Direct Value Learning: Reinforcement Learning and Anti-Imitation

Riad Akrour, Basile Mayeur, Michèle Sebag

Direct Value Learning: Reinforcement Learning by Anti-Imitation

Riad Akrour, Basile Mayeur, Michele Sebag

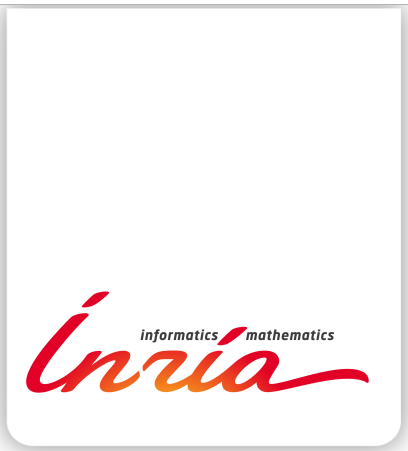
RESEARCH REPORT

N° 8836

December 2015

Project-Team TAO

ISSN 0249-6399 ISRN INRIA/RR--8836--FR+ENG



Direct Value Learning: Reinforcement Learning by Anti-Imitation

Riad Akrour, Basile Mayeur, Michele Sebag *

Project-Team TAO

Research Report n° 8836 — December 2015 — 18 pages

Abstract: The value function, at the core of the Reinforcement Learning framework, associates to each state the expected discounted cumulative reward which can be gathered after visiting this state. Given an (optimal) value function, an (optimal) policy is most simply derived by "greedification", heading in each time step toward the neighbor state with maximal value. The value function can be built by (approximate) dynamic programming, albeit facing severe scalability limitations in large state and action spaces.

As an alternative, the DIVA approach presented in this paper proposes to directly learn the value function, taking inspiration from the Energy-based learning framework (LeCun et al. 2006) and searching for a pseudo-value function such that it induces the same local 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.

DIVA uses bad demonstrations to infer the pseudo-value, as bad demonstrations are notoriously easier to generate than expert ones (e.g., as used in Inverse Reinforcement Learning). Typically, 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. In the model-based RL setting, this pseudo-value directly induces a policy by greedification. In the model-free RL setting, 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.

The DIVA approach and the use of bad demonstrations to achieve direct value learning is original to our best knowledge. The loss of optimality of the pseudo value-based policy is analyzed and it is shown that it is bounded under mild assumptions. Finally, the comparative experimental validation of DIVA on the mountain car, the bicycle and the swing-up pendulum problems demonstrates the simplicity and the merits of the approach.

Key-words: Reinforcement learning; inverse reinforcement learning; ranking; value-based RL

* TAO — CNRS — INRIA, Univ. Paris-Sud, Université Paris-Saclay, FirstName.Name@lri.fr

RESEARCH CENTRE
SACLAY – ÎLE-DE-FRANCE

1 rue Honoré d'Estienne d'Orves
Bâtiment Alan Turing
Campus de l'École Polytechnique
91120 Palaiseau

Apprentissage direct de fonction de valeur: Renforcement par Anti-Imitation

Résumé : La fonction de valeur, au coeur de l'apprentissage par renforcement, associe à chaque état l'espérance des gains cumulés recueillis par l'agent après avoir visité cet état. La fonction de valeur produit ainsi un résumé du futur, tenant compte de la fonction de gain, de la dynamique de l'environnement et de la politique suivie. Une fonction de valeur construite à partir d'une politique π donnée définit par un procédé glouton une politique meilleure que π: aller à chaque pas de temps vers l'état voisin de valeur maximale.

La construction d'une fonction de valeur par programmation dynamique (approchée) suivant les équations de Bellman ne passe toutefois pas à l'échelle pour de grands espaces d'états et d'actions. Une alternative, inspirée de l'apprentissage par énergie (energy-based learning, LeCun et al. 2006) est proposée dans cet article: l'algorithme DIVA apprend directement une fonction de pseudo-valeur. Par opposition avec l'apprentissage par démonstration, qui utilise des demonstrations expertes pour inférer la fonction de gain, DIVA exploite des mauvaises démonstrations pour inférer la fonction de pseudo-valeur. Les mauvaises démonstrations sont notoirement plus aisées à générer que les bonnes; ainsi, il suffit d'utiliser un contrôleur aléatoire en partant d'un bon état initial (e.g., partant d'une bicyclette en équilibre) pour visiter des états de valeurs successives décroissantes (jusqu'à la chute de la bicyclette).

Ces mauvaises démonstrations, générées à partir de faibles connaissances a priori, sont utilisées par DIVA pour apprendre une fonction de pseudo-valeur par apprentissage d'ordonnancement classique. Cette fonction de pseudo-valeur induit directement une politique dans le cas où le modèle de transition est connu (model-based RL). Sinon, la pseudo-valeur sur les états est exploitée avec des trajectoires générées indépendamment (off-policy learning) pour apprendre une fonction de pseudo Q-valeur sur les paires état-action, dont est extraite la politique cherchée.

L'approche DIVA et l'utilisation de mauvaises démonstrations pour apprendre directement une fonction de valeur sont à notre connaissance originales. La perte d'optimalité de la politique induite est analysée et peut être bornée sous certaines conditions. La validation expérimentale de DIVA sur trois problèmes bien étudiés de l'état de l'art, la voiture sur la montagne, la bicyclette et le pendule inversé démontre la simplicité et les mérites de l'approche proposée comparé à l'état de l'art.

Mots-clés : Apprentissage par renforcement; apprentissage par renforcement inverse; ranking; apprentissage de fonction de valeur

1 Introduction

Reinforcement learning aims at building optimal policies by letting the agent interact with its environment [29, 30]. 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: vast amounts of trajectories are required to observe the whole space and accurately estimate the delayed effects of actions.

These issues have been addressed along two main trends in the last decade (more in section 2). The former trend, thereafter referred to as educated RL, relies on the human expert's support to avoid the lengths of serendipitous exploration and speed up the discovery of relevant behaviors. Two main research avenues have been investigated within educated RL. The first one, including inverse reinforcement learning [26, 1], learning by imitation [27], or learning from demonstrations [21, 7], exploits a few quasi-optimal trajectories demonstrated by the expert, in order to learn the reward function, or the value function, and possibly the environment dynamics. The second one relies on the expert in the loop, providing feedback about the agent's trajectories [32, 2, 17, 3].

The latter trend, or autonomous RL, relies on the illimited 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. A pioneering work along this line is the TD-Gammon [31], using self-play to refine its value function and build a world champion of backgammon. Other approaches build upon the Multi-Armed Bandit framework and its sequential extensions [20, 12, 4, 9], illustrated by the Monte-Carlo Tree Search MoGo system [13, 11], first to match the performance of professional Go players. While early TD-Gammon used handcrafted features, and MoGo used the explicit description of the goban, new advances in autonomous RL with deep learning show that the agent can build from scratch an efficient representation of the state space, using as raw features as the screen pixels in Atari video games [25].

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 *Direct Value Learning* (DIVA) is based on a very general and weak form of knowledge: *when in a good state, any inexpert policy will on average tend to deteriorate the state, and lead to states with lower and lower value*. For instance, starting from the state where the bicycle is in equilibrium, any inexpert policy will lead the bicycle to sooner or later fall down. This principle, commonly known as Murphy's law, 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 what 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. After the Murphy's law, 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 [18], exploiting the fact that each visited state has a lower

value than the previously visited state. In the model-based RL framework, this value function directly defines a policy, enabling the agent to reach the target state. In the model-free framework, the value function is exploited together with off-policy learning to build a Q-value function. The optimality loss of the resulting policy is bounded under mild assumptions (section 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 (section 4).

The paper concludes with a discussion about the limitations of the DIVA 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 [29], where $\mathcal{S}$ 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, 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. To each policy $\pi : \mathcal{S} \mapsto \mathcal{A}$, mapping each state onto an action, is associated the value function $V^\pi : \mathcal{S} \mapsto \mathbb{R}$, yielding for each state the expected discounted cumulative reward gathered by following π from this state, with $0 \leq \gamma < 1$ a discount factor:

$$\begin{aligned} V^\pi(s) &= r(s) + \mathbb{E}_{s_t \sim p(s_{t-1}, \pi(s_{t-1}), \cdot)} \left[\sum_t \gamma^t r(s_t) | s_0 = s\right] \\ &= r(s) + \gamma \sum_{s'} p(s, \pi(s), s') V^\pi(s') \end{aligned} \tag{1}$$

Likewise, the Q-function $Q^\pi : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ defines the expected discounted cumulative reward $Q^\pi(s, a)$ gathered by selecting action a in state s and following policy π ever after:

$$Q^\pi(s, a) = r(s) + \gamma \sum_{s'} p(s, a, s') V^\pi(s') \tag{2}$$

2 State of the art

Mainstream RL approaches [29] invoke the value function V^π, indicating for each state the best cumulative reward which can possibly be expected in this state, relatively to a policy π and discount factor γ. Based on value function V^π, a better policy π' can be built by "greedification", by selecting in each state s the action a leading to the best value in expectation:

$$\pi'(s) = \underset{a \in \mathcal{A}}{argmax} \sum_{s' \in \mathcal{S}} p(s, a, s') V^\pi(s')$$

The so-called policy iteration process (building the value function associated to the current policy π, defining a new policy from V^π, and iterating the process) is known to converge toward a globally optimal policy − usually in a few steps. Unfortunately, the building of the value function based on dynamic programming and approximate dynamic approaches [5] boils down to solving fixed point equations (the Bellman equation, Eq. 1) and is impractical in large state and action spaces. The Q-learning approach, gradually updating the Q value function by requiring all actions to be taken in all states and enforcing a judicious exploration *vs* exploitation trade-off likewise faces scalability issues. The recent advances of Q-learning within the so-called Deep Reinforcement Learning [25] overcome the scalability issue by using a deep neural

net (DQN) as non-linear continuous approximation of the Q-value function. The optimization of the DQN weights is achieved using a stochastic gradient descent, ensuring the convergence of the DQN toward a fixed point and its generalization over the state space, and enforcing its stability through a sophisticated update mechanism. Additionally, state/action/next-state triplets are randomly selected in the trajectories (randomized replay) to prevent the update of the DQN weights from suffering from the spurious correlations among the successive states in a trajectory.

However, the relevance of learning value functions is debated in the RL community, on the ground that solving an RL problem only requires to associate an action to each state – the sought policy. Associating a value to each state or each state-action pair thus requires more effort than needed to solve the RL problem. Along this line, direct policy search (DPS) (see [8] for a comprehensive presentation) directly tackles the optimization of the policy. Additionally, 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[1]. 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 (see for instance [16]).

As mentioned in the introduction, another RL trend addressing the limitations of learning either value functions or policies is 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, the expert explicitly sets the learning dynamics through demonstrating a few expert trajectories, narrowing down the exploration in the vicinity of these trajectories [26, 1, 27, 21, 7]. 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 [19, 10], fragments of behaviors [32], or full-length trajectories [17, 3] 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.

Discussion

The approach proposed in this paper reconsiders the role of the value function. To our best knowledge (with the notable exception of [31]), the value function has mostly been considered in RL in the context of the Bellman equations (Eq. 1), aggregating the reward function, the current policy and the transition dynamics. By construction, value function V^π enables a policy better than π to be characterized by simple greedification of V^π.

Another framework related to value function learning is that of energy-based learning (EBL) [23, 15]. Let us consider the goal of finding a classifier $h : \mathcal{X} \mapsto \mathcal{Y}$, mapping

[1] The Markovian assumption, at the core of the fixed point formulation of the value function, requires the state description to provide every information relevant to action selection, thus possibly carrying on the memory of the past trajectory.

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 $\mathcal{E} : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$, and defining $h(x)$ as

$$h(x) = \underset{y \in \mathcal{Y}}{argmax}\, \mathcal{E}(x, y)$$

leads to significantly more robust results than learning directly h, typically when the output space is high-dimensional or continuous, or when there are ties (a same x can be associated to distinct ys). An appealing argument for the EBL approach is that $\mathcal{E}$ is only defined up to a monotonous transformation[2].

Along these lines, it is suggested that RL might be content with learning an energy-like value function $U(s, a)$, such that policy

$$\pi_U(s) = \underset{a \in \mathcal{A}}{argmax}\, U(s, a)$$

is a (nearly) optimal policy, regardless of whether U satisfies the Bellman optimality equation. Next section presents a methodology for learning such an energy-based value function, with as little help from the human expert as possible.

3 Overview of DIVA

After presenting the DIVA principle, this section describes the algorithm in the model-based and model-free settings. Only the case of a continuous state space is considered in the following ($\mathcal{S} \subseteq \mathbb{R}^d$).

3.1 Principle

DIVA is based on the remark that, while quite some expertise is required to perform an expert demonstration, it is usually very easy to generate a bad demonstration. Informally, a bad demonstration is a sequence of states such that the (unknown) optimal value of the states decreases along the sequence.

Let V^* be the optimal value function, i.e., that satisfies

$$V^*(s) = \max_{\pi} V^{\pi} = r(s) + \underset{a \in \mathcal{A}}{argmax} \sum_{s' \in \mathcal{S}} p(s, a, s') V^*(s') \tag{3}$$

Definition 1 *Let a bad demonstration, thereafter referred to as* Murphy State Sequence[3] *(MSS), be defined as*

$$MSS = (s_1, \ldots s_T)\ s.t.\ \forall\, 1 \leq i < j \leq T,\ V^*(s_i) > V^*(s_j)$$

Definition 2 *Let $\mathcal{E} = \{MSS_1, \ldots MSS_n\}$ be a set of n MSSs, with $MSS_i = (s_{i,t}, t = 1 \ldots T_i)$. The learning to rank problem associated to $\mathcal{E}$ is defined from the set of constraints $s_{i,t} \prec s_{i,t'}$ forall $1 \leq i \leq n$ and forall $1 \leq t < t' \leq T_i$.*

$$\textit{Find } \hat{U} = \underset{U : \mathcal{S} \mapsto \mathbf{R}}{argmin} \{\mathcal{L}(U, \mathcal{E}) + \mathcal{R}(U)\} \tag{4}$$

[2] By noting that for any non-decreasing scalar function g, $\mathcal{E}$ and $g o \mathcal{E}$ define the same classifier.

[3] In honor of Murphy's law, *If something can go wrong, it will*.

with $\mathcal{L}$ a loss function, e.g. the sum over all MSSs ($i = 1 \dots n$) and all pairs of time steps $1 \leq t < t' \leq T_i$ of the hinge loss,

$$\mathcal{L}(U, \mathcal{E}) = \sum_{i=1}^{n} \sum_{1 \leq t < t' \leq T_i} max(0, U(s_{i,t'}) + 1 - U(s_{i,t}))$$

and $\mathcal{R}(U)$ a regularization term.

Problem (4) can be solved using state-of-art ranking algorithms [18, 24]. A solution $\hat{U}$ thereof is thereafter referred to as *pseudo-value function*. Naturally, it is unlikely that pseudo-value $\hat{U}$ satisfies the Bellman optimality equation (3). Nevertheless, it will be seen that the optimality of the policy based on $\hat{U}$ can be assessed in some cases, when V^* and $\hat{U}$ define sufficiently similar orderings on the state space.

In summary, the core assumption done in DIVA is that MSSs can be easily generated without requiring any strong expertise. MSSs 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 section 4.

3.2 Model-based DIVA

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. The greedy policy based on $\hat{U}$ selects the action leading to the arrival state with maximal $\hat{U}$ value:

$$\pi_{\hat{U}}(s) = \underset{a \in \mathcal{A}}{argmax} \left\{ \hat{U}(s'_{s,a}) \right\} \tag{5}$$

It immediately follows from this definition that:

Proposition 3.1 *In the deterministic transition setting, if $\hat{U}$ derives the same order on the state space as the optimal value function V^*, i.e.,*

$$(\hat{U}(s) > \hat{U}(s')) \Leftrightarrow (V^*(s) > V^*(s'))$$

then the greedy policy $\pi_{\hat{U}}$ 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 $p(s, a, \cdot)$. The greedy policy based on $\hat{U}$ selects the action leading to the maximal $\hat{U}$ value expectation:

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

It is clear that Eq. (6) boils down to Eq. (5) when the support of distribution $p(s, a, \cdot)$ is restricted to $s'_{s,a}$.

Three assumptions on the regularity of the transition model, on V^* and $\hat{U}$ are required to establish a result analogous to Proposition 3.1 in the stochastic case. The first assumption regards the bounded noise of the transition model: an arrival state $s'_{s,a}$ is required to be sufficiently close to its expectation $\mathbf{E}_{p(s,a,\cdot)}\ s'_{s,a}$ (noted $\mathbf{E}s'_{s,a}$ when no confusion is to fear) in terms of Euclidean distance on $\mathcal{S} \subset \mathbb{R}^d$. Note that, while this assumption does not hold for e.g., Gaussian transition noise, it is realistic in robotic

settings, where the distance between two consecutive states is bounded due to physical and mechanical constraints.
Bounded Transition Noise hypothesis There exists $\epsilon > 0$ such that, for all state action pairs (s, a), $||\mathbb{E}s'_{s,a} - s'_{s,a}|| < \epsilon$ for all $s'_{s,a}$ drawn after $p(s, a, \cdot)$.

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

Proposition 3.2 *Assume*

- *A bounded transition noise*
- *The optimal value function V^* is continuous on $\mathcal{S}$*
- *The pseudo value $\hat{U}$ is Lipschitz-continuous with constant M,*

$$\forall s, s' \in \mathcal{S}, |\hat{U}(s) - \hat{U}(s')| < M||s - s'||$$

If, for all states s, the second best action after $\hat{U}$ is significantly worse than the best one (margin assumption):

$$\forall a' \neq a = \pi_{\hat{U}}(s), \hat{U}(\mathbb{E}s'_{s,a}) > \hat{U}(\mathbb{E}s'_{s,a'}) + 2M\epsilon \tag{7}$$

Then $\pi_{\hat{U}}$ is an optimal policy.

Proof 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 $\mathbf{E}s'_{s,a}$ 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 $\hat{U}$ and V^* define same orderings on $\mathcal{S}$,

$$\hat{U}(s'_{s,a',V}) > \hat{U}(s'_{s,a,V})$$

But $|\hat{U}(s'_{s,a',V}) - \mathbb{E}\hat{U}(s'_{s,a'})| < M\epsilon$ due to $\hat{U}$ Lipschianity and the transition noise boundedness. Likewise, $|\hat{U}(s'_{s,a,V}) - \mathbb{E}\hat{U}(s'_{s,a})| < M\epsilon$.
Hence if

$$\hat{U}(s'_{s,a',V}) > \hat{U}(s'_{s,a,V})$$

then we have

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

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

Overall, the model-based DIVA (Alg. 1) proceeds by generating the MSSs, defining the associated ranking problem, finding a solution $\hat{U}$ thereof and building policy $\pi_{\hat{U}}$ by greedification of $\hat{U}$.

Algorithm 1 Model-based DIVA

Input: MSS_i, $i = 1 \dots n$
$\hat{U}$ = Solution(Ranking problem (4) from MSSs)
Return: $\pi_{\hat{U}}$ (Eq. (5))

3.3 Model-free DIVA

When the transition model is unknown, an approximation of the optimal Q-value function, referred to as pseudo Q-value, is built from the pseudo-value $\hat{U}$ learned from the MSSs using off-policy learning. Informally, given $\hat{U}$ and triplets (s_1, a_1, s'_1) and (s_2, a_2, s'_2), the idea is that if state s'_1 has a lower pseudo-value than s'_2 ($\hat{U}(s'_1) < \hat{U}(s'_2)$) then the pseudo Q-value of state-action pair (s_1, a_1) is lower than for state action pair (s_2, a_2).

Definition 3 *As in Definition 2, let $\mathcal{E} = \{MSS_1, \dots MSS_n\}$ be a set of n MSSs. Let $\hat{U}$ be a pseudo value function defined on $\mathcal{S}$, solution of Problem (4), 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 constraints $(s_i, a_i) \prec (s_j, a_j)$ forall i, j such that $\hat{U}(s'_i) < \hat{U}(s'_j)$.

$$\textit{Find } \hat{Q} = \underset{Q:\mathcal{S}\times\mathcal{A}\mapsto\mathbf{R}}{\textit{argmin}} \{\mathcal{L}(Q, \mathcal{G}) + \mathcal{R}(Q)\} \tag{8}$$

with $\mathcal{L}$ a loss function, e.g. the sum − over all 4-uples (s_i, a_i, s_j, a_j) such that $\hat{U}(s'_i) < \hat{U}(s'_j)$ − of the hinge loss,

$$\mathcal{L}(Q, \mathcal{G}) = \sum_{i=1}^{m} \sum_{j=1}^{m} 1_{\hat{U}(s'_i) < \hat{U}(s'_j)} \cdot max(0, Q((s_i, a_i) + 1 - Q(s_j, a_j))$$

and $\mathcal{R}(Q)$ a regularization term.

Letting $\hat{Q}$ be a pseudo Q-value function learned from Problem (8), policy $\pi_{\hat{Q}}$ is defined as:

$$\pi_{\hat{Q}}(s) = \underset{a\in\mathcal{A}}{\textit{argmax}} \left\{\hat{Q}(s, a)\right\} \tag{9}$$

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 $\hat{U}(s_1) \gg \hat{U}(s_2)$, it is likely that $\hat{U}(s'_1) > \hat{U}(s'_2)$ and therefore the impact of actions a_1 and a_2 is very limited: in other words, the learned $\hat{Q}$ does not deliver any extra information compared to $\hat{U}$. This drawback is addressed by filtering the constraints in Problem (8) and requiring that the triplets used to learn $\hat{Q}$ are such that

$$||s_1 - s_2|| < \eta \tag{10}$$

with η a hyper-parameter of the DIVA 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 results in improving the state value more than selecting action a_2 in state s_2:

$$\hat{U}(s_1) - \hat{U}(s'_1) > \hat{U}(s_2) - \hat{U}(s'_2) \tag{11}$$

Overall, the model-free DIVA (Alg. 2) proceeds by solving the model-based problem (Alg. 1), using traces (s, a, s') to build the learning-to-rank problem (8), finding a solution $\hat{Q}$ thereof and building policy $\pi_{\hat{Q}}$ by greedification of $\hat{Q}$.

Algorithm 2 Model-based DIVA

Input: $\mathcal{E} = \{MSS_1, \ldots MSS_n\}$, a set of n MSSs
Input: $\mathcal{G} = \{(s_i, a_i, s'_i), i = 1 \ldots m\}$
$\hat{U}$= Solution(Ranking problem (4) from $\mathcal{E}$)
$\hat{Q}$ = Solution(Ranking problem (8) from $\hat{U}$ and $\mathcal{G}$)
Return: $\pi_{\hat{Q}}(s) = \underset{a \in \mathcal{A}}{argmax} \left\{ \hat{Q}(s, a) \right\}$ (Eq. (9))

3.4 Discussion

In the model-based setting, the quality of the DIVA policy essentially depends on sufficiently many MSSs to be generated with limited expertise, and on the coverage of the state space enforced by these MSSs. 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, MSSs can be generated by simply setting the starting state as target state, and following a random or neutral policy ever after. Such an inexpert policy will with high probability deviate from the good state region, and visit states with lower and lower values, thus producing an MSS. Additionally, the MSSs will sample the neighborhood of the target state; the pseudo value function $\hat{U}$ learned from these MSSs will then provide a useful guidance toward the (usually narrow) good region. The intuition behind DIVA is similar to that of TD-gammon [31]: the value function should steadily increase when reaching the desirable states, regardless of satisfying the Bellman equation.

In the model-based setting, under the assumption that the pseudo value $\hat{U}$ induces the same ordering on the state space as V^* $\hat{U}$ is optimal in the deterministic case. Under mild additional assumptions on the regularity of the optimal value function V^*, of $\hat{U}$ and on the transition noise, the optimality still holds in the stochastic transition case.

In the model-free setting, the constraints used to learn $\hat{Q}$ introduce a systematic bias, except in the case where 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:

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

and then, if $\hat{U}$ induces same ordering on the state space as V^*, the constraints on $\hat{Q}$ derived from the traces are satisfied by Q^*, and the learning-to-rank problem (8) 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. 11), requiring the $\hat{U}$ value gap between s_1 and s'_1 be larger than between s_2 and s'_2. Further work is needed to define the assumptions on the reward function, required to provide optimality guarantee on $\hat{Q}$.

Overall, the main claim of the DIVA approach is that generating MSSs, though requiring much less expertise than generating a good behavior (akin inverse reinforcement learning ([1]), or putting the expert in the loop like preference-learning based RL approaches ([19, 32, 17, 3]), can still yield reasonably competent policies. This claim will be examined experimentally.

4 Experimental Validation

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

4.1 Experimental Setting

The DIVA performance is assessed on three standard benchmark problems: The *mountain car* problem, using SARSA as baseline [28]; The *bicycle balancing* problem, using preference-based reinforcement learning as baseline [32, 3]; the *under-actuated swing-up pendulum* problem, using [14] as baseline. In all experiments, the pseudo-value $\hat{U}$ and $\hat{Q}$ functions are learned using Ranking-SVM with Gaussian kernel [18]. 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 MSS, enabling DIVA to yield state-of-art performances. This issue regards the starting state of an MSS and the controller used to generate an MSS, the number and length of the MSSs.

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 DIVA performance w.r.t. the algorithm hyper-parameters (Table 1), including: i) the number and length of the MSSs; ii) the Ranking-SVM hyper-parameters C (weight of the data fitting term) and σ (Gaussian kernel width); iii) the DIVA parameter η used to filter the ordering constraints used to learn $\hat{Q}$ (Eq. 10).

4.2 The mountain car

Following [28], 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].

The performances are excellent in both model-based (1 MSS with length 1,000) and model-free (500 constraints) settings, with negligible runtime. Fig. 1.a depicts the MSS in the 2D (position, speed) space, starting from the target state and selecting the neutral action for 1,000 time steps. The pseudo-value $\hat{U}$ function learned by DIVA and value V^* learned by SARSA in two representative runs are respectively displayed in Fig. 1.b and 1.c, showing that $\hat{U}$ is very smooth compared to V^*.

Fig. 1.e and 1.f display the policy based on $\hat{Q}$ in the model-free case, and the optimal policy learned by SARSA, suggesting that the policy learned by DIVA is much

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

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

simpler than for SARSA. Fig. 1.d shows a typical trajectory based on the DIVA policy in the model-free case. The proximity threshold is set to $\eta = 10\%$.

The sensitivity analysis shows that the main parameter governing the DIVA performance on the mountain car problem is the friction (Fig. 2). 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]), DIVA 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]), DIVA significantly outperforms SARSA.

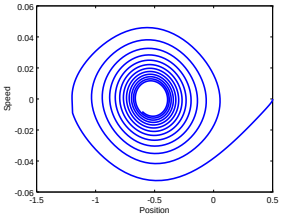
4.3 The bicycle balancing

Following [22], 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. From the initial $(0, 0, 0, 0)$ state, a random controller leads the bicycle to fall after 200 steps on average.

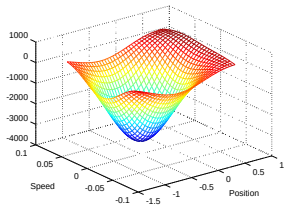
Model-based setting The MSSs are generated using a random starting state and a random controller. The definition of the policy $\pi_{\hat{U}}$ (Eq. 5) is adapted to account for the fact that, due to the temporal discretization of the transition model [22], the effect of action a_t on the angle values is only visible in state s_{t+2}. Some look-ahead is thus required to define the greedy policy $\pi_{\hat{U}}$, in order to select the action which leads to the state with the maximum pseudo value. Formally, the selected action is obtained by maximization of the value obtained after two time steps:

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

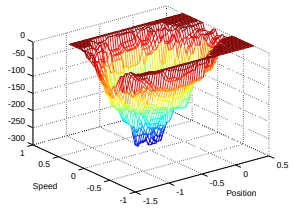
Given this definition, DIVA only requires 20 MSS of length 5 to learn a competent policy, keeping the bicycle in equilibrium for over 30,000 time steps with high probability (100 times out of 100 runs, Fig.3). In comparison, the state of the art requires



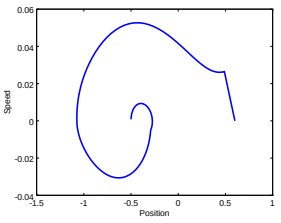
(a) MSS: Mcar falling from the target



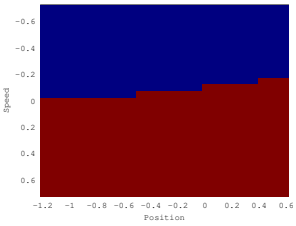
(b) XX pseudo-value



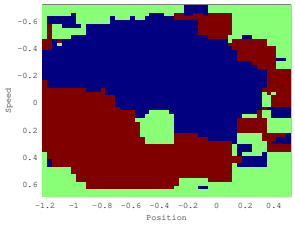
(c) SARSA value after 1000 iterations



(d) DIVA learned trajectory



(e) DIVA policy map



(f) SARSA policy map

Figure 1: The mountain car problem: Comparative evaluation of DIVA 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).

a few dozen trajectories to be ranked by the expert (15 for [3] and 20 for [32]), the starting point of which is close to the equilibrium. Under the same condition (having a starting point close to the equilibrium) DIVA reaches the goal (keeping the bicycle in equilibrium 100 times out of 100 runs) with a single MSS of length 5.

Model-free setting The ordering constraints on the state-action pairs likewise take into account the temporal discretization and the delayed impact of the actions. Formally, 2 time-steps traces are considered: from sequences $(s_1, a_1, s'_1, a'_1, s''_1)$ and $(s_2, a_2, s'_2, a'_2, s''_2)$ (with random a'_1 and a'_2), constraint $(s_1, a_1) \succ (s_2, a_2)$ is generated if

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

The proximity threshold is set to $\eta = 1\%$, as the state-action space is bigger than for the mountain car, $\mathbb{R}^4$ instead of $\mathbb{R}^2$). 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 instable one. The most ambitious goal, starting from the stable state (bottom position) is to reach the instable one (top position); a less difficult goal is to reach the horizontal line. 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 20s or when the agent successfully maintains the pendulum in an up-state ($\theta < \pi/4$) for 3 time steps.

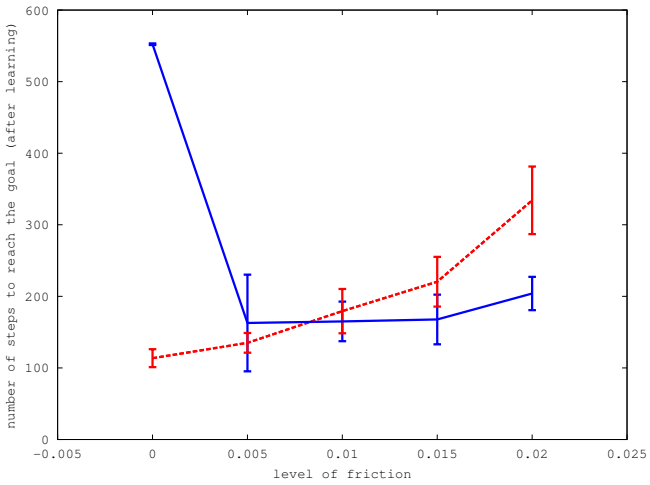


Figure 2: Mountain car: Number of time steps to reach the goal for DIVA (solid blue line) and SARSA (dashed red line) *vs* friction value (average and standard deviation over 20 runs)

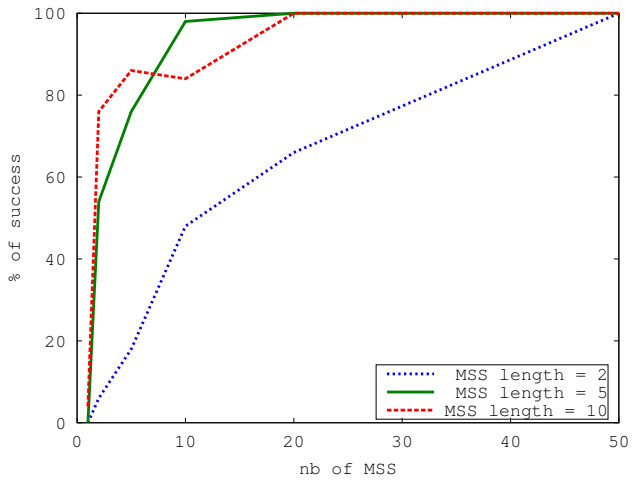


Figure 3: Bicycle (model-based setting): Fraction of success of DIVA w.r.t. number and length of MSSs

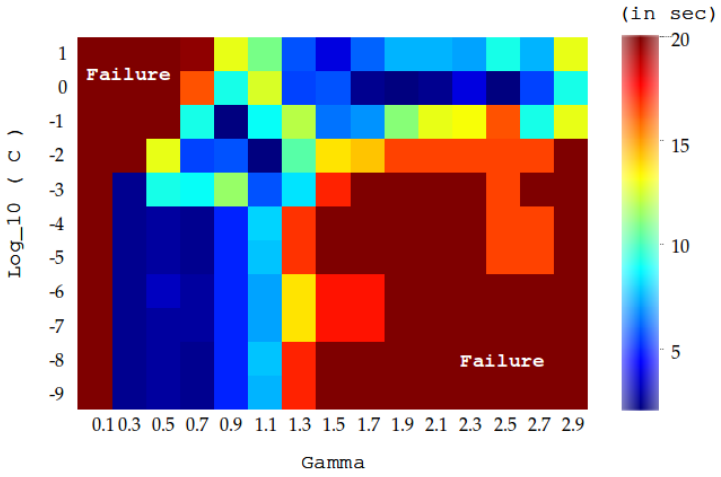


Figure 4: The swing-up pendulum problem: Sensitivity of the DIVA performance w.r.t the Ranking-SVM hyper-parameters C and $\gamma = 1/\sigma^2$ the inverse of the Gaussian kernel width. The performance is the averaged time (over 10 runs) for the policy to reach the target state and stay therein (section 4.4). For small values of C and $1/\sigma$ ($c = 10^{-5}$, $1/\sigma = 0.5$), DIVA finds a policy reaching and staying in the target state in 3 seconds. Two failure regions are observed: for low σ values, the width of the Gaussian kernel being too small, there is no generalization and the pendulum does not manage to go up. Quite the contrary when σ is too large, $\hat{U}$ underfits the goal, the pendulum goes up but fail to decrease its speed fast enough to stay in the target state.

Only the model-based setting has been considered for the pendulum problem, with a computational cost of 3 seconds.

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 $\hat{U}$ from a single MSS of length 1,000: if the kernel width is too small, there is no generalization and the pendulum does not reach the top. 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 seconds and stays there.

The DIVA 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 sec) after on average 10s. of interaction.

5 Discussion and Perspectives

The DIVA 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, and to the inverse reinforcement learning [1, 21] and the preference-based learning [19, 32, 17, 3] settings. On the one hand, DIVA directly learns a pseudo-value function, while IRL learns a reward function and preference-based learning learns a reward function [32] or a return value on the policy space [3]. Clearly, building a policy from the pseudo-value function (that is, by greedification)

is much easier than based on the reward (where it amounts to solving a full-fledge RL problem) or the return value (where it amounts to solving a difficult optimization problem).

On the other hand, and most importantly, DIVA requires much less expertise than other educated RL approaches. IRL requires the expert to be able to perform (nearly) optimal demonstrations. Preference-based RL, though requiring less expertise than IRL from the human instructor, still relies on the expert's ability to compare, and possibly repair, trajectories. In DIVA, the user is only required to know what will go wrong.

In the mountain car and the pendulum problems, DIVA uses informed MSSs (starting in the target state). In the bicycle problem however, the MSS 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 DIVA 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[4]. 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. 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 [6]) to address this limitation. A third and most important limitation concerns the non-reversible MDP case, where the transition from s to s' takes much longer than from s' to s. Further work is on-going to address the non reversible case.

A main perspective for further research is to investigate the quality of the DIVA policy in the model-free setting, depending on the structure of the MDP dynamics and the sparsity of the reward function.

References

[1] P. Abbeel. *Apprenticeship Learning and Reinforcement Learning with Application to Robotic Control*. PhD thesis, Stanford, CA, USA, 2008. AAI3332983.

[2] R. Akrour, M. Schoenauer, and M. Sebag. Preference-Based Policy Learning. In D. Gunopulos et al., editor, *Proc. Eur. Conf. on Machine Learning and Knowledge Discovery from Databases*, volume 6911 of *LNAI*, pages 12–27. Springer Verlag, 2011.

[3] R. Akrour, M. Schoenauer, M. Sebag, and J.-C. Souplet. Programming by Feedback. In *Proc. Int. Conf. on Machine Learning (ICML)*, volume 32 of *JMLR Proceedings*, pages 1503–1511. JMLR.org, 2014.

[4] P. Auer and R. Ortner. Logarithmic Online Regret Bounds for Undiscounted Reinforcement Learning. In B. Schölkopf et al., editor, *NIPS 19*, pages 49–56. MIT Press, 2007.

[4] This phenomenon is distinct from the delayed rewards of the actions: it only concerns the dynamics.

[5] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2nd edition, 2000.

[6] C. Burges, R. Ragno, and Q. Le. Learning to Rank with Non-Smooth Cost Functions. In B. Schölkopf et al., editor, *NIPS 19*, pages 193–200. MIT Press, 2006.

[7] S. Calinon and A. Billard. Active Teaching in Robot Programming by Demonstration. In *Proc. 16th IEEE RO-MAN*, pages 702–707. IEEE, 2007.

[8] M. P. Deisenroth, G. Neumann, and J. Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142, 2013.

[9] S. Filippi, O. Cappé, and A. Garivier. Optimism in Reinforcement Learning and Kullback-Leibler Divergence. In *Proc. Allerton Conf. on Communication, Control, and Computing*, pages 115 – 122, 2010.

[10] J. Fürnkranz, E. Hüllermeier, W. Cheng, and S. Park. Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Machine Learning*, 89(1-2):123–156, 2012.

[11] S. Gelly, L. Kocsis, M. Schoenauer, M. Sebag, D. Silver, C. Szepesvári, and O. Teytaud. The grand challenge of computer go: Monte carlo tree search and extensions. *Commun. ACM*, 55(3):106–113, 2012.

[12] S. Gelly and D. Silver. Combining Online and Offline Knowledge in UCT. In Z. Ghahramani, editor, *Proc. Int. Conf. on Machine Learning (ICML)*, pages 273–280. ACM, 2007.

[13] S. Gelly and D. Silver. Combining online and offline knowledge in UCT. In *Proc. Int. Conf. on Machine Learning (ICML)*, pages 273–280, 2007.

[14] A. Guez, N. Heess, D. Silver, and P. Dayan. Bayes-Adaptive Simulation-based Search with Value Function Approximation. In Z. Ghahramani et al., editor, *NIPS 27*, pages 451–459. Curran Associates, Inc., 2014.

[15] N. Heess, D. Silver, and Y. W. Teh. Actor-critic reinforcement learning with energy-based policies. In M. P. Deisenroth et al,, editor, *Eur. Wshop on Reinforcement Learning*, volume 24 of *JMLR Proceedings*, pages 43–58. JMLR.org, 2012.

[16] V. Heidrich-Meisner and C. Igel. Hoeffding and Bernstein Races for Selecting Policies in Evolutionary Direct Policy Search. In A.P. Danyluk et al., editor, *Proc. Int. Conf. on Machine Learning (ICML)*, volume 382 of *ACM Intl Conf. Proc. Series*, page 51, 2009.

[17] A. Jain, B. Wojcik, T. Joachims, and A. Saxena. Learning Trajectory Preferences for Manipulators via Iterative Improvement. In Ch. C. Burges et al., editor, *NIPS 26*, 2013.

[18] T. Joachims. A support vector method for multivariate performance measures. In L. D. Raedt and S. Wrobel, editors, *Proc. Int. Conf. on Machine Learning (ICML)*, pages 377–384. ACM, 2005.

[19] W. B. Knox, P. Stone, and C. Breazeal. Training a robot via human feedback: A case study. In G. Herrmann et al., editor, *Proc. 5th Intl Conf. on Social Robotics*, pages 460–470, 2013.

[20] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In J. F. et al., editor, *Proc. Eur. Conf. on Machine Learning and Knowledge Discovery from Databases (ECML PKDD)*, pages 282–293, 2006.

[21] G. Konidaris, S. Kuindersma, A. Barto, and R. Grupen. Constructing skill trees for reinforcement learning agents from demonstration trajectories. In J. D. Lafferty et al., editor, *NIPS 23*, pages 1162–1170. MIT Press, 2010.

[22] M. G. Lagoudakis, R. Parr, and L. Bartlett. Least-squares policy iteration. *JMLR*, 4:2003, 2003.

[23] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. A tutorial on energy-based learning. In G. Bakir, T. Hofman, B. Schölkopf, A. Smola, and B. Taskar, editors, *Predicting Structured Data*. MIT Press, 2006.

[24] P. Li, C. J. C. Burges, and Q. Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Advances in Neural Information Processing Systems 20*, pages 897–904, 2007.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[26] A. Y. Ng and S. Russell. Algorithms for Inverse Reinforcement Learning. In P. Langley, editor, *Proc. Int. Conf. on Machine Learning (ICML)*, pages 663–670. Morgan Kaufmann, 2000.

[27] S. Schaal, A. Ijspeert, and A. Billard. Computational approaches to motor learning by imitation. (1431):537–547, 2003.

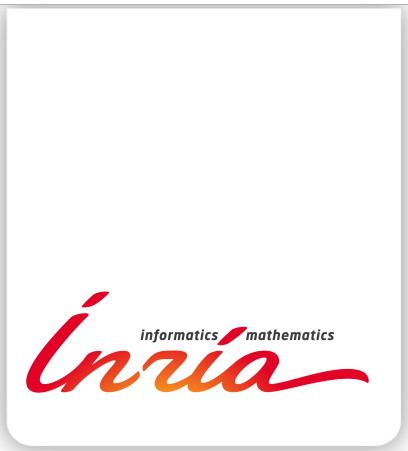
[28] R. S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In D. S. Touretzky et al., editor, *NIPS 8*, pages 1038–1044. MIT Press, 1995.

[29] R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[30] C. Szepesvári. *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2010.

[31] G. Tesauro and T. J. Sejnowski. A parallel network that learns to play backgammon. *Artif. Intell.*, 39(3):357–390, 1989.

[32] A. Wilson, A. Fern, and P. Tadepalli. A Bayesian Approach for Policy Learning from Trajectory Preference Queries. In P. L. Bartlett et al., editor, *NIPS 25*, pages 1142–1150, 2012.



RESEARCH CENTRE
SACLAY – ÎLE-DE-FRANCE

1 rue Honoré d'Estienne d'Orves
Bâtiment Alan Turing
Campus de l'École Polytechnique
91120 Palaiseau

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399