Learning from Demonstrations: Is It Worth Estimating a Reward Function?

Bilal Piot[1,2], Matthieu Geist[1], Olivier Pietquin[1,2]

[1] Supélec, IMS-MaLIS Research group, France
{bilal.piot,matthieu.geist,olivier.pietquin}@supelec.fr
[2] GeorgiaTech-CNRS UMI 2958, France

Abstract. This paper provides a comparative study between Inverse Reinforcement Learning (IRL) and Apprenticeship Learning (AL). IRL and AL are two frameworks, using Markov Decision Processes (MDP), which are used for the imitation learning problem where an agent tries to learn from demonstrations of an expert. In the AL framework, the agent tries to learn the expert policy whereas in the IRL framework, the agent tries to learn a reward which can explain the behavior of the expert. This reward is then optimized to imitate the expert. One can wonder if it is worth estimating such a reward, or if estimating a policy is sufficient. This quite natural question has not really been addressed in the literature right now. We provide partial answers, both from a theoretical and empirical point of view.

1 Introduction

This paper provides a comparative study between two methods, using the Markov Decision Process (MDP) paradigm, that attempt to solve the imitation learning problem where an agent (called the apprentice) tries to learn from demonstrations of an expert. These two methods are Apprenticeship Learning (AL) [1] and Inverse Reinforcement Learning (IRL) [8]. In the AL framework, the agent tries to learn the expert policy or at least a policy which is as good as the expert policy (according to an unknown reward function). In the IRL framework, the agent tries to learn a reward which can explain the behavior of the expert and which is optimized to imitate it. AL can be reduced to classification [7,3,6,11] where the agent tries to mimic the expert policy via a Supervised Learning (SL) method such as classification. There exist also several AL algorithms inspired by IRL such as [1,10] but they need to solve recursively MDPs which is a difficult problem when the state space is large and the dynamics of the MDP is unknown.

The key idea behind IRL is that the reward is the most succinct representation of the task. However, as the outputs of IRL algorithms are rewards, it is still required to solve an MDP to obtain an optimal policy with respect to this reward. With AL algorithms, the output is a policy which can be directly used. However, this policy is fixed and cannot adapt to a perturbation of dynamics which could be done if one knew the true reward, as it is a representation of the

task possibly independent of the dynamics. Thus, a natural question arises: in which circumstances is it interesting to use an IRL algorithm, knowing that it still needs to solve an MDP in order to obtain a policy?

First, we analyse the difference of value functions between the apprentice and the expert policies when a classifier is used as AL method (in the infinite horizon case). When compared to the sole (as far as we know) related result in IRL, quantifying the quality of an apprentice trained with the recently introduced SCIRL (Structured Classification based IRL) algorithm [5], this analysis tells us that estimating a reward only adds errors. Then, we perform an empirical study on the generic Garnet framework [2] to see if this first partial answer is confirmed. It turns out that it actually strongly depends on the (unknown) reward optimized by the expert: roughly, the less informative the reward is, the more IRL provides gains compared to AL. Finally, we push this empirical study even further by perturbing the dynamics of the MDP, which goes beyond the studied theory. In this case, the advantage of IRL is even clearer.

2 Background and Notations

2.1 General Notations

Let $\mathcal{X} = (x_i)_{\{1 \leq i \leq N_\mathcal{X}\}}$ be a finite set and $f \in \mathcal{R}^\mathcal{X}$ a function, f is identified to a column vector and f^T is the transposition of f. The powerset of $\mathcal{X}$ is noted $\mathbb{P}(X)$. The set of probability distributions over $\mathcal{X}$ is noted $\Delta_\mathcal{X}$. Let $\mathcal{Y}$ be a finite set, $\Delta_\mathcal{X}^\mathcal{Y}$ is the set of functions from $\mathcal{Y}$ to $\Delta_\mathcal{X}$. Let $\zeta \in \Delta_\mathcal{X}^\mathcal{Y}$ and $y \in \mathcal{Y}$, $\zeta(y) \in \Delta_\mathcal{X}$, which can be seen as the conditional distribution probability knowing y, is also noted $\zeta(.|y)$ and $\forall x \in \mathcal{X}, \zeta(x|y) = [\zeta(y)](x)$. Besides, let $A \in \mathbb{P}(\mathcal{X})$, then $\chi_A \in \mathbb{R}^\mathcal{X}$ is the indicator function on the subset $A \subset \mathcal{X}$. The support of f is noted $\text{Supp}(f)$. Moreover, let $\mu \in \Delta_\mathcal{X}$, $\mathbb{E}_\mu[f]$ is the expectation of the function f with respect to the probability μ. Let $x \in \mathcal{X}$, $x \sim \mu$ means that x is sampled according to μ. Finally, we define also for $p \in \mathbb{N}^*$, the $\mathbb{L}_p$-norm of the function f: $\|f\|_p = (\sum_{x \in \mathcal{X}} (f(x)^p))^{\frac{1}{p}}$, and $\|f\|_\infty = \max_{x \in \mathcal{X}} f(x)$.

2.2 Markov Decision Process

A finite Markov Decision Process (MDP) is a tuple $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}$ where $\mathcal{S} = (s_i)_{\{1 \leq i \leq N_\mathcal{S}\}}$ is the finite state space, $\mathcal{A} = (a_i)_{\{1 \leq i \leq N_\mathcal{A}\}}$ is the finite action space, $\mathcal{P} \in \Delta_\mathcal{S}^{\mathcal{S} \times \mathcal{A}}$ is the Markovian dynamics of the MDP, $\mathcal{R} \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$ is the reward function and γ is the discount factor. A stationary and Markovian policy $\pi \in \Delta_\mathcal{A}^\mathcal{S}$ represents the behavior of an agent acting in the MDP $\mathcal{M}$. The set of all Markovian and stationary policies is noted $\Pi_{MS} = \Delta_\mathcal{A}^\mathcal{S}$. When the policy π is deterministic, it can also be seen as an element of $\mathcal{A}^\mathcal{S}$ and $\pi(s)$ is the action chosen by the policy π in state s. The quality of this behavior in the infinite horizon framework is quantified by the value function $v_\mathcal{R}^\pi \in \mathbb{R}^\mathcal{S}$ which maps to each state the expected and discounted cumulative reward for starting in this state and following the policy π afterwards: $\forall s \in \mathcal{S}, v_\mathcal{R}^\pi(s) = \mathbb{E}[\sum_{t \geq 0} \gamma^t \mathcal{R}(s_t, a_t) | s_0 = s, \pi]$.

A policy $\pi^*_{\mathcal{R}}$ (according to the reward $\mathcal{R}$) is said optimal if its value function $v^*_{\mathcal{R}}$ satisfies $v^*_{\mathcal{R}} \geq v^{\pi}_{\mathcal{R}}$ for any policy π and component wise.

Let P_π be the stochastic matrix $P_\pi = (\sum_{a \in A} \pi(a|s)\mathcal{P}(s'|s,a))_{\{(s,s') \in \mathcal{S}^2\}}$ and $\mathcal{R}_\pi \in \mathbb{R}^{\mathcal{S}}$ the function such that: $\forall s \in \mathcal{S}, \mathcal{R}_\pi(s) = \sum_{a \in A} \pi(a|s)\mathcal{R}(s,a)$. With a slight abuse of notation, we may write a the policy which associates the action a to each state s. The Bellman evaluation (resp. optimality) operators $T^{\pi}_{\mathcal{R}}$ (resp. $T^*_{\mathcal{R}}$) : $\mathcal{R}^{\mathcal{S}} \to \mathcal{R}^{\mathcal{S}}$ are defined as $T^{\pi}_{\mathcal{R}} v = \mathcal{R}_\pi + \gamma P_\pi v$ and $T^*_{\mathcal{R}} v = \max_\pi T^{\pi}_{\mathcal{R}} v$. These operators are contractions and $v^{\pi}_{\mathcal{R}}$ and $v^*_{\mathcal{R}}$ are their respective fixed-points: $v^{\pi}_{\mathcal{R}} = T^{\pi}_{\mathcal{R}} v^{\pi}_{\mathcal{R}}$ and $v^*_{\mathcal{R}} = T^*_{\mathcal{R}} v^*_{\mathcal{R}}$. The action-value function $\mathcal{Q}^{\pi}_{\mathcal{R}} \in \mathcal{S} \times \mathcal{A}$ adds a degree of freedom on the choice of the first action, it is formally defined as $\mathcal{Q}^{\pi}_{\mathcal{R}}(s,a) = [T^a_{\mathcal{R}} v^{\pi}_{\mathcal{R}}](s)$. We also write, when it exists, $\rho_\pi \in \mathbb{R}^{\mathcal{S}}$ the stationary distribution of the policy π (satisfying $\rho^T_\pi P_\pi = \rho^T_\pi$). The existence and uniqueness of ρ_π is guaranteed when the Markov chain induced by the matrix of finite size P_π is irreducible which will be supposed true in the remaining of the paper.

2.3 AL and IRL

AL and IRL are two methods that attempt to solve the imitation problem using the MDP paradigm. More precisely, in the AL framework, the apprentice, given some observations of the expert policy π_E, tries to learn a policy π_A which is as good as the expert policy according to the unknown reward $\mathcal{R}$ that the expert is trying to optimize (often the expert is considered optimal: $v^{\pi_E}_{\mathcal{R}} = v^*_{\mathcal{R}}$). This can be expressed numerically: the apprentice tries to find a policy π_A such that the quantity: $\mathbb{E}_\nu[v^{\pi_E}_{\mathcal{R}} - v^{\pi_A}_{\mathcal{R}}]$ is the lowest possible, where $\nu \in \Delta_S$. In general $\nu = \rho$ where ρ is the uniform distribution or $\nu = \rho_{\pi_E}$ (ρ_{π_E} is also noted ρ_E). In the IRL framework, the apprentice is trying to learn a reward $\hat{\mathcal{R}}$ which could explain the expert behavior. More precisely, given some observations of the expert policy π_E, the apprentice is trying to learn $\hat{\mathcal{R}}$ such that $\pi_E \approx \pi^*_{\hat{\mathcal{R}}}$. This can be expressed numerically, the apprentice is trying to learn a reward $\hat{\mathcal{R}}$ such that the quantities $\mathbb{E}_\nu[v^{\pi^*_{\hat{\mathcal{R}}}}_{\hat{\mathcal{R}}} - v^{\pi_E}_{\hat{\mathcal{R}}}]$ or $\mathbb{E}_\nu[v^{\pi_E}_{\mathcal{R}} - v^{\pi^*_{\hat{\mathcal{R}}}}_{\mathcal{R}}]$ are the lowest possible.

3 Theoretical Study

This section gives some theoretical insights into the question: Is it worth estimating a reward. First, we present a theoretical result for AL reduced to classification for the infinite horizon case. A proof of this result is given on the appendix 6. The result is an upper bound on the difference of the value functions of the expert and apprentice policies. As a previous bound for AL reduced to classification in the finite horizon case had been proposed in [11], we give an informal comparison of the two results. Besides, there is also a performance bound for an IRL algorithm [5] (SCIRL) which allows us to compare IRL and AL performances from a theoretical point of view. We choose to compare those bounds because the classification and the SCIRL algorithms does not need to resolve iteratively MDPs. Thus, there is no Approximate Dynamic programming error to deal with and to propagate to obtain the performance of the algorithm.

3.1 AL Reduced to Classification for the Infinite Horizon Case

A simple way to realize an AL method is by pure mimicry via an SL method such as classification. More precisely, we assume that some demonstrations examples $D_E = (s_i, a_i)_{\{1 \leq i \leq N\}}$ where $a_i \sim \pi_E(.|s_i)$ are available. Without loss of generality, we assume that the states s_i are sampled according to some probability distribution $\nu \in \Delta_{\mathcal{S}}$. So, the data (s_i, a_i) are sampled according to the distribution μ_E such that: $\mu_E(s,a) = \nu(s)\pi_E(a|s)$. Then, a classifier is learnt based on these examples (with discrete actions, it is a multi-class classification problem) thanks to an SL algorithm. This outputs a policy $\pi_C \in \mathcal{A}^{\mathcal{S}}$, which associates to each state an action. The quality of the classifier is quantified by the classification error: $\epsilon_C = \mathbb{E}_{\mu_E}[\chi_{\{(s,a) \in \mathcal{S} \times \mathcal{A}, \pi_C(s) \neq a\}}] = \sum_{s \in S} \sum_{a \in \mathcal{A}, a \neq \pi_C(s)} \nu(s)\pi_E(a|s)$. The quality of the expert (respectively to the unknown reward function $\mathcal{R}$) may be quantified with $v_{\mathcal{R}}^{\pi_E}$. Usually, it is assumed that the expert is optimal (that is, $v_{\mathcal{R}}^{\pi_E} = v_{\mathcal{R}}^{*}$), but it is not necessary for the following analysis (the expert may be sub-optimal respectively to $\mathcal{R}$). The quality of the policy π_C can also be quantified by its value function $v_{\mathcal{R}}^{\pi_C}$. In the following, we bound $\mathbb{E}_{\nu}[v_{\mathcal{R}}^{\pi_E} - v_{\mathcal{R}}^{\pi_C}]$ which represents the difference between the quality of the expert and the classifier policy. If this quantity is negative, that is fine, because (in mean), π_C is better than π_E. So, only an upper bound is computed. This upper-bound shows the soundness of the AL through classification method for the infinite horizon case.

Let define the following concentration coefficient: $C_{\nu} = (1-\gamma)\sum_{t \geq 0} \gamma^t c_{\nu}(t)$ where $\forall t \in \mathbb{N}, c_{\nu}(t) = \max_{s \in \mathcal{S}} \frac{(\nu^T P_{\pi_E}^t)(s)}{\nu(s)}$. Notice that if $\nu = \rho_E$, which is a quite reasonable assumption, then $C_{\nu} = C_{\rho_E} = 1$.

Theorem 1. *Let π_C be the classifier policy (trained on the data set D_E to imitate the expert policy π_E). Let also ϵ_C be the classification error and C_{ν} the above defined concentration coefficient. Then $\forall \mathcal{R} \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$:*

$$\mathbf{E}_{\nu}[v_{\mathcal{R}}^{\pi_E} - v_{\mathcal{R}}^{\pi_C}] \leq \frac{2C_{\nu}\|\mathcal{R}\|_{\infty}}{(1-\gamma)^2}\epsilon_C.$$

The proof of Th. 1 is given on the appendix 6 and is based on the propagation of the classification error. In [11], the authors have established similar bounds in the finite horizon case. However, as most of AL and IRL algorithms considered so far the infinite horizon framework, we think that our result has its interest.

3.2 The Bound on the Finite-Horizon Case

In this section, we introduce specific notations to the finite horizon case and we interpret the results from [11]. Let consider a finite MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}\}$ with horizon H and without discount factor γ. A Markovian and non-stationary policy is an element of the set Π_{MS}^H; if π is non-stationary, then π^t refers to the stationary policy that is equal to the t^{th} component of π. Similarly to the

infinite horizon case, we define the value function of the policy π at time t:

$$\forall s \in \mathcal{S}, v^{\pi}_{t,\mathcal{R}}(s) = \mathbb{E}[\sum_{t'=t}^{H} \mathcal{R}(s_{t'}, a_{t'})|s_t = s, \pi].$$

Let D^t_π be the distribution on state-action pairs at time t under policy π. In other words, a sample (s, a) is drawn from D^t_π by first drawing $s_1 \sim \nu \in \Delta_{\mathcal{S}}$, then following policy π for time steps 1 through t, which generates a trajectory $(s_1, a_1, \ldots, s_t, a_t)$, and then letting $(s, a) = (s_t, a_t)$. More formally, we have:

$$\forall 1 \leq t \leq H, \forall (s,a) \in \mathcal{S} \times \mathcal{A}, D^t_{\pi,\nu}(s,a) = (\nu^T (P_{\pi^1} \times \cdots \times P_{\pi^{t-1}}))(s)\pi^t(a|s).$$

In [11], the authors suppose the availability of the set of trajectories $D_E = (\omega_i)_{\{1 \leq i \leq N\}}$ where $\omega_i = (s^i_1, a^i_1, \ldots, s^i_H, a^i_H)$ with $s^i_1 \sim \nu \in \Delta_{\mathcal{S}}$ and $(s^i_t, a^i_t) \sim D^t_{\pi_E}$ where $1 \leq t \leq H$ and π_E is the non-stationary and Markovian expert policy. In the finite horizon case, Apprenticeship Learning through classification will consists in learning an apprentice policy $\pi_C = (\pi^t_C)_{\{1 \leq t \leq H\}}$ thanks to H classifiers trained on the sets $D^t_E = (s^i_t, a^i_t)_{\{1 \leq i \leq N\}}$. Thus, for each set $D^t_E = (s^i_t, a^i_t)_{\{1 \leq i \leq N\}}$, we train a multi-class classifier and learn a deterministic policy π^t_C with classification error:

$$\epsilon^t_C = \mathbb{E}_{D^t_E}[\chi_{\{(s,a) \in \mathcal{S} \times \mathcal{A}, \pi^t_C(s) \neq a\}}].$$

We note $\epsilon_C = \max_{1 \leq t \leq H} \epsilon^t_C$. Then we have the following theorem:

Theorem 2. *Let π_E be the expert non-stationary and Markovian expert policy, D_E a set of N trajectories with $s^i_1 \sim \nu \in \Delta_{\mathcal{S}}$ and π_C the policy learnt by the H classifiers, then:*

$$\mathbb{E}_\nu[v^{\pi_E}_{1,\mathcal{R}} - v^{\pi_C}_{1,\mathcal{R}}] \leq \min(2\sqrt{\epsilon_C}H^2, 4\epsilon_C H^3 + \delta_{\pi_E})\|\mathcal{R}\|_\infty,$$

where $\delta_{\pi_E} = \frac{\mathbb{E}_\nu[v^*_{1,\mathcal{R}} - v^{\pi_E}_{1,\mathcal{R}}]}{\|\mathcal{R}\|_\infty}$ represents the sub-optimality of the expert.

It is possible to compare these results with our bound, even if one deals with the infinite horizon case and the other with the finite horizon case, by informally noticing that the introduction of the discount factor γ in the infinite horizon corresponds to an horizon of length $\frac{1}{1-\gamma}$: $\sum_{t \geq 0} \gamma^t = \frac{1}{1-\gamma}$. By replacing H by $\frac{1}{1-\gamma}$ in the the precedent bound, we obtain:

$$\mathbb{E}_\nu[v^{\pi_E}_{1,\mathcal{R}} - v^{\pi_C}_{1,\mathcal{R}}] \leq \min(\frac{2\sqrt{\epsilon_C}}{(1-\gamma)^2}, \frac{4\epsilon_C}{(1-\gamma)^3} + \delta_{\pi_E})\|\mathcal{R}\|_\infty.$$

So, if we informally identify the classification errors and the horizon H to $\frac{1}{1-\gamma}$, our bound is slightly better either by $\sqrt{\epsilon_C}$ or by $\frac{2}{1-\gamma}$. Moreover, as our bound is specific to the infinite horizon, it is more adapted to AL and IRL algorithms as most of them consider the infinite horizon case.

3.3 SCIRL and Its Performance Bound

[5] assume that the unknown reward is linearly parameterized by some feature vector. More precisely, let $\phi(s,a) = (\phi_1(s,a), \ldots, \phi_p(s,a))^T$ be a feature vector composed of $p \in \mathbb{N}^*$ basis functions $\phi_i \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$, the parameterized reward function is $\mathcal{R}_\theta(s,a) = \theta^T\phi(s,a) = \sum_{1\leq i\leq p}\theta_i\phi_i(s,a)$. Searching a good reward thus reduces to searching a good parameter vector $\theta \in \mathbb{R}^p$. The choice of features is done by the user. Moreover, SCRIL needs the estimation of the expert feature expectation ω_{π_E} [5] which is the expected discounted cumulative feature vector for starting in a given state, applying a given action and following the expert policy:

$$\omega_{\pi_E}(s,a) = \mathbb{E}[\sum_{t\geq 0}\gamma^t\phi(s_t,a_t)|s_0 = s, a_0 = a, \pi_E].$$

It can be seen that: $\mathcal{Q}^{\pi_E}_{\mathcal{R}_\theta}(s,a) = \theta^T\omega_{\pi_E}(s,a)$. An estimation of the feature expectation $\hat{\omega}_{\pi_E}$ is done via the expert data set: D_E. The problem of estimating the expert feature is a policy evaluation problem. Then, SCIRL uses the estimation of the expert feature expectation $\hat{\omega}_{\pi_E}$ as the basis function of a linearly parameterized score-based multi-class classifier fed by the set D_E. The classification error is $\epsilon_C = \mathbb{E}_{\mu_E}[\chi_{\{(s,a)\in\mathcal{S}\times\mathcal{A},\pi_C(s)\neq a\}}]$ with $\pi_C(s) = \text{argmax}_{a\in\mathcal{A}}\,\theta_C^T\hat{\omega}_{\pi_E}(s,a)$ and θ_C the output of the score-based classifier. The reward outputted by the SCIRL algorithm is $\mathcal{R}_C = \theta_C^T\phi$. Then, the performance bound for this algorithm is:

$$0 \leq \mathbb{E}_{\rho_E}[v^*_{\mathcal{R}_C} - v^{\pi_E}_{\mathcal{R}_C}] \leq \frac{C_f}{(1-\gamma)}\left(\frac{2\|\mathcal{R}_C\|_\infty\epsilon_C}{1-\gamma} + \bar{\epsilon}_Q\right),$$

With $C_f = (1-\gamma)\sum_{t\geq 0}\gamma^t c_f(t)$ where $\forall t \in \mathbb{N}, c_f(t) = \max_{s\in\mathcal{S}}\frac{(\rho_E^T P^t_{\pi^*_{\mathcal{R}_C}})(s)}{\rho_E(s)}$. Moreover, $\bar{\epsilon}_Q = \mathbb{E}_{\rho_E}[\max_{a\in\mathcal{A}}\epsilon_Q(.,a) - \min_{a\in\mathcal{A}}\epsilon_Q(.,a)]$, where $\epsilon_Q(s,a) = \theta_C^T(\hat{\omega}_{\pi_E}(s,a) - \omega_{\pi_E}(s,a))$, is a measure of the error estimation of the feature expectation. This bound is specific to the reward $\mathcal{R}_C$ and the constant C_f is not equal to 1 when $\nu = \rho_E$, which makes this bound possibly quite worst than the pure classification bound, even when the expert feature expectation is perfectly estimated ($\bar{\epsilon}_Q = 0$). This seems to indicate that this IRL algorithm is less interesting than a simple classification algorithm in theory. However, in practice, we will see that for specific unknown rewards SCIRL can have much better performance than a classification algorithm (see Sec. 4).

4 Empirical Study

This section shows through experiments that the previous theoretical bounds does not tell everything about AL methods and IRL methods. Here, several experiments are conducted and show the interest of finding a reward thanks to a general framework of experiments called the Garnet framework. We choose a particular framework where all the problems are finite MDPs with a tabular representation. Even if those problems are not challenging, they allow comparing fairly the different approaches without the problem of bias induced by the

choice of representation. The comparison is done between a pure classification algorithm and two recently published IRL algorithms which are SCIRL and Relative Entropy IRL (RE) [4], for which there is no known error analysis. The pure classification algorithm was chosen as a benchmark for the AL approach because it has a theoretical performance guarantee and does not need to resolve iteratively MDPs unlike most of the other algorithms. SCIRL and RE were chosen as benchmarks for the IRL approach because they also do not need to resolve iteratively MDPs which reduces the impact of Approximate Dynamic Programming (ADP) in the interpretation even if the outputted reward is optimized via the policy iteration algorithm. These experiments show that the choice of the underlying unknown reward, which is used in order to create the expert policy thanks to the policy iteration algorithm, is crucial. Indeed when the unknown reward is normally distributed on each state-action-couple the classification has quite good performance whereas it has quite low performance when the reward is sparse or state-only-dependent. The intuitive idea behind those results is: when the reward is too informative, the impact of the optimization horizon is reduced, which favors the classification approach.

4.1 AL and IRL Algorithms

The first algorithm is a pure classification algorithm. More precisely, it is multi-class classification algorithm fed by the set D_E using a structured large-margin approach [12] which consists in minimizing the following criterion with respect to $\mathcal{Q} \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$:

$$\mathfrak{L}_0(\mathcal{Q}) = \frac{1}{N}\sum_{i=1}^{N} \max_{a\in\mathcal{A}}[\mathcal{Q}(s_i,a) + l(s_i,a)] - \mathcal{Q}(s_i,a_i) + \lambda\|\mathcal{Q}\|_2^2,$$

where $l(s,a) = 0$ when $\exists 1 \leq i \leq N, (s,a) = (s_i,a_i)$ and $l(s,a) = 1$ otherwise. The minimization is realized via a sub-gradient descent [9]. Then the policy obtained by the algorithm is a deterministic policy such that $\pi_C(s) \in \operatorname{argmax}_{a\in\mathcal{A}} \mathcal{Q}^*(s,a)$ where $\mathcal{Q}^*$ is the output of the minimisation of the criterion $\mathfrak{L}_0$ via the sub-gradient descent. The two other algorithms are IRL algorithms. SCIRL (presented in Sec. 3.3) needs only the set D_E to be implemented and outputs a reward $\mathcal{R}_C$. The instantiation of SCIRL, in our experiments, is the one described in the original paper. In order to obtain a policy π_C, this reward is optimized by the policy iteration algorithm with respect to the reward $\mathcal{R}_C$. The policy iteration algorithm needs the knowledge of the whole dynamics of the MDP to be implemented but allows a comparison which does not depend on the choice of an ADP algorithm (we need solving an MDP to measure the efficiency of the estimate, but not to obtain the estimate). Like SCIRL, the RE algorithm supposes a linear parametrization of the reward. The principle of the Relative Entropy method is based on minimizing the relative entropy (KL divergence) between the empirical distribution of the state-action trajectories under a random policy and the distribution of the trajectories under a policy that matches the expert feature expectation [4]. The RE algorithm used in this paper is the

one described in the original paper. It needs the set D_E and also requires a set D_P of sampled trajectories according to a non-expert policy. In the experiments, the random policy will be chosen in order to generate the set D_P (see Sec. 4.3). The output of the algorithm is a reward $\mathcal{R}_C$ and a policy iteration algorithm is also used to obtain the policy π_C relative to the outputted reward.

4.2 The Garnet Framework

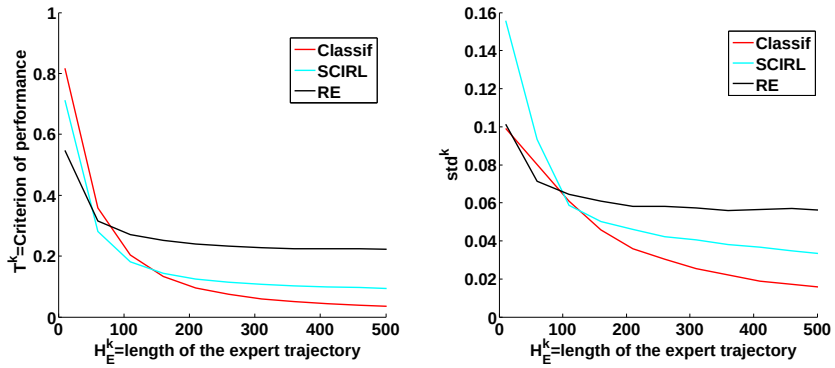
The Garnet problems are a class of randomly constructed finite MDPs meant to be totally abstract while remaining representative of the kind of finite MDPs that might be encountered in practice [2]). The routine to create an instance of a stationary Garnet problem is characterized by 3 parameters and written as $Garnet(N_S, N_A, N_B)$. The parameters N_S and N_A are the number of states and actions respectively, and N_B is a branching factor specifying the number of next states for each state action pair. The next states are chosen at random from the state set without replacement. The probability of going to each next state is generated by partitioning the unit interval at $N_B - 1$ cut points selected randomly. The reward $\mathcal{R}(s, a)$ will be chosen depending on the experiments. For each Garnet problem, it is possible to compute an expert policy π_E thanks to the reward $\mathcal{R}$ via the policy iteration algorithm. Finally, the discount factor is fixed to 0.99.

4.3 Pure Classification Versus SCRIL and RE

The idea, in order to obtain a general result, is to run the same experiment on hundreds of MDPs and regroup the results at the end. All the algorithms are fed with data sets of the the following type: $D_E = (s_i, a_i)_{\{1 \leq i \leq N\}}$ where $a_i \sim \pi_E(.|s_i)$. More particularly, $D_E = (\omega_j)_{\{1 \leq i \leq K_E\}}$ where $\omega_j = (s_{i,j}, a_{i,j})_{\{1 \leq i \leq H_E\}}$ is a trajectory obtained by starting from a random state $s_{1,j}$ (chosen uniformly) and applying the policy π_E H_E times ($s_{i+1,j} \sim P(.|s_{i,j}, a_{i,j})$). So, D_E is composed by K_E trajectories of π_E of length H_E and we have $K_E H_E = N$. We also fed the RE algorithms with a data set of sampled transitions $D_P = (s_i, a_i, s'_i)_{\{1 \leq i \leq N'\}}$ where $a_i \sim \pi_R(.|s_i)$ with π_R the random policy (uniform distribution over the actions for each state) and where $s'_i \sim P(.|s_i, a_i)$. Actually, D_P has the particular form $D_P = (\tau_j)_{\{1 \leq j \leq K_P\}}$ where $\tau_j = (s_{i,j}, a_{i,j}, s'_{i,j})_{\{1 \leq i \leq H_P\}}$ is a trajectory obtained by starting from a random state $s_{1,j}$ (chosen uniformly) and applying the policy π_R H_P times ($s'_{i,j} = s_{i+1,j} \sim P(.|s_{i,j}, a_{i,j})$). So, D_P is composed by K_P trajectories of π_R of length H_P and we have $K_P H_P = N'$. Therefore, if we have for a given Garnet problem π_E and π_R, the set of parameters (K_E, H_E, K_P, H_P) is sufficient to instantiate sets of types D_E and D_P.

Our first experiment shows the performance of the algorithms when H_E is increasing and when the reward for each Garnet is chosen normally distributed for each state-action couple. The reward $\mathcal{R}(s, a)$ is selected randomly according to a normal distribution with mean 0 and with standard deviation 1. It consists in generating 100 Garnet problems of the type $Garnet(N_S, N_A, N_B)$, where N_S is uniformly chosen between 50 and 100, N_A uniformly chosen between 5 and 10 and

N_B uniformly chosen between 2 and 5 . This gives us the set of Garnet problems $\mathfrak{G} = (G_p)_{\{1 \leq p \leq 100\}}$. On each problem p of the set $\mathfrak{G}$, we compute π_E^p and π_R^p. The parameter H_E takes its values in the set $(H_E^k)_{\{1 \leq k \leq 11\}} = (50, 100, 150, .., 500)$, $K_E = 1$, $H_P = 10$, $K_P = 50$. Then, for each set of parameters (K_E, H_E^k, K_P, H_P) and each G_p, we compute 100 expert policy sets $(D_E^{i,p,k})_{\{1 \leq i \leq 100\}}$ and 100 random policy sets $(D_P^{i,p,k})_{\{1 \leq i \leq 100\}}$. Our criteria of performance for each couple $(D_E^{i,p,k}, D_P^{i,p,k})$ is the following: $T^{i,p,k} = \frac{\mathbb{E}_\rho[v_{\mathcal{R}}^{\pi_E^p} - v_{\mathcal{R}}^{\pi_C^{i,p,k}}]}{\mathbb{E}_\rho[v_{\mathcal{R}}^{\pi_E^p}]}$, where π_E^p is the expert policy, $\pi_C^{i,p,k}$ is the policy induced by the algorithm fed by the couple $(D_E^{i,p,k}, D_P^{i,p,k})$ and ρ is the uniform distribution over the state space S. For the pure classifier, we have $\pi_C^{i,p,k}(s) \in \text{argmax}_{a \in A} \hat{Q}^*(s,a)$ where $\hat{Q}^*$ is the minimizer of $\mathfrak{L}_0$. For the SCIRL and RE algorithms, $\pi_C^{i,p,k}$ is the policy obtained by optimizing the reward $\mathcal{R}_C$ outputted by the algorithm via the policy iteration algorithm. Our mean criterion of performance for each set of parameters (K_E, H_E^k, K_P, H_P) is: $T^k = \frac{1}{10000} \sum_{1 \leq p \leq 100, 1 \leq i \leq 100} T^{i,p,k}$. For each algorithm we plot $(H_E^k, T^k)_{\{1 \leq k \leq 15\}}$. Another criterion is also useful in order to interpret the results. For each Garnet problem and each set of parameters, we calculate the standard deviation $\text{std}^{p,k}$ for each algorithm: $\text{std}^{p,k} = \left\{ \frac{1}{100} \sum_{1 \leq i \leq 100} [T^{i,p,k} - \sum_{1 \leq j \leq 100} T^{j,p,k}]^2 \right\}^{\frac{1}{2}}$. Then we compute the mean standard deviation over the 100 Garnet problems for each set of parameters: $\text{std}^k = \frac{1}{100} \sum_{1 \leq p \leq 100} \text{std}^{p,k}$. For each algorithm we plot $(H_E^k, \text{std}^k)_{\{1 \leq k \leq 15\}}$. Results are reported on Fig. 1. Here, we see that the pure classification algorithm



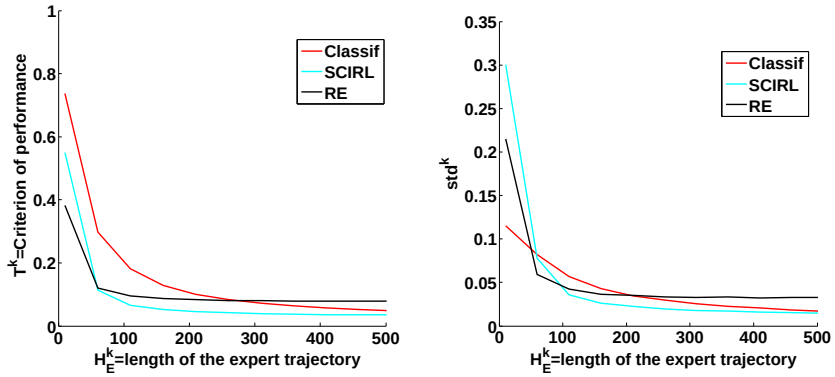
(a) Performance (b) Standard deviation

Fig. 1. Garnets experiment: normally distributed reward.

has a better performance over the IRL algorithms when the number of data is increasing. This can be explained by the particular shape of the reward which is particularly suited to make the pure classification algorithm work well and IRL

algorithms work bad. Indeed, as there are rewards for each state-action couples which are normally chosen, doing a misclassification is not so important as there will be rewards with the same form in the next states. However, as there are a lot of rewards everywhere, a lot of data is needed for an IRL algorithm to be able to estimate a meaningful reward. Another possible but complementary interpretation of those results is: as the reward is very informative, the choice of the action does not depend too much on the future states and the impact of the optimization horizon is strongly reduced.

The second experiment is exactly the same as the first one, except that the reward is no longer normally distributed. For each Garnet, we generate a reward with a small support: $\text{Supp}(\mathcal{R}) \leq \frac{N_S N_A}{50}$ by randomly choosing between 1 and $\frac{N_S N_A}{50}$ couples (s, a) such that $\mathcal{R}(s, a) \neq 0$ (reward randomly chosen between 0 and 1). For the other couples (s, a), $\mathcal{R}(s, a) = 0$. Results are reported on Fig. 2. Here, we see that the IRL algorithms work better than previously and the pure



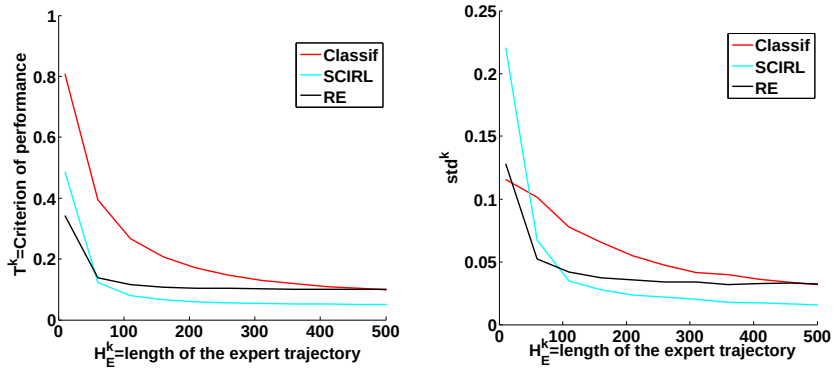
(a) Performance (b) Standard deviation

Fig. 2. Garnets experiment: sparse reward.

classification algorithms has its performance deteriorated a little bit compared to the previous experiment. This can be explained by the shape of the unknown reward. As the unknown reward is sparse, doing a misclassification on a state where the expert choose the action that gives a reward is important as there are only few state-action couples with rewards. Thus, the pure classification algorithm may have some problems with few data which is what we observe on Fig. 2(a). Moreover, the IRL algorithms have a better performance, maybe because the unknown reward has a simpler structure to learn. Again as the reward is less informative, the impact of the optimization horizon may be more important than for the previous reward which deteriorates the performance of the classification.

The third experiment is exactly the same as the first one, except that the reward is state-only-dependent. To construct a state-only-dependent reward, it is

sufficient for each $s \in \mathcal{S}$ to select randomly a value $\mathcal{R}(s)$ according to a normal distribution with mean 0 and with standard deviation 1 and then $\forall(s,a) \in \mathcal{S} \times \mathcal{A} = \mathcal{R}(s,a) = \mathcal{R}(s)$. Results are reported on Fig 3. Here, the performance



(a) Performance (b) Standard deviation

Fig. 3. Garnets experiment: state-only-dependent reward.

of the IRL algorithms is better than the second experiment and than the pure classification. This can be explained by the fact that the structure of the reward is even simpler. The pure classification see its performance deteriorated compared to the second experiment. As the unknown reward depends only on the state and not on the action, it is very important to follow the path of the expert to obtain a good performance. Thus, a misclassification on a given state which leads to a bad path can be very damageable and lead to bad performance.

5 Dynamics Perturbations

In this section, we want to show that it can be interesting to retrieve the reward in order to be more stable to dynamics perturbations. As the reward is seen as the most succinct hypothesis explaining the behavior of the expert, we can expect that the reward outputted by the IRL algorithms is such that its optimization will lead to a near-optimal behavior even if there is a dynamics perturbation. The dynamics perturbations considered are the ones which keep identical the structure of the MDP. The structure of the MDP is for a given state-action couple (s,a) the different states that could be reached by choosing the action a in state s, that is $\text{Supp}(P_{s,a})$. The structure of the MDP is the set $(\text{Supp}(P_{s,a}))_{\{(s,a)\in\mathcal{S}\times\mathcal{A}\}}$. Therefore a dynamic perturbation is the choice of a dynamics $\tilde{\mathcal{P}}$ different from $\mathcal{P}$ such that: $(\text{Supp}(P_{s,a}))_{\{(s,a)\in\mathcal{S}\times\mathcal{A}\}} = (\text{Supp}(\tilde{P}_{s,a}))_{\{(s,a)\in\mathcal{S}\times\mathcal{A}\}}$.

The first experiment consists in in generating 100 Garnet problems of the type $Garnet(N_S, N_A, N_B)$, where N_S is chosen randomly between 50 and 100,

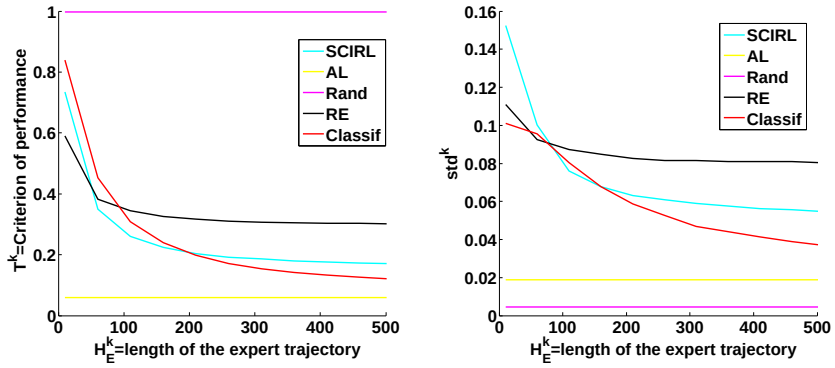
N_A randomly chosen between 5 and 10 and N_B chosen randomly between 2 and 5 . This gives us the set of Garnet problems $\mathfrak{G} = (G_p)_{\{1 \leq p \leq 100\}}$. Here, The reward $\mathcal{R}(s,a)$ is selected randomly according to a normal distribution with mean 0 and with standard deviation 1. Then for each G_p, we realize 50 dynamics perturbation and we obtain the set of Garnets problems $\tilde{\mathfrak{G}} = (G_{p,q})_{\{1 \leq p \leq 100, 1 \leq q \leq 50\}}$. On each problem p,q of the set $\tilde{\mathfrak{G}}$, we compute $\pi_E^{p,q}$ and $\pi_R^{p,q}$ and on each problem p of the set $\mathfrak{G}$, we compute π_E^p and π_R^p. The parameter H_E takes its values in the set $(H_E^k)_{\{1 \leq k \leq 15\}} = (50, 100, 150, .., 500)$, $K_E = 1$, $H_P = 10$, $K_P = 50$. Then, for each set of parameters (K_E, H_E^k, K_P, H_P) and each G_p, we compute 100 expert policy sets $(D_E^{i,p,k})_{\{1 \leq i \leq 100\}}$ and 100 random policy sets $(D_P^{i,p,k})_{\{1 \leq i \leq 100\}}$. Our criteria of performance for each couple $(D_E^{i,p,k}, D_P^{i,p,k})$ on each $G_{p,q}$ problem is the following: $T^{i,p,q,k} = \frac{\mathbb{E}_\rho[v_{\mathcal{R}}^{\pi_E^{p,q}} - v_{\mathcal{R}}^{\pi_C^{i,p,k}}]}{\mathbb{E}_\rho[v_{\mathcal{R}}^{\pi_E^{p,q}}]}$, where $\pi_E^{p,q}$ is the expert policy on the problem $G_{p,q}$, $\pi_C^{i,p,k}$ is the policy induced by the algorithm fed by the couple $(D_E^{i,p,k}, D_P^{i,p,k})$ and ρ is the uniform distribution over the state space S. For the pure classifier, we have $\pi_C^{i,p,k}(s) \in \text{argmax}_{a \in A} \hat{Q}^*(s,a)$ where $\hat{Q}^*$ is the output. For the SCIRL and RE algorithms, $\pi_C^{i,p,k}$ is the policy obtained by optimizing the reward $\mathcal{R}$ outputted by the algorithm via the policy iteration algorithm. Moreover, when $\pi_C^{i,p,k} = \pi_E^p$, then $T^{i,p,q,k}$ represents the best performance possible to achieve by an AL algorithm: this curve will be noted AL in our figures. Finally, when $\pi_C^{i,p,k} = \pi_R^p$, then $T^{i,p,q,k}$ represents the performance of the random policy and this curve will be noted Rand in our figures.

Our mean criterion of performance for each set of parameters (K_E, H_E^k, K_P, H_P) is: $T^k = \frac{1}{500000} \sum_{1 \leq p \leq 100, 1 \leq q \leq 50, 1 \leq i \leq 100} T^{i,p,q,k}$. For each algorithm we plot $(H_E^k, T^k)_{\{1 \leq k \leq 15\}}$. Another criterion is also useful in order to interpret the results. For each Garnet problem G_p and each set of parameters, we calculate the standard deviation $\text{std}^{p,k}$ for each algorithm:

$$\underset{\text{std}}{p,k} = \left\{ \frac{1}{5000} \sum_{1 \leq i \leq 100}^{1 \leq q \leq 50} [T^{i,p,q,k} - \sum_{1 \leq j \leq 100}^{1 \leq q' \leq 50} T^{j,p,q',k}]^2 \right\}^{\frac{1}{2}}.$$

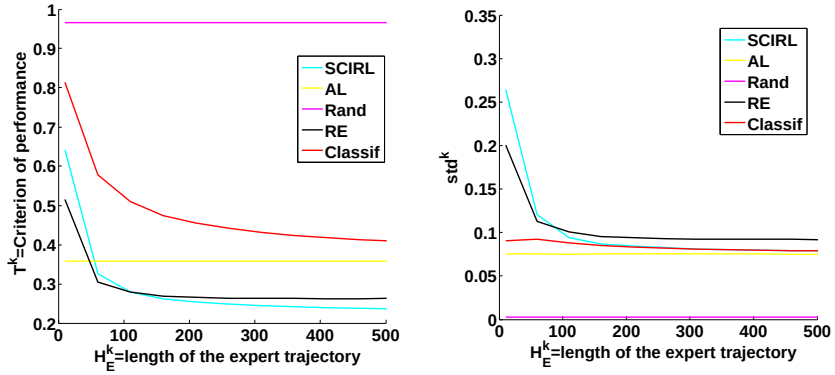
Then we compute the mean standard deviation over the 100 Garnet problems for each set of parameters: $\text{std}^k = \frac{1}{100} \sum_{1 \leq p \leq 100} \text{std}^{p,k}$. For each algorithm we plot $(H_E^k, \text{std}^k)_{\{1 \leq k \leq 15\}}$. Results are reported on Fig. 4. Here, the reward is normally distributed so a dynamic perturbation may not deteriorate too much the expert policy. Indeed, as the reward is very informative, the impact of the optimization horizon must be very small and the perturbation of dynamics will not change too much the optimal policy. We can observe this on Fig. 4(a), where we see that the yellow curve noted AL is not so far away from 0. With this shape of reward, it is better to use a pure classification algorithm to have this stability property.

The second experiment is exactly the same as the previous one, except that the reward is sparse. Results are reported on Fig. 5. As the reward is sparse, we can expect that a dynamic perturbation leads to an important deterioration



(a) Performance (b) Standard deviation

Fig. 4. Perturbed dynamics: normally distributed reward.

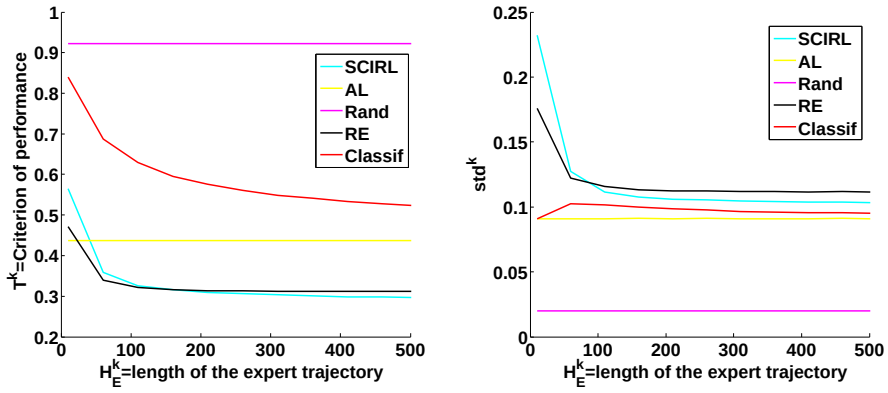


(a) Performance (b) Standard deviation

Fig. 5. Perturbed dynamics: sparse reward.

of the performance of the expert policy. Here, we see that IRL algorithms are under the yellow curve when the number of data is increasing, which means that no AL algorithms will be able to reach that level of stability. Thus, it seems that estimating a reward function in that case can be very useful because it guarantees a level of stability that no AL algorithms is able to provide.

The third experiment is exactly the same as the previous one, except that the reward is state-only-dependent. Results are reported on Fig. 6. Here the shape of reward is even simpler that the previous experiment. It seems that the IRL algorithms are even more stable with less data. Again, as the impact of the optimization horizon becomes important, the performance of the pure classification and the one of the best possible AL algorithm are really deteriorated.



(a) Performance (b) Standard deviation

Fig. 6. Perturbed dynamics: state-only-dependent reward.

6 Conclusion and Perspectives

In this paper, we tried to give some theoretical and empirical insights into the following question: is it worth estimating a reward function? First, we upper-bounded the difference between the value function of the expert and the value function of the apprentice policy, for AL reduced to classification in the infinite horizon case. This result gives a better bound than the theoretical performance bound of the SCIRL algorithm and is informally better than the bound in the finite horizon case proved in [11]. Thus, in theory, there are no specific reason to use an IRL algorithm which still needs to solve an MDP in order to obtain an optimal policy according to the reward found by the algorithm.

However, in practice, the experiments conducted in this paper on a generic task (Garnet problems) show that for specific shapes of the unknown reward function, IRL algorithms have better performance than the pure classification algorithms and possess a stability property that no AL algorithm will be able to achieve. Besides, it seems that the reward functions that favor the IRL algorithms are the less informative ones. We think that the less informative the reward is, the bigger the impact of the optimization horizon is. This is an obvious disadvantage for the pure classification method which doest not take into account this optimization horizon.

However, there is no theoretical proof explaining why IRL algorithms work better with specific forms of reward functions. This can be an interesting perspective to give more soundness to the experiments leaded in this paper. Moreover, it would be interesting to create an algorithm able to use data coming from different perturbed dynamics of the same MDP in order to learn a reward function which will be even less sensible to perturbed dynamics. This can be useful with applications where human are involved: in those kind of real-life applications, each human can be seen as a perturbed version of an MDP.

Acknowledgements. The research leading to these results has received partial funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n°270780.

References

1. Abbeel, P., Ng, A.: Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the 21st International Conference on Machine Learning (ICML) (2004)
2. Archibald, T., McKinnon, K., Thomas, L.: On the generation of markov decision processes. Journal of the Operational Research Society (1995)
3. Atkeson, C., Schaal, S.: Robot learning from demonstration. In: Proceedings of the 14th International Conference on Machine Learning (ICML) (1997)
4. Boularias, A., Kober, J., Peters, J.: Relative entropy inverse reinforcement learning. In: JMLR Workshop and Conference Proceedings Volume 15: AISTATS 2011 (2011)
5. Klein, E., Geist, M., Piot, B., Pietquin, O.: Inverse reinforcement learning through structured classification. In: Advances in Neural Information Processing Systems 25 (NIPS) (2012)
6. Langford, J., Zadrozny, B.: Relating reinforcement learning performance to classification performance. In: Proceedings of the 22nd International Conference on Machine Learning (ICML) (2005)
7. Pomerleau, D.: Alvinn: An autonomous land vehicle in a neural network. Tech. rep., DTIC Document (1989)
8. Russell, S.: Learning agents for uncertain environments. In: Proceedings of the 11th annual conference on Computational Learning Theory (COLT) (1998)
9. Shor, N., Kiwiel, K., Ruszcaynski, A.: Minimization methods for non-differentiable functions. Springer-Verlag (1985)
10. Syed, U., Schapire, R.: A game-theoretic approach to apprenticeship learning. In: Advances in Neural Information Processing Systems 21 (NIPS) (2008)
11. Syed, U., Schapire, R.: A reduction from apprenticeship learning to classification. In: Advances in Neural Information Processing Systems 23 (NIPS) (2010)
12. Taskar, B., Chatalbashev, V., Koller, D., Guestrin, C.: Learning structured prediction models: A large margin approach. In: Proceedings of the 22nd International Conference on Machine Learning (ICML) (2005)

Appendix: Proof of Th.1

We have:

$$\begin{aligned} v_{\mathcal{R}}^{\pi_E} - v_{\mathcal{R}}^{\pi_C} &\overset{(a)}{=} T_{\mathcal{R}}^{\pi_E} v_{\mathcal{R}}^{\pi_E} - T_{\mathcal{R}}^{\pi_E} v_{\mathcal{R}}^{\pi_C} + T_{\mathcal{R}}^{\pi_E} v_{\mathcal{R}}^{\pi_C} - v_{\mathcal{R}}^{\pi_C} \\ &\overset{(b)}{=} \gamma P_{\pi_E}(v_{\mathcal{R}}^{\pi_E} - v_{\mathcal{R}}^{\pi_C}) + T_{\mathcal{R}}^{\pi_E} v_{\mathcal{R}}^{\pi_C} - v_{\mathcal{R}}^{\pi_C}, \\ &\overset{(c)}{=} (I - \gamma P_{\pi_E})^{-1}(T_{\mathcal{R}}^{\pi_E} v_{\mathcal{R}}^{\pi_C} - v_{\mathcal{R}}^{\pi_C}), \end{aligned}$$

Equality (a) holds because $T_{\mathcal{R}}^{\pi_E} v^{\pi_E} = v_{\mathcal{R}}^{\pi_E}$, Equality (b) is obtained by definition of $T_{\mathcal{R}}^{\pi_E}$ and Equality (c) is true by invertibility of $I - \gamma P_{\pi_E}$ where $I \in \mathbb{R}^{\mathcal{S}\times\mathcal{S}}$ is the identity matrix. The next step is to work on the term $T_{\mathcal{R}}^{\pi_E} v_{\mathcal{R}}^{\pi_C} - v_{\mathcal{R}}^{\pi_C}$. For any function $v \in \mathbb{R}^{\mathcal{S}}$, by definition of $T_{\mathcal{R}}^{\pi_E}$: $T_{\mathcal{R}}^{\pi_E} v = \mathcal{R}_{\pi_E} + \gamma P_{\pi_E} v$. Noticing that:

$$T_{\mathcal{R}}^{\pi_E} v_{\mathcal{R}}^{\pi_C}(s) - v_{\mathcal{R}}^{\pi_C}(s) = \sum_{s'\in\mathcal{S}} \sum_{a\in\mathcal{A}} \pi_E(s,a)[\mathcal{R}(s,a) + \gamma \mathcal{P}(s'|s,a) v_{\mathcal{R}}^{\pi_C}(s')] - v_{\mathcal{R}}^{\pi_C}(s),$$

and by definition of $\mathcal{Q}_{\mathcal{R}}^{\pi_C}(s,a)$, we have:

$$\begin{aligned} \forall s \in S, T_{\mathcal{R}}^{\pi_E} v_{\mathcal{R}}^{\pi_C}(s) - v_{\mathcal{R}}^{\pi_C}(s) &= \sum_{a\in\mathcal{A}} \pi_E(s,a) \mathcal{Q}_{\mathcal{R}}^{\pi_C}(s,a) - v_{\mathcal{R}}^{\pi_C}(s), \\ &= \sum_{a\in\mathcal{A}, a\neq\pi_C(s)} \pi_E(a|s)[\mathcal{Q}_{\mathcal{R}}^{\pi_C}(s,a) - v_{\mathcal{R}}^{\pi_C}(s)]. \end{aligned}$$

So:

$$\begin{aligned} \nu^T(v_{\mathcal{R}}^{\pi_E} - v_{\mathcal{R}}^{\pi_C}) &= \nu^T(I - \gamma P_{\pi_E})^{-1}[T_{\mathcal{R}}^{\pi_E} v_{\mathcal{R}}^{\pi_C} - v_{\mathcal{R}}^{\pi_C}], \\ &= \sum_{s\in S} \sum_{t\geq 0} \gamma^t \frac{(\nu^T P_{\pi_E}^t)(s)}{\nu(s)} \nu(s)[T_{\mathcal{R}}^{\pi_E} v_{\mathcal{R}}^{\pi_C}(s) - v_{\mathcal{R}}^{\pi_C}(s)], \\ &= \sum_{s\in S} \sum_{t\geq 0} \gamma^t \frac{(\nu^T P_{\pi_E}^t)(s)}{\nu(s)} \nu(s) \sum_{a\neq\pi_C(s)} \pi_E(a|s)[\mathcal{Q}_{\mathcal{R}}^{\pi_C}(s,a) - v_{\mathcal{R}}^{\pi_C}(s)]. \end{aligned}$$

Thus by definition of C_ν:

$$\begin{aligned} \nu^T(v_{\mathcal{R}}^{\pi_E} - v_{\mathcal{R}}^{\pi_C}) &\leq \frac{C_\nu}{1-\gamma} \sum_{s\in S} \sum_{a\in\mathcal{A}, a\neq\pi_C(s)} \nu(s)\pi_E(a|s)|\mathcal{Q}_{\mathcal{R}}^{\pi_C}(s,a) - v_{\mathcal{R}}^{\pi_C}(s)|, \\ &\overset{(d)}{\leq} \frac{C_\nu}{1-\gamma} \frac{2\|\mathcal{R}\|_\infty}{1-\gamma} \sum_{s\in S} \sum_{a\in\mathcal{A}, a\neq\pi_C(s)} \nu(s)\pi_E(a|s), \\ &\overset{(e)}{=} \frac{2\|\mathcal{R}\|_\infty C_\nu \epsilon_C}{(1-\gamma)^2}. \end{aligned}$$

Inequality (d) is true because $|\mathcal{Q}_{\mathcal{R}}^{\pi_C}(s,a) - v_{\mathcal{R}}^{\pi_C}(s)| \leq \frac{2\|\mathcal{R}\|_\infty}{1-\gamma}$ and Equality (e) is true by definition of ϵ_C. This ends the proof.