Structured Classification for Inverse Reinforcement Learning

Édouard Klein[†‡], Bilal Piot[†*], Matthieu Geist[†], Olivier Pietquin[†*]
FIRSTNAME.LASTNAME@SUPELEC.FR
† *Équipe IMS/MaLIS (Supélec),* ‡ *Équipe ABC UMR 7503 (Loria-CNRS),* * *UMI 2958 (GeorgiaTech-CNRS).*

Abstract

This paper addresses the Inverse Reinforcement Learning (IRL) problem which is a particular case of learning from demonstrations. The IRL framework assumes that an expert, demonstrating a task, is acting optimally with respect to an unknown reward function to be discovered. Unlike most of existing IRL algorithms, the proposed approach doesn't require any of the following: complete trajectories from the expert, a generative model of the environment, the knowledge of the transition probabilities, the ability to repeatedly solve the forward Reinforcement Learning (RL) problem, the expert's policy anywhere in the state space. Using a classification approach in which the structure of the underlying Markov Decision Process (MDP) is implicitly injected, we end-up with an efficient subgradient descent-based algorithm. In addition, only a small amount of expert demonstrations (not even in the form of trajectories but simple transitions) is required.

Keywords: inverse reinforcement learning, structured multi-class classification

1. Introduction

This contribution addresses the problem of learning by watching or learning from demonstrations. In this context, an artificial agent learns to perform a task by observing an expert demonstrating the task. In order to find a solution to this problem, the Inverse Reinforcement Learning (IRL) paradigm has been proposed Russell (1998). The fundamental idea of IRL is that the expert acts so as to be rewarded by an unknown reward function. The aim of the IRL agent is then to infer the unknown reward function from the demonstrations given by the expert, the reward being considered as the most compact representation of the task to be transferred to the agent. This way, an optimal behavior can be obtained in any situation by optimizing the action selection process according to the reward function. Unlike supervised learning, the agent may not purely imitate the expert's actions but will try to accomplish the same task optimally. The literature on the subject is quite recent and finds its origin in Russell (1998). Different approaches have been explored since the first articles on the matter Ng and Russell (2000); Abbeel and Ng (2004).

In this literature (see Neu and Szepesvári (2009) for a survey), several bottlenecks are identified but often ignored when developing algorithms. Specifically, the dynamic of the environment where the expert and the agent are evolving is assumed to be known, or an environment model (via a simulator) must be available so as to test the effects of a policy. Moreover, most of existing approaches require solving the direct reinforcement learning problem (find an optimal policy knowing a given reward) several times. Finally, in many cases, the output of the algorithm is a policy but not a unique reward function which makes task transfer hard to realize. These constraints can lead to the development of algorithms unapplicable to real problems. In this paper, we propose a resolution method of the inverse reinforcement problem that is free from any other knowledge than the demonstrations given by the expert. The proposed approach, the Structured Classification for IRL (SCIRL), relies on a classification paradigm in which the structure of the environment's dynamics is implicitly introduced. Sec. 2 and 3 provide the necessary background and review a part of the state of the art. Sec. 4 introduces our contribution and Sec. 5 analyses the asymptotic algorithm. Section 6 shows results obtained on two classical benchmarks: inverted pendulum and the highway driving problems.

2. Background

Let's start with some background on RL and IRL and settle the notations.

2.1. (Inverse) Reinforcement Learning

The problem is seen as a sequential decision making problem posed in the Markov Decision Process (MDP) formalism. In this formalism, a system to be controlled can reach different configurations completely described at each discrete time t by a state $s_t \in S$. Given the knowledge of the state, the agent must take a decision $a_t \in A$. Then, the system evolves to a next state $s_{t+1} \in S$ with the Markovian probability $p(s_{t+1}|s_t, a_t)$. A policy, which is a function $\pi : S \to A$, defines the behavior of the agent. In this paper all the policies considered are deterministic, S and A will be finite sets and $\{P_{sa}\}_{s\in S, a\in A}$ will be a set of probability distributions, from S to $[0, 1]$ such that $P_{sa}(s') = p(s'|s, a)$, called transition probabilities. The RL paradigm aims at solving the problem of finding the optimal sequence of actions in an MDP. The optimality criterion is based on the definition of a reward function $R : S \to \mathbb{R}$ which defines the immediate reward $R(s)$ given to the agent when the system is in the state $s \in S$. The agent should not learn to maximize the immediate reward but a criterion taking into account future rewards. The criterion commonly chosen is the value function V^π:

$$\begin{aligned} V^\pi(s) &= E_s^\pi[\sum_{t\geq 0} \gamma^t R(s_t)] \\ &= R(s) + \gamma \sum_{s'\in S} P_{s\pi(s)}(s') V^\pi(s'), \end{aligned}$$

where $\gamma \in [0, 1)$ is a discount factor, and E_s^π is the expectation over the distribution of the state sequences $(s_0, s_1, \dots)$ the agent passes through when executing the policy π starting from $s_0 = s$. One can define the action-value function Q^π which adds a degree of freedom in the choice of the first action:

$$\begin{aligned} Q^\pi(s, a) &= E_{sa}^\pi[\sum_{t\geq 0} \gamma^t R(s_t)] \qquad (1) \\ &= R(s) + \gamma \sum_{s'\in S} P_{sa}(s') V^\pi(s'). \end{aligned}$$

An optimal policy π^* is defined as a policy with maximal value function, $V^{\pi*} = V^* = \sup_\pi V^\pi$, which means that $\forall \pi, \forall s, V^{\pi*}(s) \geq V^\pi(s)$. An optimal policy π^* can be computed easily thanks to the optimal action-value function $Q^{\pi*} = Q^* = \sup_\pi Q^\pi$ via a greedy mechanism:

$$\pi^*(s) \in \arg\max_{a \in A} Q^*(s,a). \tag{2}$$

The tuple $M = \{S, A, \{P_{sa}\}_{s\in S, a \in A}, \gamma, R\}$ is called a finite MDP. Resolving the RL problem with respect to the criterion V^π means to find π^*, usually without the full knowledge of the MDP but only by observing transitions and rewards. In a finite MDP the existence of an optimal deterministic policy is guaranteed when the reward function R is bounded. The optimal behaviour being defined relatively to the reward, its observation is mandatory to learn. Sometimes, specifying manually the reward function can be a hard job while it is natural to an operator to achieve the desired task. For instance, driving a car is such a task. One can accomplish this task very easily in his daily-life but it is very tricky to specify the precise weights given to each important driving-parameters such as the distance to keep with the leading car, the speed below which it is unreasonable to drive and so on. In such cases, inferring the reward function from observed behavior would be helpful. It is the aim of IRL. The expert, demonstrating the task, is thus considered as an optimal agent in a MDP and the IRL agent has access to the expert optimal policy π_E or to some trajectories drawn from this policy. The problem of finding a reward function from expert demonstrations in the IRL framework is ill-posed. Indeed, there is not uniqueness of the reward function for which the expert policy is optimal Ng et al. (1999). Particularly, each policy is optimal with respect to the uniformly-zero reward function. Some solutions have to be favored.

2.2. Feature expectation

The actual expert reward function R_E is unknown. The problem is thus to find a reward function R_θ for which the policy of the expert is also optimal. Let's assume that, given a set of p features $\psi : S \to \mathbb{R}^p$, each components of ψ: $\{\psi_i\}_{1 \leq i \leq p}$ being a basis function from S to $\mathbb{R}$, such a reward can be written:

$$R_\theta(s) = \theta^T \psi(s) = \sum_{i=1}^{p} \theta_i \psi_i(s).$$

Using this expression allows rewriting (Eq. (1)) as:

$$Q^\pi(s,a) = \theta^T \mu^\pi(s,a), \;\; \mu^\pi(s,a) = E^\pi_{sa}[\sum_{t\geq 0} \gamma^t \psi(s_t)].$$

The term μ^π is called the feature expectation of the policy π. We can observe that the chosen parametrization of the reward function R_θ and the dynamics resulting from the application of the policy π in the MDP gives rise to an induced parametrization for Q^π. To shorten notations, we may use μ^π_{sa} as $\mu^\pi(s,a)$.

We see also note (as made explicit in Abbeel and Ng (2004)) that if two policies share the same feature expectation, then they have the same value function with respect to any reward function $R_\theta = \theta^T \psi$, $\theta \in \mathbb{R}^p$. Indeed:

$$\mu^{\pi_1} = \mu^{\pi_2} \Rightarrow \theta^T \mu^{\pi_1} = \theta^T \mu^{\pi_2} \Rightarrow V^{\pi_1} = V^{\pi_2}.$$

That is why most of the algorithms in the literature have, as we will see, the objective of minimizing a *distance* between the feature expectation of the expert and the feature expectation of the apprentice.

3. State of the Art

Let π_θ denote an optimal policy according to a reward function $R_\theta(s) = \theta^T \psi(s)$, for some $\theta \in \mathbb{R}^p$. With a slight abuse of notation, let us write $\mu^\pi_{d_0}$ the mean feature expectation for a distribution d_0 over starting states: $\mu^\pi_{d_0} = E[\mu^\pi(s, \pi(s))|s \sim d_0]$. The classical approach to IRL (or less generally to apprenticeship learning, as the output of the algorithm may be a policy and not a reward function), initiated in Abbeel and Ng (2004), consists in finding a reward (a parameter vector) θ^* minimizing some loss function between the feature expectation of the expert and the one of the related optimal policy:

$$\theta^* = \operatorname*{argmin}_\theta J(\theta) \text{ with } J(\theta) = L(\mu^{\pi_\theta}_{d_0}, \mu^E_{d_0}).$$

Depending on the adopted point of view, game-theoretic-base Syed and Schapire (2008), linear programming-based Syed et al. (2008), maximum entropy-based Ziebart et al. (2008), *etc.*, the loss function and the associated update rule differ. Most of these approches and others are nicely reviewed in Neu and Szepesvári (2009).

In all cases, the solution θ^* is computed in an iterative way. At each step, one has to compute the feature expectation $\mu^{\pi_\theta}_{d_0}$ for some parameter vector θ. This implies computing π_θ, the optimal policy according to the reward $\theta^T \psi(s)$. Therefore, solving the IRL problem with one of these methods involves solving repeatedly the direct RL problem (some of these algorithms requiring also the model to be known). This is a drawback we wish to get rid of.

A notable exception is Boularias et al. (2011). This work, based on a relative entropy argument, finds θ^* by performing a subgradient ascent on some utility function. Estimating the subgradient would require to sample trajectories according to π_θ, with θ being the current estimate, thus would require to solve the direct problem. However, this can be avoided in this specific framework by the use of importance sampling. The gradient is estimated from trajectories sampled from some random policy. Still, this may be a problem when the only available data comes from the expert.

Another approach consists in casting imitation learning into a multi-class classification problem, introducing in some way the structure of the MDP. For example, in Melo and Lopes (2010), the agent policy is learnt as the solution of a classification problem. They consider a kernel function built on some induced MDP metric, which may be combersome to compute. Closest to our proposed contribution is the Maximum Margin Planning (MMP) algorithm Ratliff et al. (2006), which is a classification-based algorithm associating optimal policies to MDPs. More precisely, the training set is made of N couples $(\mathcal{M}_i, \pi^*_i)$, with $\mathcal{M}_i$ an MDP (without reward) and π^*_i the associated optimal policy. Let $\mu^\pi_{d_0,\mathcal{M}_i}$ be the feature expectation for the policy π, the initial distribution d_0 and the dynamics induced by the related MDP. Let also $\mathcal{L}_{\mathcal{M}_i,\pi}$ be a positive and user-defined margin function penalizing non-optimal policies. MMP minimizes the following cost function:

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} \max_{\pi}(\theta^T \mu^{\pi}_{d_0,\mathcal{M}_i} + \mathcal{L}_{\mathcal{M}_i,\pi}) - \theta^T \mu^{\pi_i^*}_{d_0,\mathcal{M}_i}.$$

This implies solving a set of MDPs at each iteration. At the other extreme, a similar cost function is considered in Ratliff et al. (2007), which searches for a score function q associating expert actions to states. The training set being N couples $(s_i, a_i = \pi^E(s_i))$ and $l(s,a)$ being a positive user-defined margin function penalizing non-optimal actions, the cost function is

$$J(q) = \frac{1}{N}\sum_{i=1}^{N} \max_{a}(q(s_i,a) + l(s_i,a)) - q(s_i,a_i).$$

The drawback of this approach is that it does not take into account the structure of the underlying MDP, which may easily be shown experimentally to be a problem (*e.g.*, see Melo and Lopes (2010)).

The proposed contribution is a trade-off between the two previous approaches. It works at the state level rather than the MDP level, considering a similar cost function. However, it takes into account the structure of the MDP. As shown in the next section, this allows avoiding to solve the direct RL problem as intermediate steps and it requires only data from the expert.

4. Structured classification for IRL

This section introduces our contribution, the SCIRL (Structured Classification for IRL), and discusses one of its key components, the estimation of the expert's feature expectation.

4.1. Principle

First, let us review more deeply the approach of Ratliff et al. (2007), to which we will actually add some structure. Assume that we have a set of transitions $\{(s_i, a_i, s_{i+1})_{1\leq i\leq N}\}$ coming from one or several trajectories of the expert. A classic approach for generalizing the expert policy (from a multi-class classification-based point of view) would consist in finding some score function $q : S \to A$ such that for all $a \neq a_i$, $q(s_i,a_i) > q(s_i,a)$. In this framework, states are inputs and actions are labels. Moreover, one may want to strengthen the optimality of the expert. Let $l : S \times A \to \mathbb{R}_+$ be a user-defined margin function, which allows to put some prior knowledge in the problem at hand. Typically, this margin function should satisfy $l(s_i,a_i) \ll l(s_i,a)$. For example, one may choose $l(s_i, a \neq a_i) = 1$ and $l(s_i,a_i) = 0$. Then, one searches for a score function q satisfying

$$\forall a,\ q(s_i,a_i) + l(s_i,a_i) \geq q(s,a) + l(s,a).$$

In order to find such a score function, a natural empirical cost function to be minimized is:

$$J_N(q) = \frac{1}{N}\sum_{i=1}^{N} \left(\max_{a\in A}(q(s_i,a) + l(s_i,a)) - q(s_i,a_i) \right). \tag{3}$$

This has been originally introduced in Ratliff et al. (2007). Unfortunately, it ignores a large part of data: from the transitions (s_i, a_i, s'_i) performed by the expert, only the state action couple (s_i, a_i) is used and the dynamic part (transiting to s'_i from this state action couple) is ignored. Moreover, it outputs a policy, whereas we are interested in the reward function, which is a much more compact and transferable representation of the task at hand.

Recall that we work in a sequential decision making framework. Therefore, there is a natural solution to optimization problem (3): the optimal Q-function according to the unknown reward function. Now, let θ be some parameter vector. The score function $q(s, a) = \theta^T \mu^E(s, a)$ is actually the Q-function of the expert policy according to the reward $\theta^T \psi(s)$. Moreover, assume that this parameter vector is such that inequalities (4.1) are satisfied. The optimal Q-function being stable by the greedy mechanism, see Eq.(2), this means that the expert policy π_E is optimal respectively to the reward function $\theta^T \psi(s)$ (and $\theta^T \mu^E(s, a)$ is the associated optimal Q-function). A natural cost function for computing a reward function explaining the expert behavior is therefore the following one:

$$J_N(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left(\max_{a \in A} (\theta^T \mu^E_{s_i a} + l(s_i, a)) - \theta^T \mu^E_{s_i a_i} \right).$$

Still, we need to minimize it.

Because of the max operator, this cost function is not linear nor differentiable everywhere. But it is clearly convex and piecewise linear. We can then use a subgradient descent algorithm. A subgradient of a convex function $f : \mathbb{R}^p \to \mathbb{R}$ in $x \in \mathbb{R}^p$ is defined as a vector g_x such that $\forall x' \in \mathbb{R}^p, g_x^T(x' - x) \leq f(x') - f(x)$. The set of all subgradients in x is noted $\partial_x f$. The unique subgradient in x of a differentiable convex function f is the gradient $\nabla_x f$. As A is finite, the convex and piecewise differentiable function $\max_{a \in A}[f(\cdot, a)]$, from $\mathbb{R}^p$ to $\mathbb{R}$, defined thanks to the convex differentiable functions $\{f(\cdot, a)\}_{a \in A}$, from $\mathbb{R}^p$ to $\mathbb{R}$, has one of its subgradient in $x \in \mathbb{R}^p$ equal to $\nabla_x f(x, a^*)$ with $a^* \in \arg\max_{a \in A} f(x, a)$. Thanks to this, we can minimize the cost function of interest with the following update rule (α_i denoting a usual learning rate):

$$\begin{aligned} \theta_{t+1} &= \theta_t - \alpha_t \frac{g_\theta}{\|g_\theta\|_2} \\ \text{with } g_\theta &= \frac{1}{N} \sum_{i=1}^{N} \left(\mu^E(s_i, a_i^*) - \mu^E(s_i, a_i) \right) \in \partial_\theta J_N \\ \text{and } a_i^* &\in \arg\max_{a \in A} (\theta_t^T \mu^E(s_i, a) + l(s_i, a)). \end{aligned}$$

The SCIRL algorithm only requires data from the expert, assuming μ^E is known on $\{(s_i, a)_{1 \leq i \leq N, a \in A}\}$. Unfortunately, this is usually not the case.

4.2. Estimation of μ^E

Estimating the feature expectation of the expert is a key component of our algorithm. We provide three possible solutions, depending on the available data.

First, notice that each component μ^E_j of the expert feature expectation μ^E is actually the Q-function of the expert policy according to the reward function ψ_j, the j^{th} basis function of the considered hypothesis space:

$$\mu_j^E(s,a) = E_{sa}^{\pi_E}[\sum_{t=0}^{\infty} \gamma^t \psi_j(s)]. \tag{4}$$

Therefore, it satisfies the Bellman evaluation equation (1). Assuming that the complete expert policy and the transition probabilities are known, the expert feature expectation can be exactly computed.

However, this assumption is rather strong (even if it is often done in the IRL literature). We may assume that we can simulate trajectories according to the expert policy on demand. In this case, one may estimate $\mu^E(s,a)$ for a given state-action couple with Monte Carlo. Assume that we have a set of N trajectories of length H, $\{s_0^i, a_0^i, \dots, s_{H-1}^i, a_{H-1}^i, s_H^i\}_{1\leq i\leq N}$, with $s_0^i = s$ and $a_0^i = a$, sampled according to the system dynamic and the expert policy. Then $\mu(s,a)$ is estimated as:

$$\hat{\mu}^E(s,a) = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{H} \gamma^t \psi(s_t^i),$$

Yet, as claimed at the beginning of this paper, we would like to solely use the least amount of available data, that is is the set of transitions $\{(s_i, a_i, s_i')_{1\leq i\leq N}\}$ from the expert. As each component of the feature expectation can be seen as a Q-function, see Eq. (4), it can be estimated solely from the available data using the on-policy LSTD algorithm Bradtke and Barto (1996). This is the basic idea of the LSTDμ approach Klein et al. (2011), which is an LSTD-like method estimating feature expectations. As a part of the computations can be factored, LSTD and LSTDμ computational costs have the same order of magnitude. We use LSTDμ in the experimental section, which is the less constraining scheme.

Other schemes may be envisioned. For example, one may use a Monte Carlo estimate for $\mu^E(s_i, a_i)$ (using only the available data) and some heuristic for $\mu^E(s_i, a \neq a_i)$. We left this for future work.

5. Asymptotic analysis

In this section, we prove, under two assumptions, that there is convergence of the subgradient algorithm applied to the asymptotic risk function J (that is $J = \lim_{N\to\infty} J_N$) defined as:

$$J(\theta) = \sum_{s\in S} \nu_E(s)(\max_{a\in A}(\theta^T \mu_{sa}^E + l(s,a)) - \theta^T \mu_{s\pi_E(s)}^E),$$

where ν_E is the asymptotic probability distribution on states induced by π_E. Moreover, we prove that the result of the subgradient algorithm, which is $\lim_{t\to+\infty}\theta_t = \tilde{\theta}$, is a minimum of J and that the expert policy π_E is the unique optimal policy for the reward function $\tilde{\theta}^T\psi$.

5.1. Result

Assumption 1 *It is assumed that we are able to estimate perfectly the term* $\{\mu_{sa}^E\}_{s\in S, a\in A}$.

Assumption 2 *There exists $\theta \in \mathbb{R}^p$ such that $\forall s \in S$:*

$$\theta^T \mu^E_{s\pi_E(s)} > \max_{a \in A \backslash \pi_E(s)} \theta^T \mu^E_{sa}. \tag{5}$$

Lemma 1 *Under assumptions 1 and 2, there exists a minimum θ^* of J. The elements of the set $M^* = \{\theta^* \in \mathbb{R}^p : J(\theta^*) = \inf_{\theta \in \mathbb{R}^p} J(\theta)\}$ satisfy equation (5).*

Proof Let the margin function l be such that: $\forall s \in S, \forall a \in A \backslash \pi_E(s), l(s,a) = c > l(s, \pi_E(s)) = 0$. Then $\forall \theta \in \mathbb{R}^p, J(\theta) \geq \sum_{s \in S} \nu_E(s) l(s, \pi_E(s)) = 0$. Let $\hat{\theta}$ be the vector verifying (5), it is clear that $\|\hat{\theta}\|_2 \neq 0$. Let $x > 0$ and $\hat{\theta}_x = x \frac{\hat{\theta}}{\|\hat{\theta}\|_2}$, then there exists x such that $J(\hat{\theta}_x) = 0$. Indeed, $\forall s \in S$:

$$\begin{aligned} &J(\hat{\theta}_x) = 0 \\ \Leftrightarrow &\hat{\theta}_x^T \mu^E_{s\pi_E(s)} \geq \max_{a \in A \backslash \pi_E(s)} (\hat{\theta}_x^T \mu^E_{sa} + l(s,a)) \\ \Leftrightarrow &x \geq \frac{c}{\hat{\theta}_1^T \mu^E_{s\pi_E(s)} - \max_{a \in A \backslash \pi_E(s)} \hat{\theta}_1^T \mu^E_{sa}}. \end{aligned}$$

Then all the vectors $\hat{\theta}_x$ with $x \geq c_{max} = \max_{s \in S} \frac{c}{\hat{\theta}_1^T \mu^E_{s\pi_E(s)} - \max_{a \in A \backslash \pi_E(s)} \hat{\theta}_1^T \mu^E_{sa}}$ are such that: $\inf_{\theta \in \mathbb{R}^p} J(\theta) \geq 0 = J(\hat{\theta}_x) \geq \inf_{\theta \in \mathbb{R}^p} J(\theta)$. So $\hat{\theta}_x = \inf_{\theta \in \mathbb{R}^p} J(\theta)$ for $x \geq c_{max}$. Finally let us show that the elements of the set $M^* = \{\theta^* \in \mathbb{R}^p : J(\theta^*) = \inf_{\theta \in \mathbb{R}^p} J(\theta)\}$ satisfy equation (5). As there exists x such that $J(\hat{\theta}_x) = 0 = \inf_{\theta \in \mathbb{R}^p} J(\theta)$, it implies $J(\theta^*) = 0$. So:

$$\begin{aligned} &J(\theta^*) = 0 \\ \Leftrightarrow &\theta^{*T} \mu^E_{s\pi_E(s)} \geq \max_{a \in A \backslash \pi_E(s)} (\theta^{*T} \mu^E_{sa} + l(s,a)) \\ \Rightarrow &\theta^{*T} \mu^E_{s\pi_E(s)} > \max_{a \in A \backslash \pi_E(s)} \theta^{*T} \mu^E_{sa}. \end{aligned}$$

■

Proposition 1 *Under assumptions 1 and 2, the subgradient algorithm defined by the sequence $\theta_0 = 0$, $\theta_{t+1} = \theta_t - \alpha_t \frac{g_{\theta_t}}{\|g_{\theta_t}\|_2}$ where $g_{\theta_t} = \sum_{s \in S} \nu_E(s)(\mu^E_{sa^*} - \mu^E_{s\pi_E(s)})$ with $a^* \in$* argmax$_{a \in A}$ *$\theta_t^T \mu^E_{sa} + l(s,a)$, $\sum_{t \geq 0} \alpha_t > \infty$ and $\sum_{t \geq 0} \alpha_t^2 < \infty$, is convergent: $\lim_{t \to +\infty} \theta_t = \tilde{\theta} \in \mathbb{R}^p$. Moreover we have that $J(\tilde{\theta}) = \inf_{\theta \in \mathbb{R}^p} J(\theta) = 0$. Finally the expert policy π_E is the unique optimal policy for the reward $\tilde{\theta}^T \psi$.*

Proof The cost function J is convex, proper, closed and with finite values (the action space being finite). Therefore, using the result in section 5 of Correa and Lemaréchal (1993) allows to conclude that $\lim_{t \to +\infty} \theta_t = \tilde{\theta} \in \mathbb{R}^p$ and $J(\tilde{\theta}) = \inf_{\theta \in \mathbb{R}^p} J(\theta)$. Moreover, Lemma 1 shows that $\inf_{\theta \in \mathbb{R}^p} J(\theta) = 0$. So we have $J(\tilde{\theta}) = \inf_{\theta \in \mathbb{R}^p} J(\theta) = 0$ which means that $\tilde{\theta} \in M^*$ and satisfies equation (5). This implies that, $\forall s \in S$, $\forall a \in A$:

$$\tilde{\theta}^T \mu^E_{s\pi_E(s)} > \tilde{\theta}^T \mu^E_{sa},$$

which proves that the expert policy π_E is the unique optimal policy for the reward $\tilde{\theta}^T \psi$. ■

5.2. Discussion

The assumptions 1 and 2 are quite important and we can of course wonder what results could be guaranteed if they were not true. Moreover, we have only sampled transitions, so we cannot use the asymptotic risk function J, only the empirical function J_N. Thus, in our future works, the aim will be to find, if possible, an upper bound for the term: $E_{s\sim\nu_E}[V^{\pi_E}(s) - V^{\pi_A}(s)]$ where $E_{s\sim\nu_E}$ is the expectation over the stationary probability distribution of states ν_E and π_A is the policy of the apprentice which is optimal for the reward $\tilde{\theta}^T\psi$ found by the subgradient algorithm. Of course, this upper-bound will depend on the feature expectation errors $\{\epsilon^{\mu}_{s_i a} = \hat{\mu}^E_{s_i a} - \mu^E_{s_i a}\}_{1\leq i\leq N, a\in A}$ (which may be quantified thanks to the finite sample analysis of LSTD Lazaric et al. (2010)) and on the approximation error ϵ^R which results from the fact that the unknown and bounded reward function $R_E \in \mathbb{R}^S$ may not be a linear combination of the features $\psi_{1\leq i\leq p}$. Formally, let T be the tuple $\{S, A, \{P_{sa}\}_{s\in S, a\in A}, \gamma\}$ which is an MDP without reward and $T(\tilde{R}) = \{S, A, \{P_{sa}\}_{s\in S, a\in A}, \gamma, \tilde{R}\}$ where $\tilde{R} \in \mathbb{R}^S$. We note C_R the reward-equivalence class of R: $C_R = \{\tilde{R} \in \mathbb{R}^S : \Pi(T(R)) = \Pi(T(\tilde{R}))\}$ where $\Pi(T(\tilde{R}))$ is the set of optimal deterministic policies of $T(\tilde{R})$ (see Ng et al. (1999) for a characterization of C_R). A formal definition of ϵ^R is: $\epsilon^R = \|R^* - \theta^{*T}\psi\|_\infty = \max_{s\in S}|R^*(s) - \theta^{*T}\psi(s)|$ with $(R^*, \theta^*) \in \operatorname{argmin}_{\tilde{R}\in C_R, \theta\in\mathbb{R}^p} \|\theta^T\psi - R\|_2$. Linking the empirical risk function J_N to the true risk function J could be done using a Vapnik-like analysis. The difficult thing of course is to have access to V^{π_A} without resolving the MDP directly.

6. Experiments

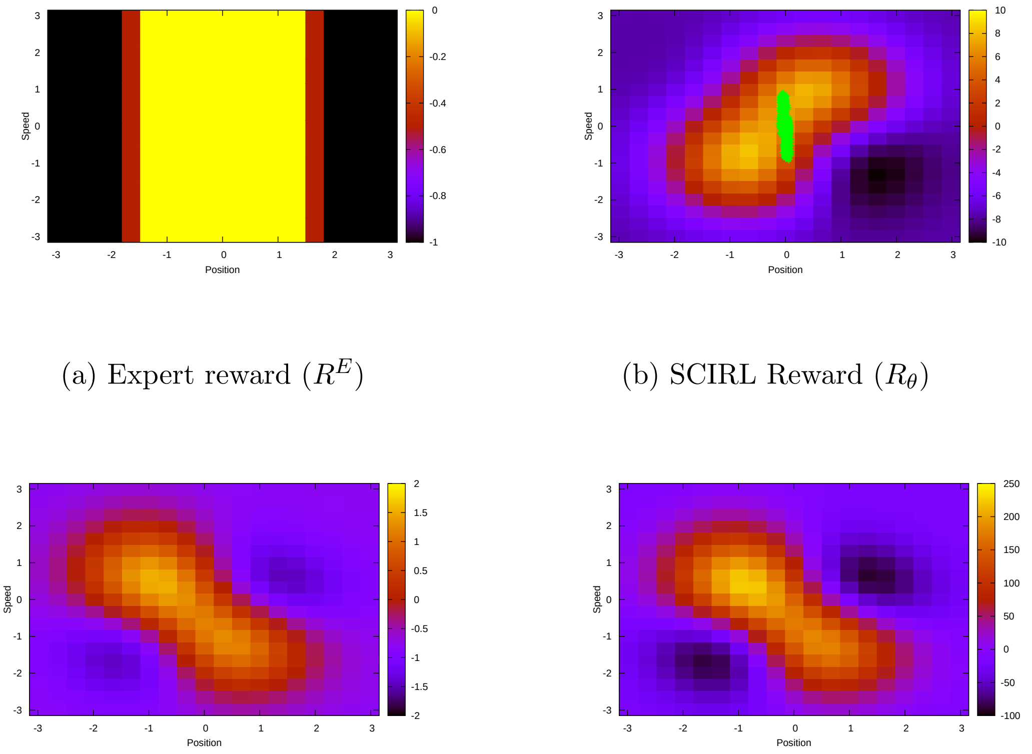
In this section, experiments on two classical benchmarks are proposed: the inverted pendulum and the car driving problems.

6.1. Inverted Pendulum

The inverted pendulum problem consists in maintaining in vertical equilibrium a pendulum with one degree of freedom by moving the cart on which it is fixed. This classical problem is extensively described in Lagoudakis and Parr (2003) from which the parameters of our experiment were taken. The state space is two-dimensional: the first dimension is the angular position of the pendulum (where the vertical position corresponds to the angle zero) and the second one is the angular speed. The policy of the expert is determined by the Least Square Policy Iteration (LSPI) algorithm from Lagoudakis and Parr (2003), using the same basis functions (Gaussian functions). The policy of the expert optimizes the reward function given Figure 1 (a). It manages systematically to maintain the pendulum during 3000 time-steps. A time-step having a value of 0.1s, the expert is interrupted after 5 minutes of successful interactions.

The main difficulty of this experiment relies in the fact that the state space is continuous and, above all, that expert trajectories (shown in green on Figure 1 (b)) are only covering a very small area of the state space. For this reason, the algorithm of Abbeel and Ng (2004) fails in finding a solution to this problem only with the transitions available from the expert demonstrations. Indeed, as shown in (Klein et al., 2011), additional data covering the whole state space is required.

6.1.1. Results



(a) Expert reward (R^E)

(b) SCIRL Reward (R_θ)

(c) Expert value function

(d) SCIRL value function

Figure 1: SCIRL results on the inverted pendulum

So as to test the SCIRL algorithm, a database D_E of transitions given by the expert policy π_E is built. It contains 10 trajectories of 300 transitions each. The margin function is $l(s,a) = 0$ if $a = \pi_E(s)$, 1 otherwise. The initial vector, θ_0, is fixed at $\begin{pmatrix} -1 & \dots & -1 \end{pmatrix}^T$, and the basis functions ($\psi : S \to \mathbb{R}^p$) used for the reward function are a Gaussian network as defined in Lagoudakis and Parr (2003). The LSTDμ algorithm Klein et al. (2011) is used to estimate the feature expectation $\mu^E(s,a)$.

Even if the reward function found by SCIRL (figure 1 (b)) is different than the expert reward (figure 1 (a)), the value functions are similar (figures 1 (c) et 1 (d)). This is an illustration that the IRL problem is ill-posed in the sense that there is not only one reward function able to explain the optimality of a policy.

With the number of samples given by the expert (3000), the apprentice is able to systematically maintain the pendulum in equilibrium during 5 minutes as the expert does. Two elements deserve to be noticed. First, the expert samples are only covering a small area of the state space (see Figure 1 (b)) and especially they are close to the vertical position. The lack of samples in the rest of the state space makes it impossible to infer with certainty the reward function. Second, the *actual* reward function R^E (Figure 1 (a)) does not live in the hypothesis space span by a linear combination of the basis functions. Despite these difficulties, SCIRL succeeds to extract a reward function leading to perform similarly to the expert.

6.2. Car driving

This task is taken from Syed and Schapire (2008). It consists in navigating a car through randomly-generated traffic on a threelane highway. The reward features are identical to Syed and Schapire (2008) paper that is: a collision feature (0 if contact with another car, and 0.5 otherwise), an off-road feature (0 if on the grass, and 0.5 otherwise), and a speed feature (0.5, 0.75 and 1 for each of the three possible speeds, with higher values corresponding to higher speeds).

6.2.1. Results

In this section, SCIRL is compared to the Multiplicative Weights for Apprenticeship Learning (MWAL) algorithm proposed by Syed and Schapire (2008). This algorithm requires the perfect knowledge of the MDP and outputs a single policy (not a reward) that imitates the expert.

The experimental setup is as follows. The highway simulator of Syed and Schapire (2008) is used[1]. It also contains an exact MDP solver for the task, the transition probabilities being available. Two reward functions were hand-tuned (θ parameters chosen by hand) so as to generate two different expert policies: one for high-speed driving and another for safe driving. The MDP solver is used to obtain the expert policies given the two reward functions. A number N of trajectories of length H is generated with each strategy providing the expert data for training the SCIRL and MWAL algorithms. The MWAL algorithm requires the knowledge of the average feature expectation $\mu_{d0}^{\pi_E}$ for the initial state (given a distribution d_0 over starting states). To reach a fair comparison with SCIRL which only uses the expert trajectories, $\mu_{d_0}^{\pi_E}$ is computed as the Monte Carlo estimate given the N trajectories, knowing that the initial state is always the same:

$$\mu_{d_0}^{\pi_E} = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{H} \gamma^t \psi(s_t^i)$$

For SCIRL, the LSTDμ algorithm is used again to approximate $\mu^E(s,a)$ from the data. For each component of μ^E, the basis functions are the same as those for the reward function. SCIRL outputs a reward function which is used to obtain a policy using the MDP solver. The SCIRL and MWAL policies are then run on the simulator for trajectories of length T. The average discounted cumulative reward, computed with regard to the initial handcrafted rewards, serves as the performance measure. It is a Monte Carlo estimate of $V^\pi(s_0)$ for each policy.

This experiment is repeated M times for each policy and results are averaged. The results are shown in Tables 1 and 2 for $N \in \{1, 3, 5, 10\}$, $H = 30$, $T = 50$ and $M = 10$.

[1] available at http://www.cs.princeton.edu/~usyed/

N	MWAL	SCIRL	V_{opt}
1	7.25 ± 0.10	7.03 ± 0.09	7.30
3	7.28 ± 0.06	7.07 ± 0.07	7.30
5	7.27 ± 0.11	6.98 ± 0.10	7.30
10	7.26 ± 0.07	7.05 ± 0.10	7.30

Table 1: SCIRL results on the car driving problem (high-speed)

N	MWAL	SCIRL	V_{opt}
1	4.95 ± 0.05	3.27 ± 0.13	5.00
3	4.85 ± 0.13	4.90 ± 0.10	5.00
5	4.90 ± 0.09	4.83 ± 0.08	5.00
10	4.98 ± 0.03	4.81 ± 0.12	5.00

Table 2: SCIRL results on the car driving problem (safe)

From these tables, one can conclude that the SCIRL method reaches approximately the same performance as MWAL (given that enough samples are provided in the case of safe driving). Yet, SCIRL doesn't need any other information than the expert demonstrations while MWAL requires solving exactly and repeatedly the MDP.

7. Conclusion

In this paper, we propose an algorithm for inverse reinforcement learning (IRL) that we call SCIRL. It is based on a classification approach in which the structure of the underlying Markov Decision Process (MDP) is taken into account.

This approach relaxes most of the constrains of other IRL algorithms found in the literature. It doesn't require the resolution of the direct problem and is able to infer a reward function which is consistent with the expert's behavior in a purely *batch* mode. A mere trajectory of the expert, even if it covers a small part of the state space, is sufficient to infer a reward function and no additional data nor a simulator is required. It also outputs a reward and not a strategy which is the real goal of the IRL paradigm. The learnt reward function, which is a compact description of the task executed by the expert, could then be optimized by an agent with abilities (action set) that are different from those of the expert which places us in in the even more challenging framework of *transfer learning*.

SCIRL is shown to compare positively to a state-of-the-art algorithm (the MWAL algorithm) on a standard benchmark (the car driving problem). It also performs well on a problem with a two-dimensional continuous state space which is sparsely explored by the expert (the inverted pendulum problem). This later problem could not be solved with state-of-the-art algorithms without additional knowledge.

Yet, SCIRL requires the computation of $\mu^E(s,a)$ everywhere on the expert trajectories. Here, we made use of the LSTDμ algorithm to estimate it. However LSTDμ shares the drawbacks of value function estimation algorithms. In the future, we plan to extend our approach so as to avoid the computation of the expert's feature expectations. Empirical tests on more complex problems will also be considered.

References

P. Abbeel and A.Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proc. ICML*, 2004.

A. Boularias, J. Kober, and Peters. Relative entropy inverse reinforcement learning. *Proc. ICAPS*, 15:20–27, 2011.

S.J. Bradtke and A.G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1):33–57, 1996. ISSN 0885-6125.

R. Correa and C. Lemaréchal. Convergence of some algorithms for convex minimization. *Mathematical Programming*, 62(1):261–275, 1993.

Edouard Klein, Matthieu Geist, and Olivier Pietquin. Batch, Off-policy and Model-Free Apprenticeship Learning. In *Proc. EWRL 2011)*, Lecture Notes in Computer Science (LNCS). Springer Verlag - Heidelberg Berlin, september 2011.

M.G. Lagoudakis and R. Parr. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4:1107–1149, 2003. ISSN 1532-4435.

A. Lazaric, M. Ghavamzadeh, and R. Munos. Finite-sample analysis of lstd. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

F. Melo and M. Lopes. Learning from demonstration using mdp induced metrics. *Machine Learning and Knowledge Discovery in Databases*, pages 385–401, 2010.

G. Neu and C. Szepesvári. Training parsers by inverse reinforcement learning. *Machine learning*, 77(2):303–337, 2009.

A.Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Proc. ICML*, pages 663–670. Morgan Kaufmann Publishers Inc., 2000.

A.Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proc. ICML*, pages 278–287, 1999.

N. Ratliff, J.A. Bagnell, and S.S. Srinivasa. Imitation learning for locomotion and manipulation. In *International Conference on Humanoid Robots*, pages 392–397. IEEE, 2007.

N.D. Ratliff, J.A. Bagnell, and M.A. Zinkevich. Maximum margin planning. In *Proc. ICML*, page 736. ACM, 2006.

S. Russell. Learning agents for uncertain environments (extended abstract). In *Annual Conference on Computational Learning Theory*, page 103. ACM, 1998.

U. Syed and R.E. Schapire. A game-theoretic approach to apprenticeship learning. *Proc. NIPS*, 20:1449–1456, 2008.

U. Syed, M. Bowling, and R.E. Schapire. Apprenticeship learning using linear programming. In *Proc. ICML*, pages 1032–1039. ACM, 2008.

B.D. Ziebart, A. Maas, J.A. Bagnell, and A.K. Dey. Maximum entropy inverse reinforcement learning. In *Proc. AAAI*, pages 1433–1438, 2008.