Direct Value Learning: a Rank-Invariant Approach to Reinforcement Learning

Basile Mayeur
TAO, INRIA-LRI
Univ. Paris-Sud
91405 France
basile.mayeur@inria.fr

Riad Akrour
TAO, INRIA-LRI
Univ. Paris-Sud
91405 France
riad.akrour@inria.fr

Michele Sebag
TAO, CNRS-INRIA-LRI
Univ. Paris-Sud
91405 France
michele.sebag@inria.fr

Abstract

Taking inspiration from inverse reinforcement learning, the proposed Direct Value Learning for Reinforcement Learning (DIVA) approach uses light priors to generate inappropriate behaviors, and uses the corresponding state sequences to directly learn a value function. When the transition model is known, this value function directly defines a (nearly) optimal controller. Otherwise, the value function is extended to the state-action space using off-policy learning.

The experimental validation of DIVA on the mountain car problem shows the robustness of the approach comparatively to SARSA, based on the assumption that the target state is known. The experimental validation on the bicycle problem shows that DIVA still finds good policies when relaxing this assumption.

1 Introduction

In the last decade, significant advances have been made in reinforcement learning (RL) by exploiting nearly-optimal expert demonstrations in the framework of inverse reinforcement learning [1], learning by imitation [2], or learning from demonstrations [3]. New approaches, referred to as preference-based RL and allegedly requiring less expertise from the teacher, have also been proposed [4–6].

In this paper, a new approach based on learning-to-rank and aimed at directly learning the value function, referred to as Direct Value Learning for Reinforcement Learning (DIVA), is presented, together with a theoretical and experimental analysis. After introducing the formal background and the state of the art in section 2, section 3 gives an overview of DIVA. DIVA is empirically validated on the mountain car and the bicycle balancing problems comparatively to the state of the art in section 4. The paper concludes with some perspectives for further study.

2 Position of the problem

After the standard reinforcement learning background [7], the RL goal is formalised as a Markov decision problem (MDP) $(\mathcal{S}, \mathcal{A}, p, r)$, with $\mathcal{S}$ and $\mathcal{A}$ respectively the state and action spaces, $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ the transition model, with $p(s, a, s')$ the probability of reaching state s' after selecting action a in state s, and $r : \mathcal{S} \mapsto \mathbb{R}$ the reward function[1].

Mainstream RL approaches rely on the notion of value function. Letting $\pi : \mathcal{S} \mapsto \mathcal{A}$ denote a policy, mapping each state onto an action, the value function V^π associates to each state the (expected)

[1] In the canonical setting [7] $r : (\mathcal{S}, \mathcal{A}) \mapsto \mathbb{R}$

discounted cumulative reward gathered by following π:

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

Value iteration and policy iteration algorithms [7, 8] proceed by iterating a two-step process, computing the value function associated to the current policy policy, and computing the greedy policy associated to this value function, such that

$$\pi^V(s) = \underset{a}{\text{argmax}}\{\sum_{s'} p(s, a, s') V(s')\}$$

The limitation of these approaches is twofold. On the one hand, a good reward function requires significant expertise in the domain application and in the underlying optimisation algorithm. On the other hand, the exploration of the state-action search space required to enforce the convergence of the RL process hardly scales up with the size of the state and action space.

The celebrated Inverse Reinforcement Learning (IRL) approach [3, 9] addresses both limitations, based on the exploitation of nearly optimal demonstrations provided by the expert. These demonstrations are used to infer the reward function and guide the exploration toward good regions of the state-action pairs. The limiting factor then becomes the expertise of the human teacher. A new setting, the preference-based RL approach [4–6] proceeds by asking the user in the loop to rank state-action pairs [10], fragments of behaviors [4], or full-length trajectories [5, 6].

All abovementioned approaches critically depend on the available expert knowledge. The expert knowledge is used primarily to represent the problem (state and action spaces). It is further used to specify the target behavior through the reward function, or demonstrations, or preferences. The general trend – from specifying the reward function to expressing preferences on trajectories – goes toward relaxing the expertise requirement, and correspondingly increasing the autonomy of the learner.

Along this line, the new Direct Value Learning for Reinforcement Learning RL approach, only requiring light expert priors, is presented.

3 Direct Value Learning

This section gives an overview of the DIVA algorithm.

Notation In the following, $tr(s, a)$ denotes the (random variable) state observed after executing action a in state s. Letting $f : \mathcal{S} \mapsto \mathbb{R}$, the expectation of $f(s')$ according to distribution $p(s, a, s')$ is noted $\mathbb{E}_{s' \sim s,a}[f(s')]$ and by abuse of notation $\mathbb{E}[f(tr(s, a))]$. It will be assumed in the remainder that state space $\mathcal{S}$ is a subset of $\mathbb{R}^d$, with $\overline{tr(s, a)} = \mathbb{E}_{s' \sim s,a}[tr(s, a)] \in \mathbb{R}^d$.

3.1 Principle

DIVA is based on the simple idea that, while it requires a significant expertise to demonstrate a good behavior, demonstrating a bad behavior is quite easy in many RL problems, e.g. by random action selection. A bad behavior manifests itself by visiting states with decreasing values; such a sequence of states is referred to as *Murphy State Sequence* (MSS) in honor of the Murphy's law[2].

$$MSS = (s_1, \dots s_T) \; s.t. \; V(s_t) > V(s_{t+1})$$

The core assumption of the DIVA approach is that, if the target state is known, a constant behavior starting from the target state generates a MSS; otherwise, a random behavior starting from a random state generates a MSS. A set of Murphy State Sequences defines a learning to rank problem:

$$\text{Given } MSS^{(1)}, \dots MSS^{(J)}, \text{ Find } U : \mathcal{S} \mapsto \mathbb{R} \; s.t. \; \forall j = 1 \dots J, \forall t \in 1 \dots T^{(j)} - 1, U(s_t^{(j)}) > U(s_{t+1}^{(j)}) \tag{1}$$

[2] *If something can go wrong, it will.*

Pb (1) is solved using state-of-art ranking algorithms and its solution is referred to as *utility function*. By construction, utility function U is only defined up to a monotonous transformation: for any monotonous scalar function g ($g : \mathbb{R} \mapsto \mathbb{R}, x > x' \Rightarrow g(x) > g(x')$), $g \circ U$ also is solution of (1). Two functions U and U' are said to be *rank-invariant equivalent*, noted $U \equiv_r U'$, iff there exists a monotonous scalar function g such that $U' = g \circ U$.

3.2 Model-based setting

In the case of a deterministic transition model, let the greedy policy π^U defined from utility function U be defined as $\pi^U(s) = \underset{a}{\text{argmax}}\{U(tr(s,a))$. It follows immediately that two rank-invariant equivalent utility functions define the same greedy policy:

$$U \equiv_r U' \Rightarrow \pi^{(U)} = \pi^{(U')}$$

In the case of a stochastic transition model, let $\overline{tr(s,a)}$ denote the next state expectation upon selecting action a in state s. Let the U-greedy policy π^U be defined as:

$$\pi^U(s) = \underset{a}{\text{argmax}}\{U(\overline{tr(s,a)}))\} \tag{2}$$

For U a rank-invariant equivalent of value function V, the value loss entailed by following π^U instead of π^V is bounded under mild assumptions:

Proposition 1 *Let $\mathcal{S} \subseteq \mathbb{R}^d$. Let U and V be two rank-invariant equivalent functions mapping $\mathcal{S}$ onto $\mathbf{R}$, with V differentiable and Lipschitz with constant M ($\forall s, s' \in \mathcal{S}, |V(s) - V(s')| < M||s - s'||$). Let us further assume that the transition model noise is bounded[3], with $\forall (s,a)\ ||tr(s,a) - \overline{tr(s,a)})|| < c$.*
Let $a^{*,V}(s)$ denote the best action associated to state s with respect to V,

$$a^{*,V}(s) = \mathit{argmax}_a\{\mathbb{E}[V(tr(s,a))]\}$$

If the V-margin of $a^{*,V}(s)$ w.r.t all other actions is greater than $2Mc$ ($\mathbb{E}[V(tr(s,a^{*,V}(s)))] > \mathbb{E}[V(tr(s,a'))] + 2Mc$ for all $a' \neq a^{*,V}(s)$), then

$$\pi^U(s) = a^{*,V}(s)$$

and therefore greedy policies based on V and U select same action in s.
Otherwise, the V-value loss entailed by following $\pi^U(s)$ instead of selecting $a^{*,V}(s)$ is at most $2Mc$.

Proof The proof follows from using Taylor-Lagrange decomposition on V and upper-bounding the remainder.

Algorithm In the case where the transition model is known, the DIVA algorithm learns the utility function U from problem (1), and follows the greedy policy based on U (2).

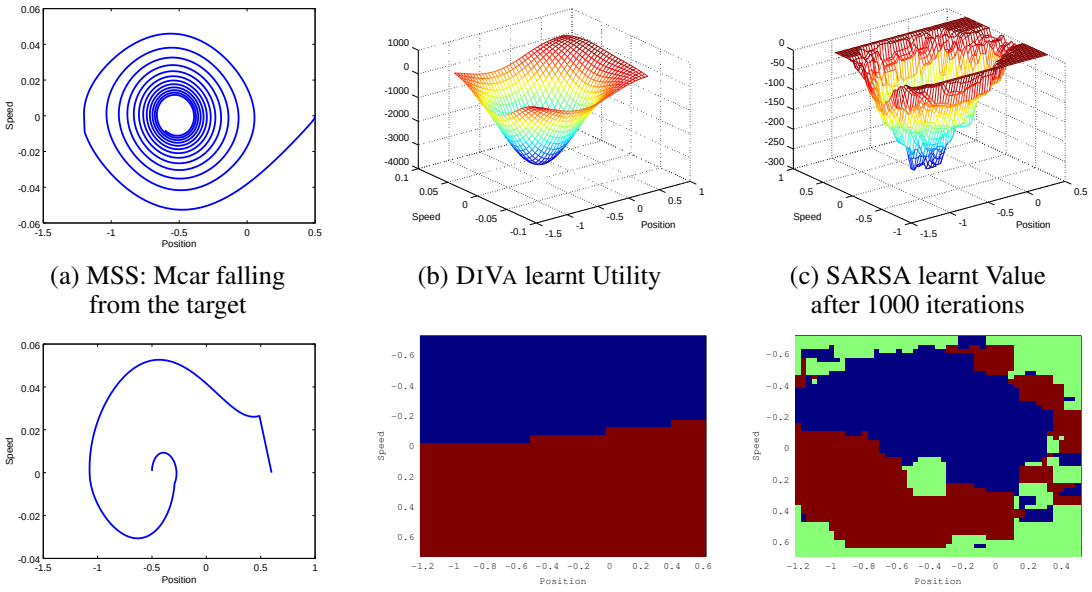
Discussion Whether utility function U is a rank-invariant equivalent of the optimal value function V^* depends on the representativity and noise of the Murphy state sequences. In practice however, what matters is whether U preserves the state ordering defined by V^* in the neighborhood of the good trajectories (more on this in section 5).

3.3 Model-free setting

When the transition model is unknown, the DIVA approach is extended to learn an approximation of the quality function $Q(s,a)$, by taking inspiration from [11]. Intuitively, given the utility function U and triplets (s_1, a_1, s'_1) and (s_2, a_2, s'_2), one requires that $(s_1, a_1) \succ (s_2, a_2)$ if $U(s'_1) \succ U(s'_2)$. More formally, model-free DIVA proceeds as follows. A set $\mathcal{E}$ of triplets (s, a, s') is generated, and ordering constraints on $\mathcal{S} \times \mathcal{A}$ are defined with:

$$\forall((s_1,a_1,s'_1);(s_2,a_2,s'_2)\ in\ \mathcal{E}\ s.t.\ U(s'_1) \succ U(s'_2), \qquad (s_1,a_1) \succ (s_2,a_2)$$

[3]While this assumption does not hold for Gaussian noise, it is realistic in e.g. robotic settings, where the distance between two consecutive states is bounded anyway due to mechanical constraints.



(a) MSS: Mcar falling from the target

(b) DIVA learnt Utility

(c) SARSA learnt Value after 1000 iterations

(d) DIVA learnt trajectory

(e) DIVA decision map

(f) SARSA decision map

Figure 1: The mountain car problem: Comparative evaluation of DIVA and SARSA in the model-free setting, on two representative runs (friction = .01). Decision maps report the best action for each state in the 2D (position, speed) space (red: forward, blue: backward, green: neutral)

A Qutility function Q_U on $\mathcal{S} \times \mathcal{A}$ is learned from the above ordering constraints, and policy $\pi^{Q,U}$ is defined as:

$$\pi^{Q,U}(s) = \underset{a}{\text{argmax}}\{Q_U(s,a)\} \tag{3}$$

It is straightforward to show that if Q and Q' are rank-invariant equivalent on $\mathcal{S} \times \mathcal{A}$, then they induce the same greedy policy:

Proposition 2 ***If*** $\forall s, \forall (a_i, a_j), (U(\overline{tr(s,a_i)}) > U(\overline{tr(s,a_j)}) \Rightarrow (Q_U(s,a_i) > Q_U(s,a_j))$

then $\underset{a}{argmax}\{Q_U(s,a)\} = \underset{a}{argmax}\{U(\overline{tr(s,a)})\}$

Some care must be exercised to prevent learning trivial quality functions. Typically if s_1 and s_2 have very different utility ($U(s_1) \gg U(s_2)$) then the learned quality function Q_U hardly reflects the action impact. In practice, pairs of triplets (s_1, a_1, s_1') and (s_2, a_2, s_2') are uniformly sampled subject to $||s_1 - s_2|| < \kappa$ with κ a fraction of the state space diameter (typically 5 to 10%), and ordering constraint $(s_1, a_1) \succ (s_2, a_2)$ is generated if $U(s_1) - U(s_1') > U(s_2) - U(s_2')$.

4 Experimental Validation

This section reports on the empirical validation of DIVA comparatively to SARSA [12]. Utility and Qutility functions are computed using RankingSVM with Gaussian kernel [13] .

4.1 The mountain car problem

The mountain car problem is defined with same setting as in [12], using a 2D state space (position, speed), and a discrete action space ({backward, neutral position, forward}). The friction coefficient ranges from 0 to .02.

In each run, a single 1,000 time step Murphy state sequence is generated by starting from the target state on the top of the hill and selecting the constant action "neutral" (Fig1(a)), yielding a very

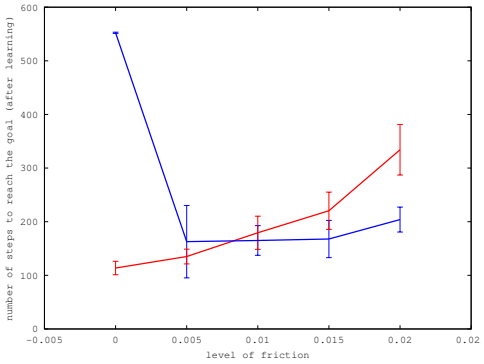


Figure 2: Mountain car: Number of time steps to reach the goal for DIVA and SARSA *vs* friction value (av. on 20 runs)

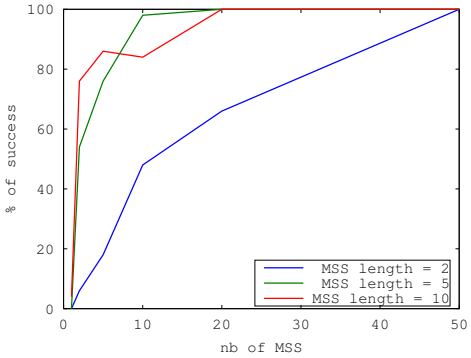


Figure 3: Bicycle (model-based setting): Fraction of success of DIVA w.r.t. number and length of MSS

smooth utility function compared to SARSA value function [12] (Figs. 1(b) and (c) respectively display the utility and value function on two representative runs). In the model-free setting, 500 pairs of triplets are considered and state-action ordering constraints are generated ($\kappa = 10\%$). Likewise, the Qutility function is much smoother than SARSA's (Fig.1(e) and (f)). Fig.2 displays the comparative performances of DIVA and SARSA depending on the friction range (averaged on 20 runs). For low friction value, time is reversible: letting the car fall down from the target state does not generate a sequence of states with decreasing value and DIVA is dominated by SARSA. For high friction value ($> .02$), 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.

4.2 The Bicycle problem

With same setting as [14], the state space of the bicycle riding problem is $\mathbb{R}^4$ (the angles of the handlebar and of the bicycle to the vertical; associated angular velocities); the action space is discrete (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 will make the bicycle fall after approximately 200 steps.

Model-based setting Murphy state sequences are generated by following a random controller starting from a random state. Due to the temporal discretisation of the transition model [14], action a_t affects the angle values only in step $t + 2$. Some look-ahead is thus required: policy π^U selects the action that maximises the maximum of the Utility at state s_{t+2}:

$$\pi^U(s) = \hat{a}^* = \operatorname*{argmax}_a \left(\max_{a'} U(\overline{tr(\overline{tr(s,a)}, a')})\right)$$

Fig. 3 shows that 20 MSS of length 5 generated by a random controller yield a competent utility function (keeping the bicycle in equilibrium for over 30,000 steps) with high probability (100 out of 100 runs).

Model-free setting The ordering constraints on the state-action pairs must also account for the temporal discretisation. Formally, considering 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 $U(s''_1) - U(s_1) > U(s''_2) - U(s_2)$.

5,000 pairs are required to achieve same performances as in the model-based setting (with proximity threshold $\kappa = 1\%$, as the state-action space is bigger than for the mountain car, $\mathbb{R}^4$ instead of $\mathbb{R}^2$).

5 Discussion and Perspectives

The main contribution of this paper is i) to suggest that utility functions, rank-invariant equivalent of the (nearly) optimal value functions, can support satisfactory policies; ii) to show that such utility functions can be learned from bad behaviors. It is believed that this approach makes a step toward reinforcement learning with light prior knowledge, as generating bad behaviors usually requires much less expertise than good behaviors.

A first limitation of the DIVA approach is some look-ahead can be required to compute the utility-based policy, depending on the latency of the transition model. A second limitation is that, in some problems e.g. the bicycle driving task, it might be hard to generate bad behaviors. A perspective to alleviate this limitation is to consider bad behaviors with additional constraints.

Further work will consider richer state spaces (e.g. robot controllers with some dozen degrees of freedom), and investigate the DIVA scalability. Other ranking approaches with linear complexity will be considered [15]. Finally, DIVA will be combined with approximate RL methods by taking inspiration from [16].

References

[1] Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *Proc. 17th International Conf. on Machine Learning*, pages 663–670. Morgan Kaufmann, 2000.

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

[3] George Konidaris, Scott Kuindersma, Andrew Barto, and Roderic Grupen. Constructing skill trees for reinforcement learning agents from demonstration trajectories. In *Advances in Neural Information Processing Systems*. MIT Press, 2010.

[4] Aaron Wilson, Alan Fern, and Prasad Tadepalli. A bayesian approach for policy learning from trajectory preference queries. In *Advances in Neural Information Processing Systems*, 2012.

[5] Riad Akrour, Marc Schoenauer, and Michèle Sebag. Preference-Based Policy Learning. In *Eur Conf on Machine Learning*, volume 6911 of *LNAI*. Springer Verlag, 2011.

[6] A. Jain, B. Wojcik, T. Joachims, and A. Saxena. Learning trajectory preferences for manipulators via iterative improvement. In *Neural Information Processing Systems*, December 2013.

[7] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

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

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

[10] W. Bradley Knox, Peter Stone, and Cynthia Breazeal. Training a robot via human feedback: A case study. In *Social Robotics*, October 2013.

[11] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *J. Mach. Learn. Res.*, 6:503–556, December 2005.

[12] Richard S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems*. MIT Press, 1996.

[13] Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 217–226, New York, NY, USA, 2006. ACM.

[14] Jette Randlov and Preben Alstrom. Learning to drive a bicycle using reinforcement learning and shaping, 1998.

[15] C.J.C. Burges, R. Ragno, and Q.V. Le. Learning to rank with non-smooth cost functions. In *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA, 2007.

[16] Beomjoon Kim, Amir Massoud Farahmand, Joelle Pineau, and Doina Precup. Learning from limited demonstrations. In Christopher J. C. Burges, Lon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *NIPS*, pages 2859–2867, 2013.