also to state space regions not covered by the demonstrations or to compute a new policy whenever the system dynamics change. In the last decade, many IRL algorithms have been proposed (see (Zhifei and Meng Joo, 2012) for a recent survey), most of which are approaches that require the MDP model and/or need to iteratively compute the optimal policy of the MDP obtained by considering intermediate reward functions. Since an accurate solution to the IRL problem may require many iterations, when the optimal policy for the MDP cannot be efficiently computed, these IRL approaches are not practical. For this reason, some recent works have proposed model-free IRL approaches that do not require to solve MDPs. Boularias et al. (2011) propose a model-free version of the Maximum Entropy IRL approach (Ziebart et al., 2008) that minimizes the *relative entropy* between the empirical distribution of the state-action trajectories demonstrated by the expert and their distribution under the learned policy. Even if this approach avoids the need of solving MDPs, it requires sampling trajectories according to a non-expert policy. The approach proposed by Dvijotham and Todorov (2010) does not need to solve many MDPs, but it can be applied only to *linearly solvable* MDPs. Classification-based approaches (SCIRL and CSI) (Klein et al., 2012, 2013) can produce near-optimal results when accurate estimation of the feature expectations can be computed and heuristic versions have been proved effective even when a few demonstrations are available. While SCIRL is limited to linearly parametrized reward functions, CSI can deal with nonlinear functions. However, both the algorithms require the expert to be deterministic and need to use heuristic approaches in order to learn with the only knowledge of expert's trajectories. Neu and Szepesvári (2007) have proposed an IRL approach that can work even with non-linear reward function parametrizations, but it needs to compute the optimal action-value function for each intermediate parametrization.

The approach proposed in this paper is based on the following observation: given a parametric representation of the expert's policy, its policy gradient computed according to the "true" reward function is zero. So, given a parametrized representation of the reward function, we solve the IRL problem by searching the reward function that minimizes some norm of the policy gradient. We show how this result can be obtained using a model-free approach that is based only on the data provided by the expert's demonstrations and no MDP needs to be solved. Although no assumption on the reward model is needed, for the case of linear parametrization we provide an efficient algorithm for computing the (unique) solution to the optimization problem. Empirical results on the Linear-Quadratic Regulator (LQR) test case allow to evaluate the effectiveness of the proposed method in recovering the parameters of the reward function optimized by the expert both when the expert's policy is known and when it has to be estimated only by demonstrated trajectories.

2. Preliminaries

A Markov Decision process without reward MDP$\backslash\mathcal{R}$ is defined by $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, D\rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is a Markovian transition model where $\mathcal{P}(s'|s,a)$ defines the transition density between state s and s' under action a, $\gamma \in [0,1)$ is the discount factor for future rewards, and D is the distribution of the initial state. The control policy is characterized by a density distribution $\pi(\cdot; s)$ that specifies for each state s the density distribution over the action space $\mathcal{A}$. We observe the behavior of an expert that follows a policy π^E that is optimal w.r.t. some reward function $\mathcal{R}^E$. We assume that $\mathcal{R}^E$ can be represented through a linear or non-linear function $\mathcal{R}(s,a;\boldsymbol{\omega})$, where $\boldsymbol{\omega} \in \mathbb{R}^q$. In the case of linear reward parametrization, as done by Syed and Schapire (2007), we will restrict (without loss of generality) the expert weights $\boldsymbol{\omega}^E$ to belong to the unit $(q-1)$-simplex $\Delta^{q-1} = \{\boldsymbol{\omega} \in \mathbb{R}^q : \|\boldsymbol{\omega}\|_1 = 1 \wedge \boldsymbol{\omega} \succeq 0\}$ (where $\succeq$ denotes the component-wise inequality), We consider infinite horizon problems where the future rewards are exponentially discounted with γ. The expected discounted reward of a policy starting from a state drawn from D is:

$$J_D(\pi) = \mathop{\mathbb{E}}_{\substack{s_0 \sim D \\ a_t \sim \pi,\ s_t \sim \mathcal{P}}} \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t; \boldsymbol{\omega})\right] = \int_{\mathcal{S}} d_\mu^\pi(s) \int_{\mathcal{A}} \pi(a;s)\mathcal{R}(s,a;\boldsymbol{\omega})\mathrm{d}a\mathrm{d}s,$$

where d_μ^π is the γ–discounted feature state distribution for starting state distribution D (Sutton et al., 1999). In this work we limit our attention to parametrized *differentiable* policies $\pi(a; s, \boldsymbol{\theta})$, where $\boldsymbol{\theta} \in \mathbb{R}^d$. Where possible we will use the compact notation $\pi_{\boldsymbol{\theta}}$. For sake of notation, it is useful to introduce differential operators. The symbol $D_x f(x)$ denotes the derivative of function f w.r.t. variable x. When both the function and its input variable are vectors, the derivative represents the Jacobian matrix. The gradient of a scalar function is the transpose of the derivative and it will be denoted by $\nabla_x f(x)$.

3. Exact Expert's Policy Parameters

In the first scenario we consider to know the expert's policy π^E. Having access to the explicit formulation of the parametric policy, we can use gradient information in order to derive a new IRL algorithm that does not require to solve the forward model.

Standard policy gradient approaches require the policy to be stochastic in order to get rid of the knowledge of the transition model (Sutton et al., 1999). When the expert's policy is deterministic the model must be available or the policy must be forced to be stochastic.[1] For continuous state-action domains the latter approach can be easily implemented by adding zero-mean Gaussian noise. Instead the Gibbs model is suited for discrete actions because the stochasticity can be regulated by varying the temperature parameter.

Given a parametric (linear or non linear) reward function $\mathcal{R}(s, a; \boldsymbol{\omega})$, we can compute the associate policy gradient

$$\nabla_{\boldsymbol{\theta}} J\left(\pi^E_{\boldsymbol{\theta}}, \boldsymbol{\omega}\right) = \int_{\mathcal{S}} d_\mu^{\pi^E_{\boldsymbol{\theta}}}(s) \int_{\mathcal{A}} \nabla_{\boldsymbol{\theta}} \pi^E(a; s, \boldsymbol{\theta}) Q^\pi(s, a; \boldsymbol{\omega}) \mathrm{d}a \mathrm{d}s.$$

We assume to have an analytic description of the expert's policy, but we have a limited set of demonstrations of the expert's behavior in the environment. Let denote by $\mathcal{D} = \{\tau_i\}_{i=1}^N$ the set of expert's trajectories. Then, the gradient can be estimated off-line using the N trajectories and any standard policy gradient algorithm: REINFORCE, GPOMDP, natural gradient or eNAC (Peters and Schaal, 2008).

If the policy performance $J(\pi, \boldsymbol{\omega})$ is differentiable w.r.t. the policy parameters $\boldsymbol{\theta}$ and the expert $\pi^E_{\boldsymbol{\theta}}$ is optimal w.r.t. a parametrization $\mathcal{R}(s, a, \boldsymbol{\omega}^E)$, the associated policy gradient is identically equal to zero. Clearly, the expert's policy $\pi^E_{\boldsymbol{\theta}}$ is a stationary point for $J\left(\pi, \boldsymbol{\omega}^E\right)$.

The Gradient IRL (GIRL) algorithm aims at finding a stationary point of $J\left(\pi^E_{\boldsymbol{\theta}}, \boldsymbol{\omega}\right)$ w.r.t. the reward parameter $\boldsymbol{\omega}$, that is, for any $p, q \geq 1$

$$\boldsymbol{\omega}^A = \arg\min_{\boldsymbol{\omega}} \, \mathcal{C}_p^q\left(\pi^E_{\boldsymbol{\theta}}, \boldsymbol{\omega}\right) = \arg\min_{\boldsymbol{\omega}} \, \frac{1}{q} \left\| \nabla_{\boldsymbol{\theta}} J\left(\pi^E_{\boldsymbol{\theta}}, \boldsymbol{\omega}\right)\right\|_p^q,$$

One of the key properties of the GIRL algorithm is the convexity of the objective function whenever the parametric reward model is convex w.r.t. $\boldsymbol{\omega}$. This means that the optimization problem can be solved using any standard convex optimization approach.

As mentioned before, when the expert is optimal w.r.t. the parametric reward, the minimum of $\mathcal{C}_p^q\left(\pi^E_{\boldsymbol{\theta}}, \boldsymbol{\omega}\right)$ is attained by the expert's weights $\boldsymbol{\omega}^E$. Instead there are several reasons for which the expert may be not optimal: I) the expert is optimizing a reward function that cannot be represented by the chosen reward parametrization; II) the expert exhibits a suboptimal policy; III) the samples available to estimate the gradient are not sufficient to obtain a reliable estimate. In the suboptimal scenario, the solution $\boldsymbol{\omega}^A$ represents the minimum norm gradient, i.e., the reward that induces the minimum change in the policy parameters. In other words, GIRL tries to provide the reward weights in the chosen space that better explain the behavior of the expert's policy. Note that the result $\boldsymbol{\omega}^A$ is reliable as long as the norm of the gradient is sufficiently small[2], otherwise the optimal policy associated to reward weights $\boldsymbol{\omega}^A$ can be arbitrary different from the expert's one.

1. Recently a deterministic version of the policy gradient theorem has been provided in (Silver et al., 2014). However, it cannot be directly applied in this framework because it requires a stochastic policy for exploration.
2. The threshold between reliable and unreliable results is problem dependent.

4. Linear Reward Parametrization

In this section we reformulate the GIRL algorithm in the case of linear parametrization of the reward function $\mathcal{R}(s, a; \boldsymbol{\omega})$. This interpretation comes from a multi–objective view of the IRL problem. With linear reward parametrization the expected policy performance is

$$J(\pi, \boldsymbol{\omega}) = \sum_{i=1}^{q} \boldsymbol{\omega}_i \underset{\substack{s_0 \sim D \\ a_t \sim \pi \; s_t \sim \mathcal{P}}}{\mathbb{E}} \left[\sum_{t=0}^{\infty} \gamma^t \phi_i(s_t, a_t) \right] = \sum_{i=1}^{q} \boldsymbol{\omega}_i \boldsymbol{J}_i(\pi), \tag{1}$$

where $\boldsymbol{J}(\pi)$ are the features expectations under policy π and basis functions $\boldsymbol{\Phi}(s, a)$. Equation (1) can be interpreted as a *weighted sum* of the objectives $\boldsymbol{J}(\pi)$. This view connects our approach to the search of the reward weights that make the expert Pareto optimal.

When the expert is not optimal w.r.t. any reward function, she lies outside the Pareto frontier and it is possible to identify a set of directions (ascent cone) that simultaneously increase all the objectives. Instead, when the expert belongs to the Pareto frontier (i.e., $\exists (convex) \boldsymbol{\alpha} : D_{\boldsymbol{\theta}} \boldsymbol{J}(\pi) \boldsymbol{\alpha} = \boldsymbol{0}$), the ascent cone is empty because any change in the parameters will cause the decrease of at least one objective. Geometrically, the gradient vectors related to the different reward features result coplanar and with contrasting directions.

By computing the hyperplane tangent to the Pareto frontier (identified by the individual gradients $\nabla_{\boldsymbol{\theta}} \boldsymbol{J}_i\left(\pi^E\right), i = 1, \ldots, q$), it is possible to recover the weights of the scalarization. Geometrically, they are the unitary normal to the hyperplane. Consider the expert to be optimal w.r.t. a linear combination of the objectives, then she lies on the Pareto frontier. By exploiting the local information given by the individual gradient w.r.t. the expert parametrization $D_{\boldsymbol{\theta}} \boldsymbol{J}\left(\pi^E\right)$ we can compute the tangent hyperplane and the associated scalarization weights.

Let $D_{\boldsymbol{\theta}} \boldsymbol{J}\left(\pi^E\right) = \left[\nabla_{\boldsymbol{\theta}} \boldsymbol{J}_1\left(\pi^E\right), \ldots, \nabla_{\boldsymbol{\theta}} \boldsymbol{J}_q\left(\pi^E\right)\right]$, we can generate q points in the objective space by considering the local changes in the objective function $\boldsymbol{J}(\pi)$ generated by the individual gradients. The q points are obtained through the Gram matrix of $D_{\boldsymbol{\theta}} \boldsymbol{J}\left(\pi^E\right)$

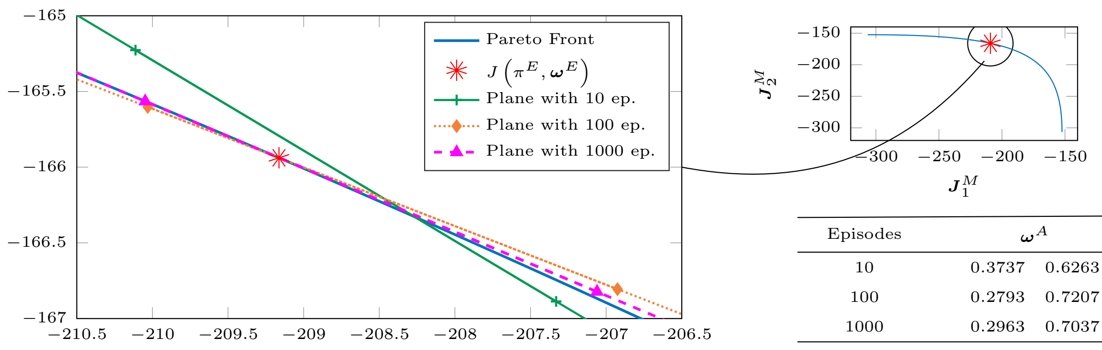
$$\boldsymbol{G} = \left(D_{\boldsymbol{\theta}} \boldsymbol{J}\left(\pi^E\right)\right)^{\mathrm{T}} D_{\boldsymbol{\theta}} \boldsymbol{J}\left(\pi^E\right).$$

Given q points in a q dimensional space, we can compute the associate hyperplane. However, we are not interested in an explicit representation of the hyperplane, but in the associated normal. The normal to a hyperplane is obtained by computing the null space of q points. Given the individual gradients, the complexity of obtaining the weights is $\mathcal{O}(q^2 d + q^3)$. Notice that the unit vector normal to the hyperplane coincides with the solution computed by GIRL. In the following, we will call this version of GIRL as PGIRL (Plane GIRL).

5. Approximated Expert's Policy Parameters

Consider to observe the expert's behavior through a set of trajectories $\{\tau_i\}_{i=0}^{N}$ of length M. In order to apply the same idea presented in the previous section we need to infer a parametric policy model from the data we have. This problem is a standard density estimation problem. Given a parametric policy model $\pi(a; s, \boldsymbol{\theta})$, a parametric density estimation problem can be defined as a maximum likelihood estimation problem

$$\widehat{\boldsymbol{\theta}}_{MLE} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{N \cdot M} \pi(a_i; s_i, \boldsymbol{\theta}).$$



Episodes	$\boldsymbol{\omega}^A$	
10	0.3737	0.6263
100	0.2793	0.7207
1000	0.2963	0.7037

(a) eNAC-PGIRL planes (b) Recovered weights

Figure 1: Behavior of the eNAC-PGIRL in 2D LQR. Figure (a) reports the planes in the objective space identified by the PGIRL algorithm with 10, 100 and 1,000 trajectories. This figure represents a zoom of the frontier around the current solution. The entire frontier and the value identified by the weights $\boldsymbol{\omega}^E = [0.3, \ 0.7]^\mathsf{T}$ are shown in the upper right figure. Table (b) contains the weights recovered by the algorithm, i.e., the normal unit vectors to the planes shown in Figure (a).

6. Experiments

In this section we provide a preliminary set of experiments in the well-known Linear Quadratic Regulator (LQR) problem (Peters and Schaal, 2008). These experiments are meant to be a proof of concept of our algorithm behavior. The LQG problem is defined by

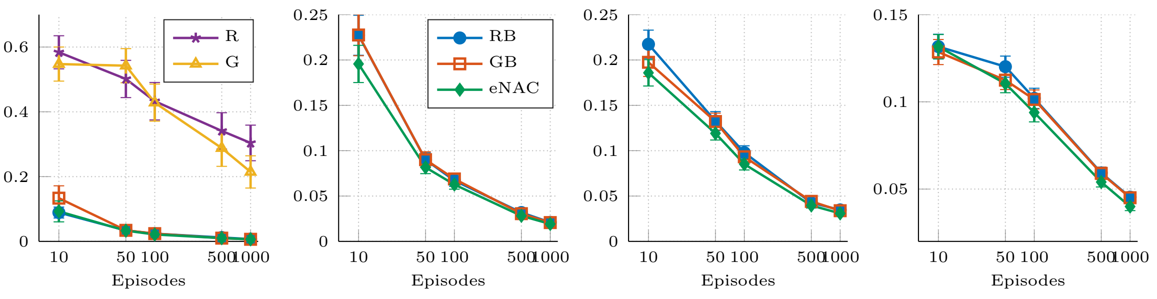
$$s_{t+1} = s_t + a_t, \quad a_t \sim \mathcal{N}(K \cdot s_t, \Sigma), \quad r_t = -{s_t}^\mathsf{T} Q s_t - {a_t}^\mathsf{T} R a_t$$

where s_t and a_t are n-dimensional column vector ($n = m$), $Q, R, K \in \mathbb{R}^{n \times n}$ and $\gamma = 0.9$. The policy is Gaussian with parameters $\boldsymbol{\theta} = vec\left(K^\mathsf{T}\right)$ and constant covariance ($\Sigma = I$).

We consider the multi-dimensional, multi-objective version of the problem provided in (Pirotta et al., 2015). The problem of minimizing the distance from the origin w.r.t. the i-th axis has been taken into account, considering the cost of the action over the other axes. Since the maximization of the i-th objective requires to have null action on the other axes, objectives are conflicting. We consider a linear parametrization of the reward function: $\mathcal{R}(s, a, \boldsymbol{\omega}) = \sum_{i=1}^{q} \boldsymbol{\omega}_i \sum_{t=0}^{\infty} -\gamma^t ({s_t}^\mathsf{T} Q_i s_t + {a_t}^\mathsf{T} R_i a_t)$.

Exact Expert's Policy Parameters. In the first test, we focus on the PGIRL approach where the gradient directions are computed using the eNAC algorithm (eNAC–PGIRL) in the 2D LQR domain. The goal is to provide a geometric interpretation of what the algorithm does. Figure 1 reports the planes and the associated weights obtained by the eNAC-PGIRL algorithm with different data set sizes. As the number of samples increases, the accuracy of the plane identified by the algorithm improves. With 1,000 trajectories, the plane is almost tangent to the Pareto frontier. The extrema of the segment that represents the plane are the point obtained from the Gram matrix (after a translation from the origin).

The next set of experiments deals with the accuracy and time complexity of the algorithms. We selected 5 problem dimensions: (2, 5, 10, 20). For each domain we selected 20 random expert's weight in the unitary simplex and we generated 5 different datasets. It is known that the contribute of the baseline for the gradient estimation is important



(a) 2D LQR (b) 5D LQR (c) 10D LQR (d) 20D LQR

Figure 2: Weight error. The L_∞ norm between the expert's and agent's weights are reported in figures for different problem and data set dimensions. Initial point is $s_0 = \mathbf{-10}$.

Episodes	PGIRL					GIRL				
	R	RB	G	GB	eNAC	R	RB	G	GB	eNAC
10	0.0010 ±0.0004	0.0008 ±0.0001	0.0008 ±0.0002	0.0008 ±0.0001	0.0008 ±0.0005	0.2573 ±0.0562	0.1344 ±0.0247	0.3479 ±0.0625	0.1156 ±0.0164	0.0728 ±0.0089
100	0.0074 ±0.0002	0.0072 ±0.0012	0.0072 ±0.0003	0.0076 ±0.0002	0.0071 ±0.0001	1.4700 ±0.3266	1.2871 ±0.1855	3.7931 ±0.9022	1.4640 ±0.1263	0.8013 ±0.0782
1000	0.0759 ±0.0048	0.0729 ±0.0047	0.0734 ±0.0043	0.0800 ±0.0046	0.0734 ±0.0044	12.2827 ±1.6163	16.3657 ±2.1328	40.3681 ±7.6919	16.9010 ±2.1461	9.2353 ±1.3415

Table 1: GIRL Computational Time (s) for 2D LQR. Table shows the average computational time spent by the algorithm for the generation of the results presented in Figure 2a.

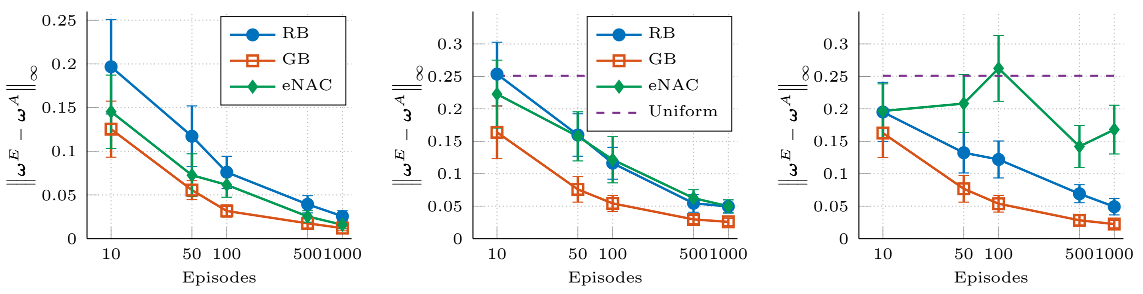
and cannot be neglected (Peters and Schaal, 2008). Consider Figure 2a, using plain REINFORCE (R) and GPOMDP (G) the GIRL algorithm is not able to recover the correct weights. Although the error decreases as the number of trajectories increases, the error obtained with $1,000$ trajectories is larger than the one obtained by the baseline–versions (RB and GB) with only 10 trajectories. For this reason we have removed the plain gradient algorithms from the other tests. Figures 2b–2d replicate the test for increasing problem dimensions. All the algorithms show a decreasing error as the number of samples increases, but no significant differences can be observed. From such results, we can conclude that, when the expert's policy is known, GIRL is able to recover a good approximation of the reward function even with a few sample trajectories.

In Tables 1 and 2 we show how the computational times of the different algorithms change as a function both of the number of available trajectories and of the number of reward parameters. PGIRL algorithm outperforms GIRL for any possible configuration. Recall that PGIRL has to compute a fixed number of gradients, equal to the reward dimensionality, while GIRL is an iterative algorithm. In our tests we have imposed a maximum number of function evaluations to 500 for the convex optimization algorithm. The results show that the difference in the time complexity exceeds two orders of magnitude[3] even in the 2D LQR (Table 1). Although, the best choice for linear reward parametrization is PGIRL, GIRL has the advantage of working even with non–linear reward parametrization. Table 2 shows that the difference increases with the problem dimensionality.

3. Clearly, the performance of the GIRL algorithm depends on the implementation of the convex algorithm and its parameters. Here we have exploited NLopt library (http://ab-initio.mit.edu/nlopt).

Episodes	PGIRL					GIRL				
	R	RB	G	GB	eNAC	R	RB	G	GB	eNAC
10	0.031 ±0.011	0.020 ±0.0011	0.020 ±0.002	0.021 ±0.002	0.037 ±0.009	10.1 ±1.5	8.4 ±1.1	46.1 ±6.2	8.0 ±1.3	3.2 ±0.3
100	0.173 ±0.007	0.168 ±0.001	0.166 ±0.003	0.170 ±0.003	0.179 ±0.003	74.0 ±19.8	70.2 ±9.8	249.7 ±73.6	50.4 ±3.3	32.0 ±6.5
1000	1.664 ±0.006	1.655 ±0.011	1.655 ±0.031	1.683 ±0.005	1.640 ±0.038	971.4 ±270.0	472.7 ±25.2	3613.0 ±1128.2	498.6 ±32.5	257.4 ±14.7

Table 2: GIRL Computational Time (s) for 5D LQR. Table shows the average computational time spent by the algorithm for the generation of the results presented in Figure 2b.



(a) Optimal Policy (linear) (b) Polynomial 3-deg (c) Gaussian RBFs

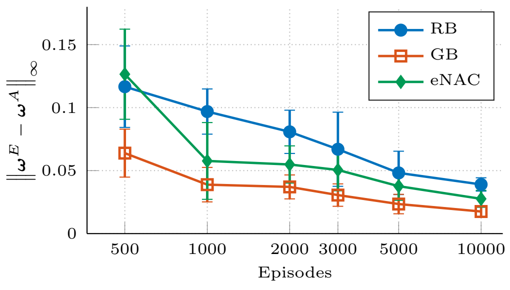
Figure 3: Weight error. The L_∞ norm between the expert's and agent's weights are reported in figures for different problem and data set dimensions.

Approximated Expert's Policy Parameters. In the following the parameters of the expert's policy are unknown and we have access only to expert's trajectories. In order to apply GIRL and PGIRL algorithms we have to learn a parametric policy from the data. We have mentioned in section 5 that this problem is a standard MLE problem.

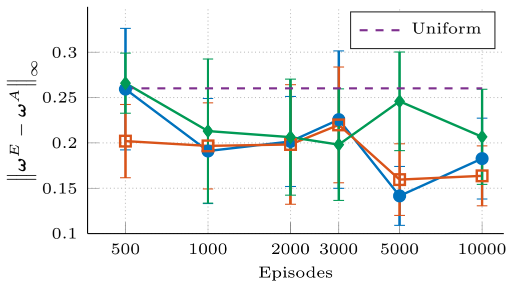
We start considering a standard 1–dimensional LQR problem. Under these settings the policy is a unimodal Gaussian $a_t \sim \mathcal{N}(ks, \sigma^2)$ and the reward is $r_t = -\boldsymbol{\omega}_1 s_t^2 - \boldsymbol{\omega}_2 a_t^2$. The reward is a compromise between fast convergence to the goal (which requires high action values) and low action values. Clearly this multi-objective scenario can be generalized to any dimension by considering the full cost matrix Q and R. The initial state is randomly selected in the interval $[-3, 3]$.

Since the action space is continuous and we have to learn stochastic policies, we limit our research among the class of Gaussian policies with fixed diagonal covariance to $2 \cdot \boldsymbol{I}$. We consider three different mean parametrizations: linear in the state (i.e., the optimal one Ks), with radial basis functions, and polynomial of degree 3.

We start considering to learn the weights for the optimal parametrization, in order to prove the correctness of the approach. Using MLE the weights are recovered almost exactly and the algorithms are able to learn in any configuration. Figure 3 reports the error achieved by the different policy parametrizations in the same settings described in the previous section. Even with a 3–degree polynomial approximation the learning process shown by the algorithm is smooth and the error decreases with the increase of the trajectory samples (Figure 3b). Finally, we consider the policy with 5 Gaussian RBFs uniformly placed in $[-4, 4]$ with an overlapping factor of 0.25. While REINFORCE and GPOMDP with



(a) Fitting original policy



(b) Fitting polynomial apx.

Figure 4: Weight error. The L_∞ norm between the expert's and agent's weights are reported for 5D LQR with different policy parametrizations.

baselines enjoy a smooth behavior, eNAC seems more affected by the estimation error in the policy parametrization and is not able to reduce the error even with $1,000$ trajectories.

To conclude, we have performed additional tests in a 5D multi–objective LQR described in the previous section. We start presenting the results in the case of the linear policy model (5 parameters). As shown in Figure 4a the policy is correctly recovered from data and the error decreases as the number of samples increases. Since the number of parameters required by the Gaussian RBFs grows exponentially with the problem dimension, we have tested only the polynomial approximation. In particular we have chosen a full polynomial of degree 2, that gives rise to 105 parameters. It can be noticed that the learning process with the full 2–degree polynomial approximation is more unstable. This problem is mainly due to the difficulty in estimating a high number of policy parameters with only a few samples available[4]. Some information about the shape of the policy is highly beneficial since it allows to build policy models with less parameters.

7. Conclusions and Future Works

We presented a novel inverse reinforcement learning algorithms that is able to recover the reward parameters by searching for the reward function that minimizes the policy gradient, that avoids to solve the "direct" MDP. While GIRL defines a minimum–norm problem for linear and non-linear reward parametrizations, PGIRL method is an efficient implementation of GIRL in case of linear reward functions. A preliminary evaluation of the algorithms has been performed in the LQR domain. Nevertheless, it will be important to evaluate the performance of the algorithms in more complex (and real) domains and compare the results against the state of the art. In addition, it would be interesting to investigate the theoretical guarantees that can be provided knowing the error in the gradient estimation.

4. This can be mainly due to our naive implementation of MLE algorithm.

References

Abdeslam Boularias, Jens Kober, and Jan Peters. Relative entropy inverse reinforcement learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, AISTATS 2011*, pages 182–189, 2011.

Krishnamurthy Dvijotham and Emanuel Todorov. Inverse optimal control with linearly-solvable mdps. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 335–342. Omnipress, 2010.

Edouard Klein, Matthieu Geist, Bilal Piot, and Olivier Pietquin. Inverse reinforcement learning through structured classification. In *Advances in Neural Information Processing Systems*, pages 1007–1015, 2012.

Edouard Klein, Bilal Piot, Matthieu Geist, and Olivier Pietquin. A cascaded supervised learning approach to inverse reinforcement learning. In *Machine Learning and Knowledge Discovery in Databases*, pages 1–16. Springer, 2013.

Gergely Neu and Csaba Szepesvári. Apprenticeship learning using inverse reinforcement learning and gradient methods. In *UAI 2007, Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, pages 295–302. AUAI Press, 2007.

Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682 – 697, 2008. Robotics and Neuroscience.

Matteo Pirotta, Simone Parisi, and Marcello Restelli. Multi-objective reinforcement learning with continuous pareto frontier approximation. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 2928–2934. AAAI Press, 2015.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin A. Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014*, volume 32 of *JMLR Proceedings*, pages 387–395. JMLR.org, 2014.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*, pages 1057–1063. The MIT Press, 1999.

Umar Syed and Robert E. Schapire. A game-theoretic approach to apprenticeship learning. In *Advances in Neural Information Processing Systems 20*, pages 1449–1456, 2007.

Shao Zhifei and Er Meng Joo. A survey of inverse reinforcement learning techniques. *International Journal of Intelligent Computing and Cybernetics*, 5(3):293–311, 2012.

Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008*, pages 1433–1438. AAAI Press, 2008.