Around Inverse Reinforcement Learning and Score-based Classification

Matthieu Geist
IMS - MaLIS Research Group (Supélec)
Metz, France
matthieu.geist@supelec.fr

Edouard Klein[*]
ABC Team (LORIA)
Nancy, France
edouard.klein@supelec.fr

Bilal Piot[†]
UMI 2958 (GeorgiaTech - CNRS)
Metz, France
bilal.piot@supelec.fr

Yann Guermeur
ABC Team (LORIA)
Nancy, France
yann.guermeur@loria.fr

Olivier Pietquin[‡]
UMI 2958 (GeorgiaTech - CNRS)
Metz, France
olivier.pietquin@supelec.fr

Abstract

Inverse reinforcement learning (IRL) aims at estimating an unknown reward function optimized by some expert agent from interactions between this expert and the system to be controlled. One of its major application fields is imitation learning, where the goal is to imitate the expert, possibly in situations not encountered before. A classic and simple way to handle this problem is to see it as a classification problem, mapping states to actions. The potential issue with this approach is that classification does not take naturally into account the temporal structure of sequential decision making. Yet, many classification algorithms consist in learning a *score function*, mapping state-action couples to values, such that the value of the action chosen by the expert is higher than the others. The *decision rule* of the classifier maximizes the score over actions for a given state. This is curiously reminiscent of the *state-action value function* in reinforcement learning, and of the associated *greedy policy*.

Based on this simple statement, we propose two IRL algorithms that incorporate the structure of the sequential decision making problem into some classifier in different ways. The first one, SCIRL (Structured Classification for IRL), starts from the observation that linearly parameterizing a reward function by some features imposes a linear parametrization of the Q-function by a so-called feature expectation. SCIRL simply uses (an estimate of) the expert feature expectation as the basis function of the score function. The second algorithm, CSI (Cascaded Supervised IRL), applies a reversed Bellman equation (expressing the reward as a function of the Q-function) to the score function outputted by any score-based classifier, which reduces to a simple (and generic) regression step. These two algorithms come with theoretical guarantees and perform competitively on toy problems.

Keywords: Inverse Reinforcement Learning, Score-based Classification.

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.

[*]Edouard Klein is also with the IMS - MaLIS Research Group
[†]Bilal Piot is also with the IMS - MaLIS Research Group
[‡]Olivier Pietquin is also with the IMS - MaLIS Research Group

1 Introduction

Inverse reinforcement learning (IRL) aims at estimating an unknown reward function optimized by some expert agent from interactions between this expert and the system to be controlled. One of its major application fields is imitation learning, where the goal is to imitate the expert, possibly in situations not encountered before. A classic and simple way to handle this problem is to see it as a classification problem, mapping states to actions. The potential issue with this approach is that classification does not take naturally into account the temporal structure of sequential decision making. Yet, many classification algorithms consist in learning a *score function*, mapping state-action couples to values, such that the value of the action chosen by the expert is higher than the others. The *decision rule* of the classifier maximizes the score over actions for a given state. This is curiously reminiscent of the *state-action value function* in reinforcement learning, and of the associated *greedy policy*.

Based on this simple statement, we propose two IRL algorithms that incorporate the structure of the sequential decision making problem into some classifier in different ways. The first one, SCIRL (Structured Classification for IRL), starts from the observation that linearly parameterizing a reward function by some features imposes a linear parametrization of the Q-function by a so-called feature expectation. SCIRL simply uses (an estimate of) the expert feature expectation as the basis function of the score function. The second algorithm, CSI (Cascaded Supervised IRL), applies a reversed Bellman equation (expressing the reward as a function of the Q-function) to the score function outputted by any score-based classifier, which reduces to a simple (and generic) regression step. These two algorithms come with theoretical guarantees and perform competitively on toy problems.

2 From Markov Decision Processes...

A Markov Decision process (MDP) [11] is a tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma\}$ where $\mathcal{S}$ is the finite state space[1], $\mathcal{A}$ the finite actions space, $\mathcal{P} = \{P_a = (p(s'|s,a))_{1\leq s,s'\leq|\mathcal{S}|}, a \in \mathcal{A}\}$ the set of Markovian transition probabilities, $\mathcal{R} \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ the state-action reward function and γ the discount factor. A deterministic policy $\pi \in \mathcal{S}^{\mathcal{A}}$ defines the behavior of an agent. The quality of this control is quantified by the value function $v^\pi_\mathcal{R} \in \mathbb{R}^\mathcal{S}$, associating to each state the cumulative discounted reward for starting in this state and following the policy π afterwards: $v^\pi_\mathcal{R}(s) = \mathbb{E}[\sum_{t\geq 0}\gamma^t\mathcal{R}(S_t, A_t)|S_0 = s, \pi]$. An optimal policy $\pi^*_\mathcal{R}$ (according to the reward function $\mathcal{R}$) is a policy of associated value function $v^*_\mathcal{R}$ satisfying $v^*_\mathcal{R} \geq v^\pi_\mathcal{R}$, for any policy π and componentwise.

Let P_π be the stochastic matrix $P_\pi = (p(s'|s,\pi(s)))_{1\leq s,s'\leq|\mathcal{S}|}$ and $\mathcal{R}_\pi$ the reward function defined as $\mathcal{R}_\pi(s) = \mathcal{R}(s,\pi(s))$. 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) operator $T^\pi_\mathcal{R}$ (resp. $T^*_\mathcal{R}$) : $\mathbb{R}^\mathcal{S} \to \mathbb{R}^\mathcal{S}$ is defined as $T^\pi_\mathcal{R} v = \mathcal{R}_\pi + \gamma P_\pi v$ (resp. $T^*_\mathcal{R} v = \max_\pi T^\pi_\mathcal{R} v$). 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 $Q^\pi \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ adds a degree of freedom on the choice of the first action, it is formally defined as $Q^\pi_\mathcal{R}(s,a) = [T^a_\mathcal{R} v^\pi_\mathcal{R}](s)$. Let $Q^*_\mathcal{R}$ be the optimal state-action value function, an important property is that any optimal policy $\pi^*_\mathcal{R}$ is greedy respectively to it:

$$\pi^*_\mathcal{R}(s) \in \operatorname*{argmax}_{a\in\mathcal{A}} Q^*_\mathcal{R}(s,a). \tag{1}$$

Reinforcement learning and approximate dynamic programming aim at estimating the optimal control policy $\pi^*_\mathcal{R}$ when the model (transition probabilities and the reward function) is unknown (but observed through interactions with the system to be controlled) and when the state space is too large to allow exact representations of the objects of interest (as value functions or policies) [2, 12, 14]. We refer to this as the direct problem. On the contrary, (approximate) inverse reinforcement learning [10] aims at estimating a reward function for which an observed policy is (nearly) optimal. Let us call this policy the expert policy, denoted π_E. We may assume that it optimizes some unknown reward function $\mathcal{R}_E$. The aim of IRL is to compute some reward $\hat{\mathcal{R}}$ such that the expert policy is (close to be) optimal, that is such that $v^*_{\hat{\mathcal{R}}} \approx v^{\pi_E}_{\hat{\mathcal{R}}}$. We refer to this as the inverse problem. This is an ill-posed problem, for which many approaches have been proposed (many of them try to learn a reward function such that the related optimal policy matches some measure of the distribution induced by the expert policy, see for example [9] for a brief overview).

3 ... to Classification

A major application of IRL is imitation learning, which aims at generalizing the observed behavior of the expert controller π_E. Using IRL, this could be done by searching for an optimal policy according to the estimated reward $\hat{\mathcal{R}}$. A more classic approach is to cast imitation as a supervised learning problem. Assume that a trajectory $\{(s_i, a_i = \pi_E(s_i), s_{i+1})_{1\leq i\leq N}\}$ drawn by the expert is available. As the action set is finite (and usually small), one can

[1] This work can be extended to compact state spaces, up to some technical aspects.

train a classifier on the dataset $\{(s_i, a_i)_{1\leq i\leq N}\}$. A classifier learns a decision rule π_c which aims at minimizing the classification error $\epsilon_c = \mathbb{E}_{s\sim\rho_E}[\chi_{\pi_c(s)\neq\pi_E(s)}]$, with χ being the indicator function and ρ_E the stationary distribution of the policy π_E. To do so, many approaches (*e.g.*, multi-class support vector machines [5], structured classification [15], *etc.*) actually learn a score function $q \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ from the dataset, ideally satisfying $q(s, \pi(s)) > q(s, a)$, for any state s and any non-optimal action $a \neq \pi_E(s)$. The decision rule is deduced from the learnt score function as

$$\pi_c(s) \in \operatorname*{argmax}_{a\in A} q(s, a). \tag{2}$$

One can notice the similarity between Equations 1 and 2. If seeing the Q-function as a score function (ranking actions according to the expected discounted cumulative rewards) is not new, seing the classifier's score function as a state-action value function (for some unknown reward) is less usual. This can bring some ideas for new IRL algorithms, as exemplified in the next sections.

4 SCIRL (Structured Classification for IRL)

Recall that IRL aims at estimating a reward function. Assume that this reward is linearly parameterized by a set of features ϕ, $\mathcal{R}_\theta(s, a) = \theta^\top\phi(s, a)$. This naturally leads to a parametrization of the state-action value function, for any policy π:

$$\begin{aligned} Q^\pi_{\mathcal{R}_\theta}(s, a) &= \mathbb{E}[\sum_{t\geq 0}\gamma^t\mathcal{R}_\theta(S_t, A_t)|S_0 = s, A_0 = a, \pi] = \theta^\top\mathbb{E}[\sum_{t\geq 0}\gamma^t\phi(S_t, A_t)|S_0 = s, A_0 = a, \pi] \\ &= \theta^\top\mu_\pi(s, a) \\ \text{with } \mu_\pi(s, a) &= \mathbb{E}[\sum_{t\geq 0}\gamma^t\phi(S_t, A_t)|S_0 = s, A_0 = a, \pi] \end{aligned} \tag{3}$$

being called the feature expectation. To sum up, choosing a parametrization for the reward imposes a parametrization for the state-action value function of any policy, and notably for the one of the expert policy, $Q^{\pi_E}_{\mathcal{R}_\theta}(s, a) = \theta^\top\mu_E(s, a)$ (using μ_E as a shorthand for μ_{π_E}). For the unknown reward function $\mathcal{R}_E$, the expert (assumed optimal) policy π_E is greedy respectively to $Q^{\pi_E}_{\mathcal{R}_E}$. Recalling Eq. 2, it is therefore quite natural to parameterize the score function of the classifier with μ_E as linear features. This is the basic principle of the SCIRL algorithm [7], which can be summarized as follows:

1. parameterize the score function with the expert feature expectation: $q_\theta(s, a) = \theta^\top\mu_E(s, a)$;
2. learn the parameter vector θ from the dataset $\{(s_i, a_i = \pi_E(s_i))_{1\leq i\leq N}\}$, such as minimizing the classification error ϵ_c;
3. output the reward $\mathcal{R}_\theta$ (θ being the learnt parameters).

Obviously, the knowledge of μ_E is not a reasonable assumption. However, one can notice the similarity between the feature expectation (Eq. 3) and a value function. Estimating μ_E essentially reduces to an (on-policy) policy evaluation problem [6], which can be done using standard approaches such as the Least-Squares Temporal Differences (LSTD) algorithm [4].

If the idea underlying SCIRL is quite intuitive, one can wonder if it comes with some sort of guarantees. The answer is positive. One can show that, if the classification error is small and if the feature expectation is well estimated, then the expert policy π_E will be near optimal for the learnt reward $\mathcal{R}_\theta$. More formally, we have

$$0 \leq \mathbb{E}_{s\sim\rho_E}[v^*_{\mathcal{R}_{\theta_c}}(s) - v^{\pi_E}_{\mathcal{R}_{\theta_c}}(s)] \leq \frac{C_f}{1-\gamma}\left(\epsilon_Q + \epsilon_c\frac{2\gamma\|\mathcal{R}_{\theta_c}\|_\infty}{1-\gamma}\right),$$

with C_f being a standard concentration coefficient, ϵ_c being the already defined classification error and ϵ_Q quantifying how well the expert feature expectation is estimated. See [7] for details. This algorithm also performs well empirically, see Sec. 6 and [7].

5 CSI (Cascaded Supervised IRL)

The CSI algorithm [8] explores another way to combine classification with the temporal structure of sequential decision making. Assuming that the optimal state-action value function is known (for some unknown reward $\mathcal{R}$), the reward can easily be estimated by reversing the Bellman optimality equation:

$$\mathcal{R}(s, a) = Q^*_{\mathcal{R}}(s, a) - \gamma\sum_{s'\in\mathcal{S}} p(s'|s, a)\max_{a'\in\mathcal{A}} Q^*_{\mathcal{R}}(s', a').$$

Assume that using some classifier, a score function q has been learnt (with associated –greedy– decision rule π_c). Then, one can compute a reward function $\mathcal{R}_c$ associated to this score function as follows:

$$\begin{aligned}\mathcal{R}_c(s,a) &= q(s,a) - \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \max_{a' \in \mathcal{A}} q(s',a') \\ &= q(s,a) - \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) q(s', \pi_c(s')).\end{aligned}$$

However, it is usually not reasonable to assume that the dynamics is known. The reward $\mathcal{R}_c$ can still be estimated using any regression algorithm. Assume that a transition set $\{(s_j, a_j, s'_j)_{1 \leq j \leq M}\}$ is available (ideally such that the distribution over (s_j, a_j) is as uniform as possible), then $r_j = q(s_j, a_j) - \gamma q(s'_j, \pi_c(s'_j))$ is an unbiased estimate of $\mathcal{R}_c(s_j, a_j)$ and an estimate $\hat{\mathcal{R}}_c$ can be computed using any regressor from the following training set

$$\left\{ \left((s_j, a_j), r_j = q(s_j, a_j) - \gamma q(s'_j, \pi_c(s'_j)) \right)_{1 \leq j \leq M} \right\}.$$

The CSI approach can be summarized as follows:

1. estimate a score function q using any score-based classifier trained on the dataset $\{(s_i, a_i = \pi_E(s_i))_{1 \leq i \leq N}\}$;
2. estimate the reward $\hat{R}_c$ using any regressor trained on $\{((s_j, a_j), r_j = q(s_j, a_j) - q(s'_j, \pi_c(s'_j)))_{1 \leq j \leq M}\}$.

CSI is derived from a simple idea, based again on the resemblance between a score function and a state-action value function. Compared to SCIRL, it is more flexible, as any classifier (not necessarily based on a linear parametrization) and any regressor can be used. One can again wonder if this comes with some sort of guarantee, the answer is here also positive. If the classification and the regression error are small enough, then the expert policy π_E is near optimal for the estimated reward $\hat{\mathcal{R}}_c$. More formally, we have

$$0 \leq \mathbb{E}_{s \sim \rho_E}[v^*_{\hat{R}_c}(s) - v^{\pi_E}_{\hat{R}_c}(s)] \leq \frac{1}{1-\gamma} \left(\epsilon_R (1 + C_g) + \epsilon_c \frac{2 \|\hat{\mathcal{R}}_c\|_\infty}{1-\gamma} \right),$$

where C_g is another concentration coefficient (satisfying $C_g \leq C_f$) and where ϵ_R quantifies the regression error (notice that it may be easier to control the term ϵ_R than the term ϵ_Q involved in the SCIRL bound). See [8] for more details. This algorithm also performs well empirically, see Sec. 6 and [8].

6 Experiments

We illustrate the proposed approach on a car driving simulator, similar to [13]. The goal is to drive a car on a busy three-lane highway with randomly generated traffic (driving off-road is allowed on both sides). The car can move left and right, accelerate, decelerate and keep a constant speed. The expert optimizes a handcrafted reward $\mathcal{R}_E$ which favours speed, punish off-road, punishes collisions even more and is neutral otherwise.

We compare SCIRL and CSI to the "relative entropy" algorithm of [3] (which shares with SCIRL and CSI the desired property of not requiring to solve repetitively MDPs, contrary to a large part of the state of the art) and to the unstructured classifier (the one which serves as a basis for both CSI and SCIRL). Algorithms are compared according to an oracle criterion, the mean value function (averaged over a uniform distribution $\mathcal{U}$) of the learnt policy relatively to the unknown reward $\mathcal{R}_E$: $\mathbb{E}_{s \sim \mathcal{U}}[v^\pi_{\mathcal{R}_E}(s)]$, with π the policy outputted by one of the considered algorithms (π optimizes the learnt reward for IRL algorithms).

Results are reported on Fig. 6. IRL approaches work much better than the standard classification. We advocate that this is due to the fact that they take into account the temporal structure of the problem. CSI and SCIRL have similar performances (CSI being slightly –but statistically significantly– better than SCIRL), both are better than the state-of-the-art algorithm of [3].

7 Perspectives

Thanks to the similarity between score functions in classification and state-action value functions in reinforcement learning, SCIRL and CSI have been introduced. Compared to the state of the art, they are quite generic (in the sense that instantiations of these approaches can be derived by "plugging" a large class of well-studied supervised learning methods) and they do not require solving repetitively the direct RL problem (which is a common drawback of most of other algorithms). Both approaches come with theoretical guarantees and perform competitively on toy problems.

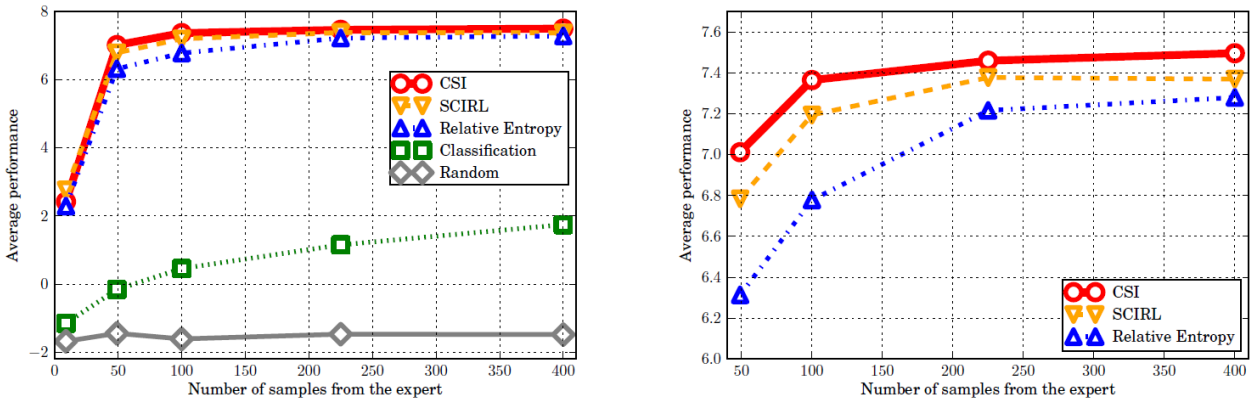

Figure 1: Results on the Highway experiment (the right panel being a zoom of the left one)

.

We plan to apply SCIRL and CSI to more challenging problems, notably to robotics and to ALE (the Arcade Learning Environment [1]). We also plan to study more deeply the theoretical aspects of these algorithms, notably if the bounds we have are tight and how these error propagations can be used to get a more practical finite sample analysis. Another perspective is to propose new algorithms based on the resemblance between score functions and state-action value functions. For example, CSI can be designed with a support vector machine (SVM) for the classification and a support vector regressor (SVR) for the regression. The two related mathematical programs can be "merged", and we are currently studying this alternative IRL algorithm.

References

[1] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *ArXiv e-prints*, 2012.

[2] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming (Optimization and Neural Computation Series, 3)*. Athena Scientific, 1996.

[3] Abdeslam Boularias, Jens Kober, and Jan Peters. Relative entropy inverse reinforcement learning. In *International Conference on. Artificial Intelligence and Statistics (AISTATS)*, 2011.

[4] Steven J. Bradtke and Andrew G. Barto. Linear Least-Squares algorithms for temporal difference learning. *Machine Learning*, 22(1-3):33–57, 1996.

[5] Yann Guermeur. A generic model of multi-class support vector machine. *International Journal of Intelligent Information and Database Systems (IJIIDS)*, 6(6):555–577, 2012.

[6] Edouard Klein, Matthieu Geist, and Olivier Pietquin. Batch, Off-policy and Model-free Apprenticeship Learning. In *European Workshop on Reinforcement Learning (EWRL)*, 2011.

[7] Edouard Klein, Matthieu Geist, Bilal Piot, and Olivier Pietquin. Inverse Reinforcement Learning through Structured Classification. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[8] Edouard Klein, Bilal Piot, Matthieu Geist, and Olivier Pietquin. A cascaded supervised learning approach to inverse reinforcement learning. In *European Conference on Machine Learning (ECML)*, 2013.

[9] Gergely Neu and Czaba Szepesvari. Training Parsers by Inverse Reinforcement Learning. *Machine Learning*, 77(2-3):303–337, 2009.

[10] Andrew Y. Ng and Stuart Russell. Algorithms for Inverse Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, 2000.

[11] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 1994.

[12] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 3rd edition, March 1998.

[13] Umar Syed and Robert Schapire. A game-theoretic approach to apprenticeship learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

[14] Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Morgan and Claypool, 2010.

[15] Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. Learning Structured Prediction Models: a Large Margin Approach. In *International Conference on Machine Learning (ICML)*, 2005.