Anti Imitation-Based Policy Learning

Michèle Sebag$^{(\boxtimes)}$, Riad Akrour, Basile Mayeur, and Marc Schoenauer

TAO, CNRS – Inria – UPSud, Université Paris-Saclay, Orsay, France
michele.sebag@lri.fr

Abstract. The *Anti Imitation-based Policy Learning* (AIPoL) approach, taking inspiration from the Energy-based learning framework (LeCun et al. 2006), aims at a pseudo-value function such that it induces the same order on the state space as a (nearly optimal) value function. By construction, the greedification of such a pseudo-value induces the same policy as the value function itself. The approach assumes that, thanks to prior knowledge, not-to-be-imitated demonstrations can easily be generated. For instance, applying a random policy on a good initial state (e.g., a bicycle in equilibrium) will on average lead to visit states with decreasing values (the bicycle ultimately falls down). Such a demonstration, that is, a sequence of states with decreasing values, is used along a standard learning-to-rank approach to define a pseudo-value function. If the model of the environment is known, this pseudo-value directly induces a policy by greedification. Otherwise, the bad demonstrations are exploited together with off-policy learning to learn a pseudo-Q-value function and likewise thence derive a policy by greedification. To our best knowledge the use of bad demonstrations to achieve policy learning is original. The theoretical analysis shows that the loss of optimality of the pseudo value-based policy is bounded under mild assumptions, and the empirical validation of AIPoL on the mountain car, the bicycle and the swing-up pendulum problems demonstrates the simplicity and the merits of the approach.

1 Introduction

Reinforcement learning aims at building optimal policies by letting the agent interact with its environment [32,33]. Among the signature challenges of RL are the facts that the agent must sufficiently explore its environment in order to ensure the optimality of its decisions, and that the consequences of its actions are delayed. Both facts raise severe scalability issues in large search spaces, which have been addressed in two ways in the last decade (Sect. 2). One way relies on the human expert's support to speed up the discovery of relevant behaviors, ranging from inverse reinforcement learning [1,27] and learning by imitation [5, 19,29] to learning from the expert's feedback [2,15,25,35]. Another way relies on the extensive interaction of the agent with its environment; it mostly operates in simulated environments, where the agent can interact with the environment and tirelessly evaluate and improve its policy without suffering exploration hazards;

© Springer International Publishing AG 2016
P. Frasconi et al. (Eds.): ECML PKDD 2016, Part II, LNAI 9852, pp. 559–575, 2016.
DOI: 10.1007/978-3-319-46227-1_35

following the pioneering TD-Gammon [34], are the Monte-Carlo Tree Search approaches [3,11,13,18] and Deep Reinforcement Learning [26].

Yet another approach is investigated in this paper, taking some inspiration from the Inverse Reinforcement Learning setting, although it almost entirely relaxes the expertise requirement on the human teacher. Specifically, the proposed approach referred to as *Anti Imitation-based Policy Learning* (AIPoL) is based on a weak prior knowledge: *when in a good state, some trivial (random or constant) policies will on average tend to deteriorate the state value, and lead to a sequence of states with decreasing value.* For instance, starting from the state where the bicycle is in equilibrium, a random policy will lead the bicycle to sooner or later fall down. This knowledge provides an operational methodology to tackle RL with very limited support from the human expert: the human expert is only asked to set the agent in a target state (e.g., the car on the top of the mountain or the bicycle in equilibrium); from this initial state, the agent applies a random policy, defining a trajectory. Contrasting with the IRL setting, this demonstration is a *bad demonstration*, showing something that should *not be done*. One merit of the approach is that it is usually much easier, and requires significantly less expertise, to generate a bad demonstration than a good one. However, such bad demonstrations provide an operational methodology to derive a good value function, as follows. Assuming that the sequence of states visited by the demonstration is such that the state value likely decreases along time (the bicycle falls down and the car arrives at the bottom of the slope), a value function can thus be derived on the state space along a learning-to-rank framework [16]. If the model of the environment is known, this value function directly defines a policy, enabling the agent to reach the target state. Otherwise, the bad demonstrations are exploited together with off-policy learning to build a Q-value function. The optimality loss of the resulting policy is bounded under mild assumptions (Sect. 3).

The empirical validation of the approach is conducted on three benchmark problems − the mountain car, the bicycle balancing and the swing-up pendulum problems − and the performances are compared to the state of the art (Sect. 4). The paper concludes with a discussion about the limitations of the AIPoL approach, and some research perspectives.

Notations. In the rest of the paper the standard Markov decision process notations $(\mathcal{S}, \mathcal{A}, p, r)$ are used: $\mathcal{S}$ and $\mathcal{A}$ respectively stand for the state and action spaces, $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ is the transition model (when known), with $p(s, a, s')$ the probability of reaching state s' after selecting action a in state s, and $r : \mathcal{S} \mapsto \mathbb{R}$ is the deterministic, bounded reward function. At the core of mainstream RL approaches are the value functions associated to every policy π. $V^\pi : \mathcal{S} \mapsto \mathbb{R}$, yields for each state the expected discounted cumulative reward gathered by following π from this state, with discount factor γ in $[0, 1]$. Likewise, Q-value $Q^\pi : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ associates to each state-action pair s, a the expected discounted cumulative reward $Q^\pi(s, a)$ gathered by selecting action a in state s and following policy π ever after.

$$V^\pi(s) = r(s) + \mathbb{E}_{s_t \sim p(s_{t-1}, \pi(s_{t-1}), \cdot)} \left[\sum\nolimits_t \gamma^t r(s_t) | s_0 = s \right]$$
$$Q^\pi(s,a) = r(s) + \gamma \sum\nolimits_{s'} p(s,a,s') V^\pi(s') \qquad (1)$$

2 State of the Art

While RL traditionally relies on learning value functions [32], their learning (using dynamic programming and approximate dynamic approaches [4]) faces scalability issues w.r.t. the size of state and action spaces.[1] In the meanwhile, there is some debate about the relevance of learning value functions to achieve reinforcement learning, on the ground that solving an RL problem and defining a policy only requires to associate an action with each state. Associating a value with each state or each state-action pair thus requires more effort than needed to solve the problem. Along this line, direct policy search (DPS) (see [9] for a comprehensive presentation) directly tackles the optimization of the policy. Furthermore, DPS does not need to rely on the Markovian assumption, thus making it possible to deal with a more agile description of the search space. DPS faces two main difficulties: (i) the choice of the parametric representation, granted that the optimization landscape involves many local optima; (ii) the optimization criterion, the policy return expectation, is approximated by an empirical estimate thereof, thus defining a noisy and expensive optimization problem.

Other RL trends addressing the limitations of learning either value functions or policies are based on the expert's help. In early RL, the human expert stayed behind the stage, providing a precious and hidden help through the design of the representation space and the reward function. In inverse reinforcement learning (IRL), the expert explicitly sets the learning dynamics through demonstrating a few expert trajectories, narrowing down the exploration in the vicinity of these trajectories [1,5,19,28]. In preference-based reinforcement learning, the expert is on the stage, interacting with the learning agent. Constrasting with IRL, preference-based RL does not require the human expert to be able to demonstrate a competent behavior; the expert is only assumed to be able to rank state-action pairs [12,17], fragments of behaviors [35], or full-length trajectories [2,15] while the RL agent achieves active ranking, focusing on the generation of most informative pairs of state-actions, behaviors or trajectories. In summary, RL increasingly puts the human expert in the learning loop, and relaxes the expertise requirement; in counterpart, the RL agent becomes more and more autonomous, striving to ask more informative preference queries to the expert and to best exploit her input [25]. Supervised learning-based policy learning, pioneered by [20], also increasingly relies on expert knowledge. In [21], a sequence of reward-sensitive classification problems is built for each time step, assuming that the optimal actions will be executed in the remaining steps (akin structured

[1] A most appealing approach sidestepping these scalability issues, Deep Reinforcement Learning (see, e.g., [26]) requires intensive interactions between the learning agent and the environment. It is outside the scope of this paper.

learning [6]). In [22], Direct Policy Iteration (DPI) handles cost-sensitive classification problems where the loss function is defined from the Q-regret of the current policy. Further work on DPI [7] use expert demonstrations. In [10], the Classification-based Approximate Policy Iteration also relies on the estimation of the Q-value based on the current policy.

As these classification-based RL approaches involve loss functions related to the value regret, they rely on the estimation of the value function, which is computationally or expertise-wise demanding. Another limitation of classification-based RL approaches is related to the ties, that is, the fact that there might be several optimal actions in a given state. A supervised learning approach addressing the tie issue is energy-based learning (EBL) [23]: When aiming at finding a classifier $h : \mathcal{X} \mapsto \mathcal{Y}$, mapping an instance space $\mathcal{X}$ onto a (possibly structured) output space $\mathcal{Y}$, the EBL claim is that in some cases, learning an energy function $g : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$, and defining $h(x)$ as $\big(\arg\max_{y \in \mathcal{Y}} g(x, y)\big)$ leads to significantly more robust results than learning directly h. An appealing argument for the EBL approach is that g is only defined up to a monotonous transformation.[2] Along these lines, RL might be content with learning an energy-like value function $U(s, a)$, such that policy $\pi_U(s) = \arg\max_{a \in \mathcal{A}} U(s, a)$ is a (nearly) optimal policy, regardless of whether U satisfies the Bellman optimality equation. Next section will present a methodology for learning such an energy-based value function, however with limited help from the human expert.

3 Overview of AIPoL

3.1 Rationale

AIPoL is based on the assumption that, while quite some expertise is required to perform an expert demonstration, it is usually very easy to generate terrible demonstrations, in a sense defined below. Let V^* be the (unknown) optimal value function, satisfying the Bellman equation for some $0 < \gamma \leq 1$:

$$V^*(s) = r(s) + \gamma \arg\max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} p(s, a, s') V^*(s') \tag{2}$$

Definition 1. *A* counter-demonstration *(*CD*), is a sequence of states $(s_1, \ldots s_T)$ with decreasing V^* values, i.e., s.t.*

$$\forall\, 1 \leq i < j \leq T,\ V^*(s_i) > V^*(s_j)$$

Definition 2. *Let $\mathcal{E} = \{CD_1, \ldots CD_n\}$ be a set of n* CDs*, with $CD_i = (s_{i,t}, t = 1 \ldots T_i)$. The learning-to-rank problem associated to $\mathcal{E}$ is defined from the set of constraints $s_{i,t} \prec s_{i,t'}$ $\forall i$ in $[1..n]$ and $\forall 1 \leq t < t' \leq T_i$.*

$$\textit{Find } \widehat{U} = \arg\min_{U : \mathcal{S} \mapsto \mathbb{R}} \{\mathcal{L}(U, \mathcal{E}) + \mathcal{R}(U)\} \tag{3}$$

with $\mathcal{L}$ a ranking loss function and $\mathcal{R}(U)$ a regularization term.

[2] For any non-decreasing scalar function f, g and $f \circ g$ define the same classifier.

Following the learning-to-rank setting [16] and denoting $(A)_+ = \max(0, A)$, the ranking loss function used in AIPoL is the sum of the hinge loss between $U(s_{i,t})$ and $U(s_{i,t'})$ over all pairs (t, t') such that $1 \leq t < t' < T_i$, with $i = 1 \ldots n$:

$$\mathcal{L}(U, \mathcal{E}) = \sum_{i=1}^{n} \sum_{1 \leq t < t' \leq T_i} \left(U(s_{i,t'}) + 1 - U(s_{i,t})\right)_+ \tag{4}$$

A solution $\widehat{U}$ of Pb (3) (solved using e.g., [16,24]) is thereafter referred to as *pseudo-value function*. Of course, it is unlikely that pseudo-value $\widehat{U}$ satisfies the Bellman optimality equation (2). Nevertheless, it will be seen that the optimality of the policy based on $\widehat{U}$ can be assessed in some cases, when V^* and $\widehat{U}$ define sufficiently similar orderings on the state space.

AIPoL relies on the assumption that CDs can be easily generated without requiring any strong expertise. CDs are additionally required to sufficiently visit the state space, or the interesting regions of the state space. How much does sufficient mean will be empirically assessed in Sect. 4.

3.2 AIPoL with a Known Transition Model

In this section, the transition model is assumed to be known. In the case of a **deterministic transition** model, let $s'_{s,a}$ denote the arrival state when selecting action a in state s.

Definition 3. *In the deterministic transition model case, the greedy policy based on $\widehat{U}$ selects the action leading to the arrival state with maximal $\widehat{U}$ value:*

$$\pi_{\widehat{U}}(s) = \arg\max_{a \in \mathcal{A}} \left\{ \widehat{U}(s'_{s,a}) \right\} \tag{5}$$

It immediately follows from this definition that:

Proposition 1. *In the deterministic transition setting, if $\widehat{U}$ derives the same order on the state space as the optimal value function V^*, i.e.,*

$$\forall (s, s') \in \mathcal{S},\ (\widehat{U}(s) > \widehat{U}(s')) \Leftrightarrow (V^*(s) > V^*(s'))$$

then greedy policy $\pi_{\widehat{U}}$ is an optimal policy.

In the case of a **stochastic transition** model, by slight abuse of notation let $s'_{s,a}$ denote a state drawn after distribution $p(s, a, \cdot)$.

Definition 4. *In the stochastic transition model case, the greedy policy based on $\widehat{U}$ selects the action leading to the maximal $\widehat{U}$ value expectation:*

$$\pi_{\widehat{U}}(s) = \arg\max_{a \in \mathcal{A}} \left\{ \mathbb{E}_{p(s,a,\cdot)} \widehat{U}(s'_{s,a}) \right\} \tag{6}$$

Some assumptions on the regularity of the transition model, on V^* and $\widehat{U}$ are required to establish a result analogous to Proposition 1 in the stochastic case. The main assumption regards the sub-Gaussianity of the transition model. Note that this assumption does hold, of course, for Gaussian transition models, and also in robotic settings, where the distance between two consecutive states is bounded due to physical and mechanical constraints.

A guarantee on the $\pi_{\hat{U}}$ optimality can be obtained under the following assumptions:

Proposition 2. *Assuming (1) a continuous optimal value function V^* on $\mathcal{S}$; (2) a pseudo-value $\widehat{U}$ deriving the same order on the state space as V^*; (3) a β-sub-Gaussian transition model, that is,*

$$\forall t \in \mathbb{R}^+, \mathbb{P}(||\mathbb{E}s'_{s,a} - s'_{s,a}||_2 > t) < 2e^{-\beta t^2}$$

(4) $\widehat{U}$ being Lipschitz with constant M, that is,

$$\forall s, s' \in \mathcal{S}, |\widehat{U}(s) - \widehat{U}(s')| < L||s - s'||_2$$

(5) for all every s, there exists a margin between the best and the second best action after $\widehat{U}$, such that:

$$\forall a' \neq a = \pi_{\hat{U}}(s), \mathbb{E}\widehat{U}(s'_{s,a}) > \mathbb{E}\widehat{U}(s'_{s,a'}) + M \tag{7}$$

Then, if $2L < M\beta$, $\pi_{\hat{U}}$ is an optimal policy.

Proof. The idea of the proof is the following: Consider the average value $\mathbb{E}V^*(s'_{s,a})$, where the expectation is taken over $p(s,a,\cdot)$. By continuity of V^* there exists a state noted $s'_{s,a,V}$ in the neighborhood of $\mathbb{E}s'_{s,a}$ such that $V^*(s'_{s,a,V}) = \mathbb{E}V^*(s'_{s,a})$. Likewise there exists a state $s'_{s,a',V}$ such that $V^*(s'_{s,a',V}) = \mathbb{E}V^*(s'_{s,a'})$.

Let us assume by contradiction that optimal policy π^* is such that $\pi^*(s) = a' \neq a$. It follows that

$$V^*(s'_{s,a',V}) > V^*(s'_{s,a,V})$$

and therefore, as $\widehat{U}$ and V^* define same orderings on $\mathcal{S}$,

$$\widehat{U}(s'_{s,a',V}) > \widehat{U}(s'_{s,a,V}) \tag{8}$$

Let us denote $K(u) = ||s'_{s,a',V} - s'_{s,a',u}||$. For any $\varepsilon > 0$, we have

$$\begin{aligned}
|\widehat{U}(s'_{s,a',V}) - \mathbb{E}_u\widehat{U}(s'_{s,a'})| &= \mathbb{E}_u|\widehat{U}(s'_{s,a',V}) - \widehat{U}(s'_{s,a',u})| \\
&= \mathbb{E}_{K(u)<\varepsilon}|\widehat{U}(s'_{s,a',V}) - \widehat{U}(s'_{s,a',u})| \\
&\qquad + \mathbb{E}_{K(u)>\varepsilon}|\widehat{U}(s'_{s,a',V}) - \widehat{U}(s'_{s,a',u})| \\
&\leq L\varepsilon + L\int_\varepsilon^\infty t \cdot 2e^{-\beta t^2} dt \quad (*) \\
&\leq L\varepsilon + \frac{L}{\beta}e^{-\beta\varepsilon^2} < L(\varepsilon + \frac{1}{\beta})
\end{aligned}$$

where (*) is derived using the Lipschitz property of $\widehat{U}$ and the sub-Gaussian property of the transition model.

If $L(\varepsilon + \frac{1}{\beta}) < M/2$, then from Eq. (8) it comes:

$$\mathbb{E}\widehat{U}(s'_{s,a'}) + M/2 \geq \widehat{U}(s'_{s,a',V}) > \widehat{U}(s'_{s,a,V}) \geq \mathbb{E}\widehat{U}(s'_{s,a}) - M/2$$

which contradicts the margin assumption (Eq. 7), hence the result. □

Overall, in the known transition model case, AIPoL proceeds by generating the CDs, defining the associated ranking problem, finding a solution $\widehat{U}$ thereof and building policy $\pi_{\hat{U}}$ by greedification of $\widehat{U}$ (Algorithm 1).

Algorithm 1. Model-based AIPoL

```
Input: $\mathcal{E} = \{CD_1, \dots CD_n\}$
$\widehat{U} = \arg\min \{\mathcal{L}(U, \mathcal{E}) + \mathcal{R}(U)\}$                    (Pb (3))
        with $\mathcal{L}(U, \mathcal{E})$ (from Eq. 4) and $\mathcal{R}(U)$ an $L_2$ regularization.
Return: $\pi_{\hat{U}}$                                                   (Eq. (5))
```

3.3 AIPoL with Unknown Transition Model

When the transition model is unknown, a pseudo Q-value $\widehat{Q}$ is built from the pseudo-value $\widehat{U}$ learned from the CDs using off-policy learning. The intuition is that, given $\widehat{U}$ and triplets (s_1, a_1, s'_1) and (s_2, a_2, s'_2), the pseudo Q-value of state-action pair (s_1, a_1) is lower than for state action pair (s_2, a_2) if state s'_1 has a lower pseudo-value than s'_2 ($\widehat{U}(s'_1) < \widehat{U}(s'_2)$).

Definition 5. *With the same notations as above, let $\widehat{U}$ be a pseudo value function solution of Pb (3), and let*

$$\mathcal{G} = \{(s_i, a_i, s'_i), i = 1 \dots m\}$$

be a set of state-action-next-state triplets. The learning-to-rank problem associated to $\mathcal{G}$ is defined from the set of ranking constraints $(s_i, a_i) \prec (s_j, a_j)$ for all i, j such that $\widehat{U}(s'_i) < \widehat{U}(s'_j)$.

$$\textit{Find } \widehat{Q} = \underset{Q:\mathcal{S}\times\mathcal{A}\mapsto\mathbf{R}}{\arg\min} \{\mathcal{L}(Q, \mathcal{G}) + \mathcal{R}(Q)\} \tag{9}$$

with $\mathcal{L}$ a loss function and $\mathcal{R}(Q)$ a regularization term.

In AIPoL, the ranking loss function is set to:

$$\mathcal{L}(Q, \mathcal{G}) = \sum_{i=1}^{m} \sum_{j=1}^{m} y_{ij} \left(Q((s_i, a_i) + 1 - Q(s_j, a_j)\right)_+ \tag{10}$$

with $y_{ij} = 1$ iff $\widehat{U}(s'_i) < \widehat{U}(s'_j)$ and 0 otherwise.

Definition 6. *Letting $\widehat{Q}$ be a pseudo Q-value function learned from Pb (9), policy $\pi_{\hat{Q}}$ is defined as:*

$$\pi_{\hat{Q}}(s) = \arg\max_{a \in \mathcal{A}} \left\{ \widehat{Q}(s,a) \right\} \tag{11}$$

Some more care must however be exercized in order to learn accurate pseudo Q-value functions. Notably, comparing two triplets (s_1, a_1, s'_1) and (s_2, a_2, s'_2) when s_1 and s_2 are too different does not yield any useful information. Typically, when $\widehat{U}(s_1) \gg \widehat{U}(s_2)$, it is likely that $\widehat{U}(s'_1) > \widehat{U}(s'_2)$ and therefore the impact of actions a_1 and a_2 is very limited: in other words, the learned $\widehat{Q}$ does not deliver any extra information compared to $\widehat{U}$. This drawback is addressed by filtering the constraints in Pb (9) and requiring that the triplets used to learn $\widehat{Q}$ be such that:

$$||s_1 - s_2||_2 < \eta \tag{12}$$

with η a hyper-parameter of the AIPoL algorithm (set to 10 % or 1 % of the state space diameter in the experiments). Empirically, another filter is used, based on the relative improvement brought by action a_1 in s_1 compared to action a_2 in s_2. Specifically, the constraint $(s_1, a_1) \succ (s_2, a_2)$ is generated only if selecting action a_1 in s_1 and going to s'_1 results in a higher value improvement than selecting action a_2 in s_2 and going to s'_2:

$$\widehat{U}(s_1) - \widehat{U}(s'_1) > \widehat{U}(s_2) - \widehat{U}(s'_2) \tag{13}$$

Overall, the model-free AIPoL (Algorithm 2) proceeds by solving the model-based problem (Algorithm 1), using traces (s, a, s') to build the learning-to-rank problem (9), finding a solution $\widehat{Q}$ thereof and building policy $\pi_{\hat{Q}}$ by greedification of $\widehat{Q}$.

Algorithm 2. Model-free AIPoL

```
Input: $\mathcal{E} = \{CD_1, \ldots CD_n\}$
Input: $\mathcal{G} = \{(s_i, a_i, s'_i), i = 1 \ldots m\}$
$\widehat{U} = \arg\min \{\mathcal{L}(U, \mathcal{E}) + \mathcal{R}(U)\}$                    (Pb (3))
        with $\mathcal{L}(U, \mathcal{E})$ (Eq. 4) and $\mathcal{R}(U)$ an $L_2$ regularization.
$\widehat{Q} = \arg\min \{\mathcal{L}(Q, \mathcal{G}) + \mathcal{R}(Q)\}$                    (Pb (9))
        with $\mathcal{L}(Q, \mathcal{G})$ (Eq. 10) and $\mathcal{R}(Q)$ an $L_2$ regularization.
Return: $\pi_{\hat{Q}}(s)$                                                   (Eq. 11)
```

3.4 Discussion

In the model-based setting, the quality of the AIPoL policy essentially depends on sufficiently many CDs to be generated with limited expertise, and on the coverage of the state space enforced by these CDs. In many benchmark problems, the goal is to reach a target state (the car on the mountain or the bicycle in equilibrium). In such cases, CDs can be generated by simply setting the starting state to the target state, and following a random or constant policy ever after. Such a trivial policy is likely to deviate from the good state region, and visit states with lower and lower values, thus producing a CD. Additionally, the CDs will sample the neighborhood of the target state; the pseudo value function $\widehat{U}$ learned from these CDs will then provide a useful guidance toward the (usually narrow) good region. The intuition behind AIPoL is similar to that of TD-gammon [34]: the value function should steadily increase when reaching the desirable states, regardless of satisfying the Bellman equation.

In the known transition model case, under the assumption that the pseudo value $\widehat{U}$ induces the same ordering on the state space as V^*, policy $\pi_{\hat{U}}$ is optimal in the deterministic case (Proposition 1). Under additional assumptions on the regularity of the optimal value function V^*, of $\widehat{U}$ and on the transition noise, the optimality still holds in the stochastic transition case (Proposition 2). This is true even though no reward is involved in the definition of $\widehat{U}$.

In the unknown transition model case, the constraints used to learn $\widehat{Q}$ introduce a systematic bias, except in the case where the reward function r is equal to 0 almost everywhere. Let us consider the deterministic case for simplicity. By definition,

$$Q^*(s_1, a_1) = r(s_1) + \gamma V^*(s'_1)$$

If the reward function is equal to 0 almost everywhere, then with high probability:

$$(V^*(s'_1) > V^*(s'_2)) \Leftrightarrow (Q^*(s_1, a_1) > Q^*(s_2, a_2))$$

Then, if $\widehat{U}$ induces the same ordering on the state space as V^*, the constraints on $\widehat{Q}$ derived from the traces are satisfied by Q^*, and the learning-to-rank problem (9) is noiseless.

Otherwise, the difference between the instantaneous rewards $r(s_1)$ and $r(s_2)$ can potentially offset the difference of values between s'_1 and s'_2, thus leading to generate noisy constraints. The generation of such noisy constraints is alleviated by the additional requirement on the constraints (Eq. 13), requiring the $\widehat{U}$ value gap between s_1 and s'_1 be larger than between s_2 and s'_2.

Overall, the main claim of the AIPoL approach is that generating CDs, though requiring much less expertise than that required to generate quasi expert behavior or asking the expert to repair or compare behaviors, can still yield reasonably competent policies. This claim will be examined experimentally in next section.

4 Experimental Validation

This section presents the experimental setting used for the empirical validation of AIPoL, before reporting and discussing the comparative results.

4.1 Experimental Setting

The AIPoL performance is assessed on three standard benchmark problems: The *mountain car* problem, using SARSA as baseline [31]; The *bicycle balancing* problem, using preference-based reinforcement learning as baseline [2,35]; the *under-actuated swing-up pendulum* problem, using [14] as baseline. In all experiments, the pseudo-value $\widehat{U}$ and $\widehat{Q}$ functions are learned using Ranking-SVM with Gaussian kernel [16]. The hyper-parameters used for all three benchmark problems are summarized in Table 1.

The first goal of the experiments is to investigate how much knowledge is required to generate sufficiently informative CD, enabling AIPoL to yield state-of-art performances. This issue regards (i) the starting state of an CD, (ii) the controller used to generate an CD, (iii) the number and length of the CDs. A second goal is to examine whether and to which extent the performances obtained in the model-free setting (transition model unknown) are degraded compared to the model-based setting. A third goal is to investigate the sensitivity of the AIPoL performance w.r.t. the algorithm hyper-parameters (Table 1), including: (i) the number and length of the CDs; (ii) the Ranking-SVM hyper-parameters C (weight of the data fitting term) and σ (Gaussian kernel width); (iii) the AIPoL parameter η used to filter the ordering constraints used to learn $\widehat{Q}$ (Eq. 12).

Table 1. AIPoL hyper parameters on the three benchmark problems: number and length of CDs, starting state and controllers used to generate the CDs; hyper parameters used to learn pseudo-value $\widehat{U}$ (parameters C_1 and $1/\sigma_1^2$; hyper-parameters used to learn pseudo value $\widehat{Q}$ (parameters C_2 and $1/\sigma_2^2$; # of constraints).

		Mountain car	Bicycle	Pendulum
CD	number	1	20	1
	length	1,000	5	1,000
	starting state	target st	random	target st
	controller	neutral	random	neutral
$\widehat{U}$	C_1	10^3	10^3	10^{-5}
	$1/\sigma_1^2$	10^{-3}	10^{-3}	.5
$\widehat{Q}$	nb const	500	5,000	–
	C_2	10^3	10^3	–
	$1/\sigma_2^2$	10^{-3}	10^{-3}	–

4.2 The Mountain Car

Following [31], the mountain car problem involves a 2D state space (position, speed), and a discrete action space (backward, neutral position, forward). The friction coefficient ranges in [0, .02]. AIPoL is compared to the baseline SARSA with $\lambda = .9$, $\alpha = .05/10m$, $\varepsilon = 0$, with 9×9 tile coding, with 100 episodes for

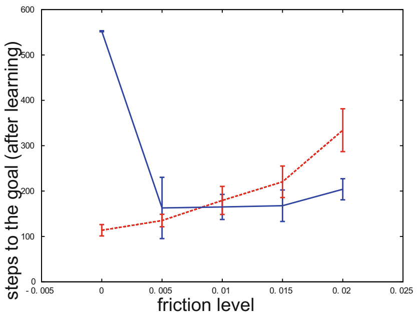


Fig. 1. Mountain car: Number of time steps to reach the goal for AIPoL (solid blue line) and SARSA (dashed red line) *vs* friction value (average and standard deviation over 20 runs) (Color figure online)

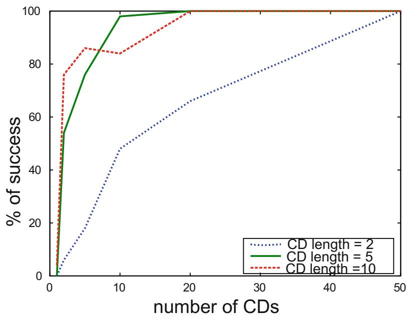


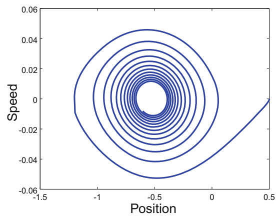
Fig. 2. Bicycle (model-based setting): Fraction of success of AIPoL w.r.t. number and length of CDs.

learning, stopping an episode whenever a terminal state is reached, or after 1000 steps. AIPoL uses 1 CD of length 1000. In the unknown transition model case, the 500 constraints are generated from random trajectories where pairs (s, a) and $(s'a')$ were selected subject to constraints (Eqs. 12 and 13) with proximity threshold $\eta = 10\,\%$.

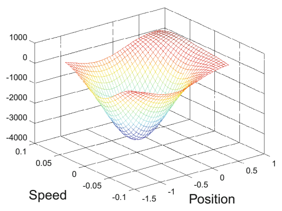
The performances are excellent for both known (1 CD with length 1,000) and unknown transition model cases, with negligible runtime. Figure 3a depicts the CD in the 2D (position, speed) space, starting from the target state and selecting the neutral action for 1,000 time steps. The pseudo-value $\widehat{U}$ function learned by AIPoL and the approximation of V^* learned by SARSA in two representative runs are respectively displayed in Fig. 3b and c, showing that $\widehat{U}$ is very smooth compared to that approximate value.

Figure 3e and f display the policy based on $\widehat{Q}$ in the model-free case, and the optimal policy learned by SARSA, suggesting that the policy learned by AIPoL is much simpler than for SARSA. Figure 3d shows a typical trajectory based on the AIPoL policy in the model-free case.

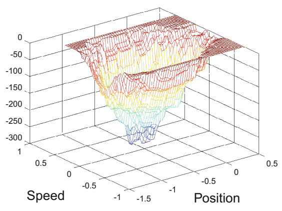
The sensitivity analysis shows that the main parameter governing the AIPoL performance on the mountain car problem is the friction (Fig. 1). For low friction values, the dynamics is quasi reversible as there is no loss of energy; accordingly, letting the car fall down from the target state does not generate a sequence of states with decreasing value (the value of the state intuitively increases with its energy). In the low friction region (friction in [0, .05]), AIPoL is dominated by SARSA. For high friction values ($>$ .02), the car engine lacks the required power to climb the hill and both approaches fail. For moderate friction values (in [.01, .02]), AIPoL significantly outperforms SARSA.



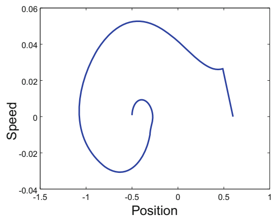
(a) CD: Mcar falling from the target



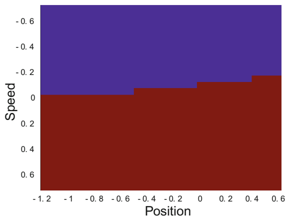
(b) AIPoL pseudo-value



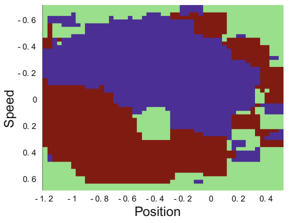
(c) SARSA value after 1000 iterations



(d) AIPoL learned trajectory



(e) AIPoL policy map



(f) SARSA policy map

Fig. 3. The mountain car problem: Comparative evaluation of AIPoL and SARSA in the model-free setting, on two representative runs (friction = .01). The policy map visually displays the selected action for each state in the 2D (position, speed) space (best seen in color: red= forward, blue= backward, green= neutral). (Color figure online)

4.3 The Bicycle Balancing

Following [20], the bicycle balancing problem involves a 4-dimensional state space (the angles of the handlebar and of the bicycle and the angular velocities), and a 3-action action space (do nothing, turn the handlebar left or right, lean the rider left or right). The goal is to maintain the bicycle in equilibrium for 30,000 time steps; note that a random controller starting from the equilibrium state $(0, 0, 0, 0)$ leads the bicycle to fall after 200 steps on average.

Known Transition Model Case. To assess the sensitivity of the approach w.r.t. the starting state, the CDs are generated using a random starting state and a random controller. The definition of the policy $\pi_{\hat{U}}$ (Eq. 5) is adapted to account for the fact that, due to the temporal discretization of the transition model [20], the effect of action a_t on the angle values is only visible in state s_{t+2}. Some look-ahead is thus required to define the greedy policy $\pi_{\hat{U}}$. Formally, the selected action is obtained by maximization of the value obtained after two time steps:

$$\pi_{\hat{U}}(s) = \underset{a \in \mathcal{A}}{argmax} \; \{\max_{a' \in \mathcal{A}} \mathbb{E}\widehat{U}(s''_{s'_{s,a},a'})\}$$

Given this definition, AIPoL only requires 20 CD of length 5 to learn a competent policy, keeping the bicycle in equilibrium for over 30,000 time steps with high probability (in all of the 100 runs, Fig. 2). In comparison, the state of the art requires a few dozen trajectories to be ranked by the expert (15 for [2] and 20 for [35]), the starting point of which is close to the equilibrium. With same starting point, AIPoL reaches the goal (keeping the bicycle in equilibrium 100 times out of 100 runs) with a single CD of length 5.

Unknown Transition Model Case. The ordering constraints on the state-action pairs likewise take into account the temporal discretization and the delayed impact of the actions. Formally, from sequences $(s_1, a_1, s'_1, a'_1, s''_1)$ and $(s_2,a_2, s'_2, a'_2, s''_2)$, constraint $(s_1, a_1) \succ (s_2, a_2)$ is generated if

$$\widehat{U}(s''_1) - \widehat{U}(s_1) > \widehat{U}(s''_2) - \widehat{U}(s_2)$$

The proximity threshold is set to $\eta = 1\,\%$. 5,000 constraints are required to achieve the same performance as in the model-based setting.

4.4 The Under-Actuated Swing-Up Pendulum

Following [14], the swing-up pendulum involves a 2-dimensional state space $(s = (\theta, \dot{\theta}))$ and a 3 action space. The pendulum has two equilibrium states, a stable one and an unstable one. The goal, starting from the stable state (bottom position) is to reach the unstable one (top position). The task is under-actuated since the agent has a limited torque and must gain some momentum before achieving its swing-up. The task is stopped after 20 s or when the agent successfully maintains the pendulum in an up-state ($\theta < \pi/4$) for 3 consecutive time steps. Only the model-based setting has been considered for the pendulum problem, with a computational cost of 3 s.

On the pendulum problem, the sensitivity of the approach w.r.t. the Ranking-SVM hyper-parameters is displayed in Fig. 4. Two failure regions appear when learning the pseudo-value $\widehat{U}$ from a single CD of length 1,000: if the kernel width is too small, there is no generalization and the pendulum does not reach the top.

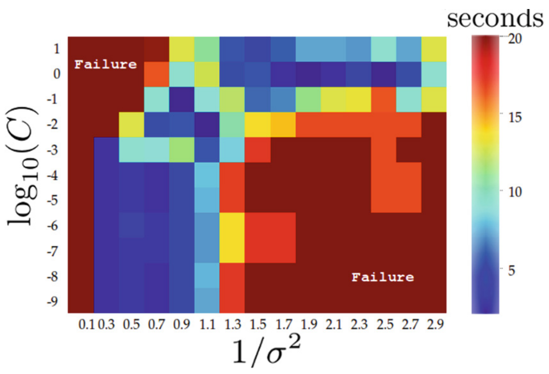


Fig. 4. The pendulum problem: Sensitivity of AIPoL performance (average over 10 runs) w.r.t Ranking-SVM hyper-parameters C and $1/\sigma^2$ (see text for details).

If the kernel width is too large, the accuracy is insufficient and the pendulum does not decrease its speed sufficiently early: it reaches the top and falls down on the other side. For good hyper-parameter settings ($C = 1$ and $1/\sigma^2$ ranging in $[1.7, 2.7]$; or C and $1/\sigma^2$ very small), the pendulum reaches the target state in 3 s and stays there. The AIPoL performance matches the state of the art [14], which relies on a continuous variant of the Bayes-adaptive planning, and achieves the goal (staying in an up-state for 3 s) after on average 10 s of interaction.

5 Discussion and Perspectives

The AIPoL approach to reinforcement learning has been presented together with an analytic and empirical study of its performances. Its main novelty is twofold compared to the state of the art. On the one hand, AIPoL learns a pseudo-value function and derives a policy by greedification; computationally-wise, it tackles a much less complex problem than e.g., inverse reinforcement learning [1,19] (learning a reward function and solving a complete RL problem) or preference-based RL [2] (learning a return value and solving a difficult optimization problem). In addition, AIPoL significantly relaxes the requirements on the human teacher. She is not required to perform (nearly) optimal demonstrations as in IRL, or to compare, and possibly repair, trajectories as in preference-based learning: she is only required to know *what will go wrong*.

In the mountain car and the pendulum problems, AIPoL uses informed CDs (starting in the target state). In the bicycle problem however, the CD sequences start in a random state. In this latter case, the pseudo value function coarsely leads to get away from state regions with low value: the inadequacy of the pseudo value in low value regions is (almost) harmless should the learning agent spend little or no time in these regions.

A first limitation of the AIPoL approach, illustrated on the bicycle problem, is when the effect of the selected actions is fully visible after a few time steps, that is, when the transition dynamics involves some latency. This latency occurs when some coordinates of action a_t (e.g. the angular speed) make no difference on state s_{t+1} and only influence e.g. $s_{t+\ell}$. In this case some look-ahead is required in the greedification process. The extra computational cost is in $\mathcal{O}(|\mathcal{A}|^\ell)$, exponential in the latency ℓ and in the size of the action set. Note that this phenomenon, distinct from the delayed rewards of the actions, only concerns the transition dynamics: An alternative could be to commit to an action for a sequence of time-steps, rather than just a single step [26]. A second limitation of the approach is that the computational cost of building the Q-value function might be high (e.g. on the swing-up pendulum) as it scales up quadratically with the number of ranking constraints. Other ranking approaches with linear learning complexity will be considered (e.g., based on neural nets [30] or ranking forests [8]) to address this limitation. A third and most important limitation concerns the non-reversible MDP case, where the transition from s to s' might take much longer than from s' to s. Further work is on-going to address the non reversible case. A main theoretical perspective is to investigate the quality of the AIPoL policy in the unknown transition model case, depending on the structure of the MDP dynamics and the sparsity of the reward function.

Acknowledgments. We wish to thank the anonymous reviewers for their insightful comments, which helped to greatly improve the paper.

References

1. Abbeel, P.: Apprenticeship learning and reinforcement learning with application to robotic control. Ph.d. thesis, Stanford, CA, USA (2008). aAI3332983
2. Akrour, R., Schoenauer, M., Sebag, M., Souplet, J.C.: Programming by feedback. In: Proceedings of the International Conference on Machine Learning (ICML). JMLR Proceedings, vol. 32, pp. 1503–1511. JMLR.org (2014)
3. Auer, P., Ortner, R.: Logarithmic online regret bounds for undiscounted reinforcement learning. In: Schölkopf, B., et al. (eds.) NIPS 19, pp. 49–56. MIT Press (2007)
4. Bertsekas, D.P.: Dynamic Programming and Optimal Control, 2nd edn. Athena Scientific (2000)
5. Calinon, S., Billard, A.: Active teaching in robot programming by demonstration. In: Proceedings of the 16th IEEE RO-MAN, pp. 702–707. IEEE (2007)
6. Chang, K., Krishnamurthy, A., Agarwal III, A., H.D., Langford, J.: Learning to search better than your teacher. In: Bach, F.R., Blei, D.M. (eds.) Proceedings of the 32nd ICML. JMLR Proceedings, vol. 37, pp. 2058–2066. JMLR.org (2015)
7. Chemali, J., Lazaric, A.: Direct policy iteration with demonstrations. In: Yang, Q., Wooldridge, M. (eds.) Proceedings of the 24th IJCAI, pp. 3380–3386. AAAI Press (2015)
8. Clémençon, S., Depecker, M., Vayatis, N.: Ranking forests. J. Mach. Learn. Res. **14**(1), 39–73 (2013)
9. Deisenroth, M.P., Neumann, G., Peters, J.: A survey on policy search for robotics. Found. Trends Robot. **2**(1–2), 1–142 (2013)
10. Farahmand, A.M., Precup, D., Barreto, A.M.S., Ghavamzadeh, M.: Classification-based approximate policy iteration. IEEE Trans. Autom. Control **60**(11), 2989–2993 (2015)
11. Filippi, S., Cappé, O., Garivier, A.: Optimism in reinforcement learning and kullback-leibler divergence. In: Proceedings of the Allerton Conference on Communication, Control, and Computing, pp. 115–122 (2010). https://hal.archives-ouvertes.fr/hal-00476116
12. Fürnkranz, J., Hüllermeier, E., Cheng, W., Park, S.: Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. Mach. Learn. **89**(1–2), 123–156 (2012)
13. Gelly, S., Silver, D.: Combining online and offline knowledge in UCT. In: Ghahramani, Z. (ed.) Proceedings of the 24th ICML, pp. 273–280. ACM (2007)
14. Guez, A., Heess, N., Silver, D., Dayan, P.: Bayes-adaptive simulation-based search with value function approximation. In: Ghahramani, Z., et al. (eds.) NIPS 27, pp. 451–459. Curran Associates, Inc. (2014)
15. Jain, A., Wojcik, B., Joachims, T., Saxena, A.: Learning trajectory preferences for manipulators via iterative improvement. In: Burges, C.C., et al. (eds.) NIPS 26 (2013)
16. Joachims, T.: A support vector method for multivariate performance measures. In: Raedt, L.D., Wrobel, S. (eds.) Proceedings of the 22nd ICML, pp. 377–384. ACM (2005)

17. Knox, W.B., Stone, P., Breazeal, C.: Training a robot via human feedback: a case study. In: Herrmann, G., et al. (eds.) Proceedings of the 5th International Conference on Social Robotics, pp. 460–470 (2013)
18. Kocsis, L., Szepesvári, C.: Bandit based Monte-Carlo planning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS(LNAI), vol. 4212, pp. 282–293. Springer, Heidelberg (2006). doi:10.1007/11871842_29
19. Konidaris, G., Kuindersma, S., Barto, A., Grupen, R.: Constructing skill trees for reinforcement learning agents from demonstration trajectories. In: Lafferty, J.D., et al. (eds.) NIPS 23, pp. 1162–1170. MIT Press (2010)
20. Lagoudakis, M.G., Parr, R., Bartlett, L.: Least-squares policy iteration. JMLR **4**, 2003 (2003)
21. Langford, J., Zadrozny, B.: Relating reinforcement learning performance to classification performance. In: Proceedings of the 22nd ICML, pp. 473–480. ACM (2005)
22. Lazaric, A., Ghavamzadeh, M., Munos, R.: Analysis of a classification-based policy iteration algorithm. In: Fürnkranz, J., Joachims, T. (eds.) Proceedings of the 27th ICML, pp. 607–614. Omnipress (2010)
23. LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., Huang, F.: A tutorial on energy-based learning. In: Bakir, G., Hofman, T., Schölkopf, B., Smola, A., Taskar, B. (eds.) Predicting Structured Data. MIT Press (2006)
24. Li, P., Burges, C.J.C., Wu, Q.: Mcrank: Learning to rank using multiple classification and gradient boosting. In: Advances in Neural Information Processing Systems, vol. 20, pp. 897–904 (2007)
25. Loftin, R.T., Peng, B., MacGlashan, J., Littman, M.L., Taylor, M.E., Huang, J., Roberts, D.L.: Learning behaviors via human-delivered discrete feedback: modeling implicit feedback strategies to speed up learning. Auton. Agent. Multi-Agent Syst. **30**(1), 30–59 (2016)
26. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**, 529–533 (2015)
27. Ng, A.Y., Russell, S.: Algorithms for inverse reinforcement learning. In: Langley, P. (ed.) Proceedings of the International Conference on Machine Learning (ICML), pp. 663–670. Morgan Kaufmann (2000)
28. Ross, S., Bagnell, J.A.: Reinforcement and imitation learning via interactive no-regret learning. CoRR abs/1406.5979 (2014)
29. Schaal, S., Ijspeert, A., Billard, A.: Computational approaches to motor learning by imitation. Philos. Trans. R Soc. Lond. B Biol. Sci. **358**(1431), 537–547 (2003)
30. Severyn, A., Moschitti, A.: Learning to rank short text pairs with convolutional deep neural networks. In: Baeza-Yates, R.A., Lalmas, M., Moffat, A., Ribeiro-Neto, B.A. (eds.) SIGIR, Research and Development in Information Retrieval, pp. 373–382. ACM (2015)
31. Sutton, R.S.: Generalization in reinforcement learning: successful examples using sparse coarse coding. In: Touretzky, D.S., et al. (eds.) NIPS 8, pp. 1038–1044. MIT Press (1995)
32. Sutton, R.S., Barto, A.G.: Introduction to Reinforcement Learning, 1st edn. MIT Press, Cambridge (1998)
33. Szepesvári, C.: Algorithms for Reinforcement Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, San Rafael (2010)

34. Tesauro, G., Sejnowski, T.J.: A parallel network that learns to play backgammon. Artif. Intell. **39**(3), 357–390 (1989)
35. Wilson, A., Fern, A., Tadepalli, P.: A bayesian approach for policy learning from trajectory preference queries. In: Bartlett, P.L., et al. (eds.) NIPS 25, pp. 1142–1150 (2012)