Boosted Bellman Residual Minimization Handling Expert Demonstrations

Bilal Piot[1,2], Matthieu Geist[1,2], and Olivier Pietquin[3]

[1] Supélec, IMS-MaLIS Research group, France
{bilal.piot,matthieu.geist}@supelec.fr
[2] UMI 2958 (GeorgiaTech-CNRS), France
[3] University Lille 1, LIFL (UMR 8022 CNRS/Lille 1) - SequeL team, Lille, France
olivier.pietquin@univ-lille1.fr

Abstract. This paper addresses the problem of batch Reinforcement Learning with Expert Demonstrations (RLED). In RLED, the goal is to find an optimal policy of a Markov Decision Process (MDP), using a data set of fixed sampled transitions of the MDP as well as a data set of fixed expert demonstrations. This is slightly different from the batch Reinforcement Learning (RL) framework where only fixed sampled transitions of the MDP are available. Thus, the aim of this article is to propose algorithms that leverage those expert data. The idea proposed here differs from the Approximate Dynamic Programming methods in the sense that we minimize the Optimal Bellman Residual (OBR), where the minimization is guided by constraints defined by the expert demonstrations. This choice is motivated by the the fact that controlling the OBR implies controlling the distance between the estimated and optimal quality functions. However, this method presents some difficulties as the criterion to minimize is non-convex, non-differentiable and biased. Those difficulties are overcome via the embedding of distributions in a Reproducing Kernel Hilbert Space (RKHS) and a boosting technique which allows obtaining non-parametric algorithms. Finally, our algorithms are compared to the only state of the art algorithm, Approximate Policy Iteration with Demonstrations (APID) algorithm, in different experimental settings.

1 Introduction

This paper addresses the problem of batch Reinforcement Learning with Expert Demonstrations (RLED) where the aim is to find an optimal policy of a Markov Decision Process (MDP) only known through fixed sampled transitions, when expert demonstrations are also provided. Thus, RLED can be seen as a combination of two classical frameworks, Learning from Demonstrations (LfD) and batch Reinforcement Learning (RL). The LfD framework is a practical paradigm for learning from expert trajectories. A classical approach to LfD is to generalize the mapping between states and actions observed in the expert data. This can be done by a Supervised Learning method such as a multi-class classification algorithm [19]. However, those methods do not generalize well to regions of the state space that are not observed in the expert data, because they do not take

T. Calders et al. (Eds.): ECML PKDD 2014, Part II, LNCS 8725, pp. 549–564, 2014.

into account the underlying dynamics of the MDP. To alleviate this, some recent methods [11,21] focus on querying the expert in some appropriate regions of the state space to improve the learning. However, this implies that the expert stays with the learning agent throughout the training process, which can reduce significantly the applicability of the technique. Therefore, the idea presented in [12] to overcome the limitation of conventional LfD methods is to use techniques from the batch Reinforcement Learning (RL) paradigm and combine them with LfD techniques. In batch Reinforcement Learning (RL), the goal is the same as in RLED, but without expert data. Usual techniques of batch RL have some difficulties to achieve good results from relatively little data. However, if some expert demonstrations are added to the set of sampled transitions, it is possible to improve significantly the results of the method [12]. Thus, a combination of expert data and non expert data offers the possibility to address the problem of learning optimal policies under realistic assumptions.

At our knowledge, there is only one algorithm taking into account expert data in order to find an optimal policy. This approach is Approximate Policy Iteration with Demonstrations (APID) [12], which consists in using expert demonstrations to define linear constraints that guide the optimization performed by Approximate Policy Iteration (API), a classical framework in RL. The practical algorithm APID is inspired by Least Squares Temporal Differences (LSTD) [6], where the choice of features is a key problem as it is a parametric method. Even if the optimization could be done in a Reproducing Kernel Hilbert Space (RKHS), which provides the flexibility of working with a non-parametric representation as claimed by [12], the choice of the kernel can be as difficult as the choice of the features. Therefore, we propose a method with no features choice in order to solve the RLED problem. Our method consists in the minimization of the norm of the Optimal Bellman Residual (OBR), guided by constraints defined by the expert demonstrations (see Sec 2.1). This minimization is motivated by the fact that if one is able to find a function with a small OBR, then this function is close to the optimal quality function. However, as far as we know, this technique is not used in RL for three reasons. First, the empirical norm of the OBR is biased. Second, it is not convex, so the minimization could lead to local minima. Third, it is not differentiable. Our contribution is to show how we can construct an empirical norm of the OBR which is not biased via the embedding of distributions in an RKHS (see Sec. 4.1) and how it is possible to minimize this non-convex and non-differentiable criterion via a boosting technique (see Sec. 4.2). In addition, boosting techniques are non-parametric methods which avoid choosing features.

In the proposed experiments (see Sec. 5), we compare our algorithms to the only state of the art algorithm APID, to an RL algorithm Least Square Policy Iteration (LSPI) [14], and to an LfD algorithm [19]. The first experiment is conducted on a generic task (randomly generated MDP called Garnet [3]) where expert demonstrations and non-expert transitions are provided and the aim is to find an optimal policy. The second experiment is realized on a classical LfD benchmark, the Highway problem.

2 Background and Notations

Let $(\mathbb{R}, |.|)$ be the real space with its canonical norm and X a finite set, $\mathbb{R}^X$ is the set of functions from X to $\mathbb{R}$. The set of probability distributions over X is noted Δ_X. Let Y be a finite set, Δ_X^Y is the set of functions from Y to Δ_X. Let $\alpha \in \mathbb{R}^X$, $p \in \mathbb{R}_+^*$ and $\nu \in \Delta_X$, we define the $\mathbf{L}_{p,\nu}$-norm of α, noted $\|\alpha\|_{p,\nu}$, by: $\|\alpha\|_{p,\nu} = (\sum_{x\in X} \nu(x)|\alpha(x)|^p)^{\frac{1}{p}}$. In addition, the infinite norm is noted $\|\alpha\|_\infty$ and defined as $\|\alpha\|_\infty = \max_{x\in X} |\alpha(x)|$. Let $x \in X$, $x \sim \nu$ means that x is sampled according to ν and $\mathbb{E}_\nu[\alpha] = \sum_{x\in X} \nu(x)\alpha(x)$ is the expectation of α under ν. Finally, $\delta_x \in \mathbb{R}^X$ is the function such that $\forall y \in X$, if $y \neq x$ then $\delta_x(y) = 0$, else $\delta_x(y) = 1$.

2.1 MDP, RL and RLED

In this section, we provide a very brief summary of some of the concepts and definitions from the theory of MDP and RL. For further information about MDP, the reader can be referred to [18]. Here, the agent is supposed to act in a finite MDP [1]. An MDP models the interactions of an agent evolving in a dynamic environment and is represented by a tuple $M = \{S, A, R, P, \gamma\}$ where $S = \{s_i\}_{1\leq i\leq N_S}$ is the state space, $A = \{a_i\}_{1\leq i\leq N_A}$ is the action space, $R \in \mathbb{R}^{S\times A}$ is the reward function, $\gamma \in]0, 1[$ is a discount factor and $P \in \Delta_S^{S\times A}$ is the Markovian dynamics which gives the probability, $P(s'|s, a)$, to reach s' by choosing the action a in the state s. A policy π is an element of A^S and defines the behavior of an agent. In order to quantify the quality of a policy π relatively to the reward R, we define the quality function. For a given MDP M and a given policy π, the quality function $Q^\pi \in \mathbb{R}^{S\times A}$ is defined as $Q^\pi(s, a) = \mathbb{E}_{s,a}^\pi[\sum_{t=0}^{+\infty} \gamma^t R(s_t, a_t)]$, where $\mathbb{E}_{s,a}^\pi$ is the expectation over the distribution of the admissible trajectories $(s_0, a_0, s_1, \pi(s_1), \dots)$ obtained by executing the policy π starting from $s_0 = s$ and $a_0 = a$. Moreover, the function $Q^* \in \mathbb{R}^{S\times A}$ defined as: $Q^* = \max_{\pi\in A^S} Q^\pi$ is called the optimal quality function. A policy π which has the following property: $\forall s \in S, \pi(s) \in \operatorname{argmax}_{a\in A} Q^*(s, a)$ is said optimal with respect to R. Thus, it is quite easy to construct an optimal policy via the knowledge of the optimal quality function. For ease of writing, for each Q and each π, we define $f_Q^* \in \mathbb{R}^S$ such that $\forall s \in S, f_Q^*(s) = \max_{a\in A} Q(s, a)$ and $f_Q^\pi \in \mathbb{R}^S$ such that $\forall s \in S, f_Q^\pi(s) = Q(s, \pi(s))$. Q^π and Q^* are fixed points of the two following contracting operators T^π and T^* for the infinite norm:

$$\forall Q \in \mathbb{R}^{S\times A}, \forall (s, a) \in S \times A, \quad T^\pi Q(s, a) = R(s, a) + \gamma \mathbb{E}_{P(.|s,a)}[f_Q^\pi],$$
$$\forall Q \in \mathbb{R}^{S\times A}, \forall (s, a) \in S \times A, \quad T^* Q(s, a) = R(s, a) + \gamma \mathbb{E}_{P(.|s,a)}[f_Q^*].$$

The aim of Dynamic Programming (DP) is, given an MDP M, to find Q^* which is equivalent to minimizing a certain norm of the OBR defined as $T^*Q - Q$:

$$J_{DP}(Q) = \|T^*Q - Q\|,$$

[1] This work could be easily extended to measurable state spaces as in [9,12]; we choose the finite case for the ease and clarity of exposition.

where $\|.\|$ is a norm which can be equal to $\|.\|_\infty$ or $\|.\|_{\nu,p}$, $\nu \in \Delta_{S\times A}$ is such that $\forall (s,a) \in S \times A, \nu(s,a) > 0$ and $p \in \mathbb{R}_+^*$. Usual techniques of DP, such as Value Iteration (VI) or Policy Iteration (PI), do not directly minimize the criterion $J_{DP}(Q)$ but uses particular properties of the operators T^π and T^* to obtain Q^*. However, the motivation to minimize the norm of the OBR is clear as:

$$\|Q^* - Q\| \leq \frac{C}{1-\gamma}\|T^*Q - Q\|,$$

where $C \in \mathbb{R}^*$ is a constant depending notably on the MDP (for details see the work of [16]). This means that if we are able to control the norm of the OBR, then we have found a function Q close to Q^*, which is the goal of DP.

Batch RL aims at estimating Q^* or finding an optimal policy when the model (the dynamics P and the reward function R) of the MDP M is known only through the RL data set noted D_{RL} which contains N_{RL} sampled transitions of the type (s_i, a_i, r_i, s'_i) where $s_i \in S$, $a_i \in A$, $r_i = R(s_i, a_i)$ and $s'_i \sim P(.|s_i, a_i)$: $D_{RL} = (s_i, a_i, r_i, s'_i)_{1\leq i\leq N_{RL}}$. For the moment no assumption is made on the distribution $\nu_{RL} \in \Delta_{S\times A}$ from which the data are drawn, $(s_i, a_i) \sim \nu_{RL}$. In batch RLED, we suppose that we also have the expert data set D_E which contains N_E expert state-action couples of the the type $(s_j, \pi_E(s_j))$ where $s_j \in S$ and π_E is an expert policy which can be considered optimal or near-optimal: $D_E = (s_j, a_j = \pi_E(s_j))_{1\leq j\leq N_E}$. The distribution from which the expert data are drawn is noted $\nu_E \in \Delta_S$, $s_j \sim \nu_E$.

3 A New Algorithm for the RLED Problem

Our problem consists in approximating Q^*. We saw in Sec. 2.1 that minimizing a certain norm of the OBR can lead us to a good approximation of the optimal quality function and of the optimal policy. However, the only available knowledge of the MDP lies in the sets of data D_{RL} and D_E. Thus for the set D_{RL}, we want to find a function $Q \in \mathbb{R}^{S\times A}$ that minimizes the empirical OBR:

$$J_{RL}(Q) = \frac{1}{N_{RL}}\sum_{i=1}^{N_{RL}} |T^*Q(s_i,a_i) - Q(s_i,a_i)|^p \underset{\text{def}}{=} \|T^*Q - Q\|^p_{D_{RL},p},$$

where $p \geq 1$. For the expert set D_E, we would like to express that the action a_j is optimal, which means that $Q(s_j, a_j)$ is greater than $Q(s_j, a)$ where $a \in A\backslash a_j$. This can be expressed by the following large margin constraints:

$$\forall 1 \leq j \leq N_E, \quad \max_{a\in A}[Q(s_j,a) + l(s_j,a_j,a)] - Q(s_j,a_j) \leq 0,$$

where $l \in \mathbb{R}_+^{S\times A\times A}$ is a margin function such that $\forall 1 \leq j \leq N_E$, $\forall a \in A\backslash a_j$, $l(s_j, a_j, a) > l(s_j, a_j, a_j)$. A canonical choice of margin function is $\forall 1 \leq j \leq N_E$, $\forall a \in A\backslash a_j$, $l(s_j, a_j, a_j) = 0, l(s_j, a_j, a) = 1$. This imposes that the function Q we are looking for is greater by a given amount determined by the margin function for the expert actions. If available, one prior knowledge can be used

to structure the margin. Those constraints guide the minimization of J_{RL} as they impose a particular structure for the quality function we are looking for. Here, it is important to note that the constraints that guide the minimization of J_{RL} are compatible with this minimization as they are satisfied by the expert policy which is near optimal. Thus, we can think that those constraints help to accelerate and improve the minimization of J_{RL}.

However, notice that it is not the case if we use T^{π} in lieu of T^{*}, because the policy π can be completely different from the expert policy. This is a problem encountered in the APID algorithm, as they choose a Policy Iteration framework where there are several steps of the minimization of the norm of the Bellman Residual $\|T^{\pi}Q - Q\|$ guided by constraints on expert data (see Sec. 3.1).

As the expert policy might be suboptimal, the constraints can be violated by an optimal policy, that is why we smooth those constraints with slack variables:

$$\forall 1 \leq j \leq N_E, \quad \max_{a \in A}[Q(s_j, a) + l(s_j, a_j, a)] - Q(s_j, a_j) \leq \xi_j,$$

where ξ_j is a positive slack variable that must be the smallest possible. So the idea is to minimize:

$$J_{RL}(Q) + \frac{\lambda}{N_E} \sum_{j=1}^{N_E} \xi_j,$$
$$\text{subject to } \max_{a \in A}[Q(s_j, a) + l(s_j, a_j, a)] - Q(s_j, a_j) \leq \xi_j, \quad \forall 1 \leq j \leq N_E.$$

where λ determines the importance between the RL data and the expert data.

Following [20], as the slack variables are tight and positive, this problem is equivalent to minimizing:

$$J_{RLE}(Q) = J_{RL}(Q) + \lambda J_E(Q),$$

where: $J_E(Q) = \frac{1}{N_E} \sum_{j=1}^{N_E} \max_{a \in A}[Q(s_j, a) + l(s_j, a_j, a)] - Q(s_j, a_j)$. The minimization of the criterion $J_E(Q)$ is known in the literature and used in the LfD paradigm [19,13,17]. The minimization of $J_E(Q)$ can be seen as a score-based multi-class classification algorithm where the states s_j play the role of inputs and the actions a_j play the role of labels.

3.1 Comparaison to the APID Method

The APID method is couched in the API framework [5], which starts with an initial policy π_0. At the $k+1$-th iteration, given a policy π_k, the quality function Q^{π_k} is approximately evaluated by $\hat{Q}_k$. This step is called the approximate policy evaluation step. Then, a new policy π_{k+1} is computed, which is greedy with respect to $\hat{Q}_k$. In the APID algorithm, the policy evaluation step is realized by the following unconstrained minimization problem:

$$\hat{Q}_k = \underset{Q \in \mathbb{R}^{S \times A}}{\operatorname{argmin}} J^{\pi}_{RL}(Q) + \lambda J_E(Q), \tag{1}$$

where: $J^{\pi}_{RL}(Q) = \frac{1}{N_{RL}} \sum_{i=1}^{N_{RL}} |T^{\pi}Q(s_i,a_i) - Q(s_i,a_i)|^p \underset{\text{def}}{=} \|T^{\pi}Q - Q\|^p_{D_{RL},p}$. The difference between $J_{RL}(Q)$ and $J^{\pi}_{RL}(Q)$ is the use of the operator T^* in lieu of T^{π}. In the APID method, the policy evaluation step is slightly different from a classical API method which consists in the minimization of $J^{\pi}_{RL}(Q)$. This introduces an error in the evaluation step for the first iterations of the APID method, which can potentially slow down the learning process. Moreover, as the APID is an iterative method, the unconstrained problem Eq. (1) has to be resolved several times which can be time expensive. However, this problem is convex and easier to resolve than the minimization of J_{RLE}. In practice, in order to resolve the problem in Eq. (1), an LSTD-like algorithm [12] is used (this is the algorithm implemented in our experiments in order to represent the APID method). This method is by nature parametric and needs the choice of features or the choice of a kernel. Thus, the APID has as advantage the simplicity of minimizing $J^{\pi}_{RL}(Q) + \lambda J_E(Q)$, which is convex, but has also some drawbacks as it is an iterative and parametric method.

Our algorithm, which consists in the minimization of $J_{RLE}(Q) = J_{RL}(Q) + \lambda J_E(Q)$, avoids the APID drawbacks as it can be in practice a non parametric and non iterative method (see Sec. 4) but it is a non-convex criterion. Indeed, let us take a closer look at $J_{RL}(Q) = \|T^*Q - Q\|^p_{D_{RL},p}$. This criterion is an empirical norm of the OBR, and the minimization of this criterion for solving the batch RL problem is an unused technique at our knowledge. There are several reasons to understand why the OBR minimization (OBRM) is usually not used. The first one is that this criterion is not convex in Q, so a direct minimization could lead us to local minima. The second reason is that this criterion is not differentiable because of the max operator, so we need to use sub-gradient techniques or generalized gradient techniques to minimize it. The third reason is that this technique is not directly inspired by a dynamic programming approach such as PI or VI.

In the next section, we exhibit the bias problem involved by J_{RL}. However, it is possible to construct two criterions $\hat{J}_{RL}$, which is a biased criterion but can be used in some specific conditions, and $\overline{J}_{RL}$ which converges in probability to $\|T^*Q - Q\|^2_{\nu,2}$ when N_{RL} tends to infinity and ν is the distribution from which the data are drawn. $\overline{J}_{RL}$ is obtained thanks to the use of a distribution embedding in an RKHS. Finally, we show how we overcome the non-convex and non-differentiability difficulties via a boosting technique which allows obtaining non-parametric algorithms.

4 Practical Minimization of J_{RLE}

In this section, we present how the criterion J_{RLE} is minimized in order to obtain a practical and non-parametric algorithm. Here, we choose $p = 2$ and we suppose that the data (s_i, a_i) are drawn identically and independently (i.i.d) from a distribution $\nu \in \Delta_{S\times A}$ such that $\forall (s,a), \nu(s,a) > 0$. The i.i.d assumption is only done here to simplify the analysis. Indeed, this assumption could be relaxed

using techniques that handle dependent data [25]. The i.i.d assumption for the data set D_{RL} in a batch RL setting is common in the literature [14,15,12].

First, let us take a closer look to the term $J_{RL}(Q) = \|T^*Q - Q\|^2_{D_{RL},2}$. If we only have the knowledge of the data set D_{RL}, we cannot compute $T^*Q(s_i, a_i)$, but we can compute an unbiased estimate $\hat{T}^*Q(s_i, a_i) = R(s_i, a_i) + \gamma \max_{a \in A} Q(s'_i, a)$. Then, our criterion becomes:

$$\hat{J}_{RL}(Q) = \frac{1}{N_{RL}} \sum_{i=1}^{N_{RL}} |\hat{T}^*Q(s_i, a_i) - Q(s_i, a_i)|^2 \underset{\text{def}}{=} \|\hat{T}^*Q - Q\|^2_{D_{RL},2}.$$

Unfortunately, this is a biased criterion. However, we have the following result:

Theorem 1.

$$\hat{J}_{RL}(Q) \underset{N_{RL} \to \infty}{\to} \|T^*Q - Q\|^2_{\nu,2} + \gamma^2 \sum_{(s,a) \in S \times A} \nu(s,a) \mathbb{E}_{P(.|s,a)}[(f^*_Q)^2 - \mathbb{E}_{P(.|s,a)}[|f^*_Q|]^2].$$

Proof. The proof follows the same line as the one of [2], where T^* replaces T^π.

So, when the number of samples tends to infinity, the criterion $\hat{J}_{RL}(Q)$ tends to $\|T^*Q - Q\|^2_{\nu,2}$, which is what we want to minimize plus a term of variance $\gamma^2 \sum_{(s,a) \in S \times A} \nu(s,a) \mathbb{E}_{P(.|s,a)}[(f^*_Q)^2 - \mathbb{E}_{P(.|s,a)}[|f^*_Q|]^2]$. This term will favor functions which are smooth, but it is not controlled by a factor that we can choose such as in regularization theory. As pointed by [2], we have the same problem in the minimization of the Bellman residual in the PI framework. It is not satisfactory to present a criterion which has a bias even if it can work in some specific conditions such as when the MDP is deterministic (in that case $\hat{J}_{RL}(Q) \underset{N_{RL} \to \infty}{\to} \|T^*Q - Q\|^2_{\nu,2}$) or when the optimal action value function we are looking for is really smooth.

Thus, we also propose a criterion which does not have this bias. Several techniques have been used to get rid off the bias in the minimization of the Bellman Residual (see [2]), but here we are going to use the work developed by [15] where a conditional distribution is embbeded in a Reproducing Kernel Hilbert Space (RKHS), more appropriate for the considered batch setting.

4.1 RKHS Embbedings for MDP

Let us start with some notations relative to RKHS [4]. Let K be a positive definite kernel on a finite set X. The unique Hilbert space H with reproducing kernel K is denoted by H_K. Correspondingly the norm will be denoted by $\|.\|_K$ and the inner product will be denoted by $\langle ., . \rangle_K$.

Now, we can use the notion of distribution embeddings [23,15]. Given any probability distribution $P \in \Delta_X$ and a positive definite kernel K on X, a distribution embbeding of P in H_K is an element $\nu \in H_K$ such that:

$$\forall h \in H_K, \langle \nu, h \rangle_K = \mathbb{E}_P[h].$$

In our application, we want to find a distribution embbeding for the conditional distribution $P(.|s,a)$. Following the work done by [15], given the data set D_{RL}, a positive definite kernel K on $S\times A$, a positive definite kernel L on S, there is a way to estimate the element $\nu_{s,a} \in H_L$ such that $\langle \nu_{s,a}, f\rangle_K = \mathbb{E}_{P(.|s,a)}[f]$, for all $f \in H_L$. The estimation of $\nu_{s,a}$ is noted $\overline{\nu}_{s,a}$ and is such that:

$$\overline{\nu}_{s,a} = \sum_{i=1}^{N_{RL}} \overline{\beta}_i(s,a) L(s'_i, .) \in H_L,$$

where $\overline{\beta}_i(s,a) = \sum_{j=1}^{N_{RL}} W_{ij} K((s_j,a_j),(s,a))$, and where $\mathbf{W} = (W_{ij})_{1\leq i,j\leq N_{RL}} = (\mathbf{K} + \lambda_K N_{RL}\mathbf{I})^{-1}$ with $\mathbf{K} = (K((s_i,a_i),(s_j,a_j)))_{1\leq i,j\leq N_{RL}}$, $\mathbf{I}$ the identity matrix of size N_{RL} and $\lambda_K \in \mathbb{R}_+$. In the case where S is finite, we can choose L to be the canonical dot product and in that case, we have $H_L = \mathbb{R}^S$ and $\forall Q \in \mathbb{S}\times\mathbb{A}, f^*_Q \in H_L$. However if S is a measurable state space, the choice of L is not canonical.

Thus, if $f^*_Q \in H_L$ (which is the case when S is finite and L is the euclidian dot product) and if we define $\overline{T}^*Q(s_i,a_i) = R(s_i,a_i) + \gamma \sum_{j=1}^{N_{RL}} \overline{\beta}_j(s_i,a_i) \max_{a\in A} Q(s'_j,a)$, we have that: $\overline{T}^*Q(s_i,a_i) = R(s_i,a_i) + \langle \overline{\nu}_{s_i,a_i}, f^*_Q\rangle$. So, if we define the following criterion:

$$\overline{J}_{RL}(Q) = \frac{1}{N_{RL}} \sum_{i=1}^{N_{RL}} |\overline{T}^*Q(s_i,a_i) - Q(s_i,a_i)|^2 \underset{\text{def}}{=} \|\overline{T}^*Q - Q\|^2_{D_{RL},2},$$

we have the following Theorem:

Theorem 2. *Under some smoothness conditions of the MDP described in [15], the strict positivity of the Kernel L and by choosing $\lambda_K \underset{N_{RL}\to\infty}{\to} 0$ and $\lambda_K N^3_{RL} \underset{N_{RL}\to\infty}{\to} \infty$, we have if $\|f^*_Q\|_L < \infty$:*

$$\overline{J}_{RL}(Q) \underset{N_{RL}\to\infty}{\overset{\nu}{\to}} \|T^*Q - Q\|^2_{\nu,2}.$$

Here, the convergence is in ν-Probability.

Proof. This comes directly from the Cauchy-Schwartz inequality and the Lemma 2.1 in [15].

$$\sup_{(s,a)\in S\times A} \|\nu_{s,a} - \overline{\nu}_{s,a}\|_L \underset{N_{RL}\to\infty}{\overset{\nu}{\to}} 0.$$

It is important to remark that we only need the coefficients of the form $(\overline{\beta}_j(s_i,a_i))_{1\leq i,j\leq N_{RL}}$ in order to construct $\overline{J}_{RL}(Q)$. Thus we only need to compute the matrix product $\mathbf{B} = (B_{ij})_{1\leq i,j\leq N_{RL}} = \mathbf{WK}$ because $B_{ij} = \sum_{k=1}^{N_{RL}} W_{ik} K((s_k,a_k),(s_j,a_j)) = \overline{\beta}_i(s_j,a_j)$. Finally, we can easily construct two criterions from the data set D_{RL}. One criterion, $\hat{J}_{RL}(Q)$, is naturally biased and the other one, $\overline{J}_{RL}(Q)$, converges in probability towards $\|T^*Q - Q\|^2_{\nu,2}$, with certain conditions of smoothness of the MDP. Those two criterions can take the same form if we rewrite

$\hat{T}^*Q(s_i,a_i) = R(s_i,a_i) + \gamma \sum_{j=1}^{N_{RL}} \hat{\beta}_j(s_i,a_i) \max_{a \in A} Q(s'_j,a)$ with $\hat{\beta}(s_i,a_i)_j = 1$ if $j = i$ and $\hat{\beta}_j(s_i,a_i) = 0$ otherwise. Thus the practical algorithms will consists in minimizing the two following criterions:

$$\hat{J}_{RLE}(Q) = \hat{J}_{RL} + \lambda J_E(Q),$$
$$\overline{J}_{RLE}(Q) = \overline{J}_{RL} + \lambda J_E(Q).$$

In order to realize it, we use a boosting technique.

4.2 Boosting

A boosting method is an interesting optimization technique: it minimizes directly the criterion without the step of choosing features, which is one of the major drawback of several RL methods. As presented by [10], a boosting algorithm is a projected sub-gradient descent [22] of a convex functional in a specific functions space (here $\mathbb{R}^{S \times A}$) which has to be a Hilbert space. The principle is to minimize a convex functional $F \in \mathbb{R}^H$ where H is a Hilbert space: $\min_{h \in H} F(h)$. This technique can be extended to non-smooth and non-convex functionals, yet the functional has to be Lipschitz in order to guarantee that the gradient of the functional exists almost everywhere [8]. For a Lipschitz and non smooth functional, the gradient can be calculated almost everywhere and if not the notion of generalized gradient is used (see [8] for details). To realize this minimization, we need to calculate the gradient $\partial_h F \in H$ and define $K \subset H$, a set of allowable directions (also called the restriction set) where the gradient is projected. Boosting algorithms use a projection step when optimizing over function space because the functions representing the gradient are often computationally difficult to manipulate and do not generalize well to new inputs [10]. In boosting literature, the restriction set corresponds directly to the set of hypotheses generated by a weak learner. The nearest direction k^*, which is the projection of the gradient $\partial_h F$, is defined by:

$$k^* = \operatorname*{argmax}_{k \in K} \frac{\langle k, \partial_h F \rangle}{\|k\|},$$

where $\langle .,. \rangle$ is the inner product associated to the Hilbert space H and $\|.\|$ is the associated canonical norm. Then, the naive algorithm to realize the minimization of F is given by Algo. 1. More sophisticated boosting algorithms and their convergence proofs are presented by [10]. However, the naive approach is sufficient to obtain good results. For our specific problem, $H = \mathbb{R}^{S \times A}$, and $\langle .,. \rangle$ is the canonical dot product. The criterions which have to be minimized are $\overline{J}_{RLE}$ and $\hat{J}_{RLE}$. As those criterions have the same form (the only difference is the value of the coefficients $\hat{\beta}_j$ and $\overline{\beta}_j$), we present the boosting technique only for $\overline{J}_{RLE}$. Moreover, in our experiments, we choose the restriction set K to be weighted classification trees [7] from $\mathbb{R}^{S \times A}$ to $\{-1, 1\}$ where each k has the same norm (as it takes its values in $\{-1, 1\}$). Thus, our algorithm is given by Algo. 2. The output $Q_T = -\sum_{i=1}^{T} \xi_i k_i^*$ is a weighted sum of T classification

Algorithm 1. Naive boosting algorithm

Require: $h_0 \in \mathbb{R}^H$, $i = 0$, $T \in \mathbb{N}^*$ (number of iterations) and $(\xi_j)_{\{j\in\mathbb{N}\}}$ a family of learning rates.

While $i < T$ do
Calculate $\partial_{h_i} F$.
Calculate k_i^* associated to $\partial_{h_i} F$ (projection step).
$h_{i+1} = h_i - \xi_i k_i^*$
$i = i + 1$
end While, output h_T

Algorithm 2. Minimization of $\overline{J}_{RLE}$ with boosting

Require: $Q_0 \equiv 0$, $i = 0$, $T \in \mathbb{N}^*$ and $(\xi_j)_{\{j\in\mathbb{N}\}}$ a family of learning rates.

While $i < T$ do
Calculate k_i^* associated to $\partial_{Q_i} \overline{J}_{RLE}$. (projection step)
$Q_{i+1} = Q_i - \xi_i k_i^*$, $i = i + 1$
end While, output Q_T

trees: $\{k_i^*\}_{1\leq i\leq T}$. Those T trees can be seen as the features of the problem which are automatically found by the boosting algorithm. The only step that we have to clarify is the calculation of k^*. For this particular choice of weak learners, we have the following Theorem that shows us how to compute it:

Theorem 3. *Calculating* $k^* = \text{argmax}_{k\in K}\langle k, \partial_Q \overline{J}_{RLE}\rangle$*, where* $Q \in \mathbb{R}^{S\times A}$*, corresponds to training a weighted classification tree with the following training data set:*

$$D_C = \big(((s_j, a_j), w_j, -1) \cup ((s_j, a_j^*), w_j, 1)\big)_{\{1\leq j\leq N_E\}}$$
$$\cup\, ((s_i, a_i), w_i, o_i)_{\{1\leq i\leq N_{RL}\}} \cup ((s'_p, a'_p), w_p, -o_p)_{\{1\leq p\leq N_{RL}\}},$$

We recall that an element of a training data set of a weighted classification tree as the following form: (x, w, o) *where* x *is the input,* w *the weight and* o *is the output. With* $(s_j, a_j) \in D_E$*,* (s_i, a_i) *corresponds to the first two elements in a sampled transition* $(s_i, a_i, r_i, s'_i) \in D_{RL}$*,* s'_p *corresponds to the fourth element in a sampled transition* $(s_p, a_p, r_p, s'_p) \in D_{RL}$ *and:*

$$a_j^* = \underset{a\in A}{\text{argmax}}[Q(s_j, a_j) + l(s_j, \pi_E(s_j), a)], \quad a'_p = \underset{a\in A}{\text{argmax}}\, Q(s'_p, a),$$

$$o_i = \text{sgn}(Q(s_i, a_i) - \overline{T}^* Q(s_i, a_i)), \quad o_p = \text{sgn}\Big(\sum_{i=1}^{N_{RL}} \Big(Q(s_i, a_i) - \overline{T}^* Q(s_i, a_i)\Big)\beta_p(s_i, a_i)\Big),$$

$$w_i = \frac{2}{N_P}|Q(s_i, a_i) - \overline{T}^* Q(s_i, a_i)|, \quad w_j = \frac{\lambda}{N_E},$$

$$w_p = \frac{2\gamma}{N_P}|\sum_{i=1}^{N_{RL}} \Big(Q(s_i, a_i) - \overline{T}^* Q(s_i, a_i)\Big)\beta_p(s_i, a_i)|.$$

Proof. Calculating $\partial_Q \overline{J}_{RLE}$ for a given $Q \in \mathbb{R}^{S\times A}$ is done as follows:

$$\partial_Q \max_{a\in A}\{Q(s_j,a)+l(s_j,\pi_E(s_j),a)\} = \delta_{(s_j,a_j^*)}, \quad \partial_Q Q(s_j,\pi_E(s_j)) = \delta_{(s_j,\pi_E(s_j))},$$

$$\partial_Q(\overline{T}^*Q(s_i,a_i)-Q(s_i,a_i))^2 = 2(Q(s_i,a_i)-\overline{T}^*Q(s_i,a_i))(\delta_{(s_i,a_i)}-\gamma\sum_{p=1}^{N_{RL}}\overline{\beta}_p(s_i,a_i)\delta_{(s'_p,a'_p)}),$$

$$\partial_Q \overline{J}_{RLE} = \frac{\sum_{1\le j\le N_{RL}}\partial_Q(\overline{T}^*Q(s_i,a_i)-Q(s_i,a_i))^2}{N_{RL}} + \frac{\lambda\sum_{1\le j\le N_E}\delta_{(s_j,a_j^*)}-\delta_{(s_j,\pi_E(s_j))}}{N_E}.$$

where $a_j^* = \text{argmax}_{a\in A}[Q(s_j,a_j)+l(s_j,\pi_E(s_j),a)]$ and $a'_p = \text{argmax}_{a\in A}\,Q(s'_p,a)$. Obtaining k^* associated to $\partial_Q\overline{J}_{RLE}$ when K is the set of classification trees from $\mathbb{R}^{S\times A}$ to $\{-1,1\}$ is done as follows. First, we calculate $\langle k, \partial_Q\overline{J}_{RLE}\rangle$:

$$\langle k,\partial_Q\overline{J}_{RLE}\rangle = \frac{2}{N_P}\sum_{i=1}^{N_{RL}}(Q(s_i,a_i)-\overline{T}^*Q(s_i,a_i))(k(s_i,a_i)-\gamma\sum_{p=1}^{N_{RL}}\overline{\beta}_p(s_i,a_i)k(s'_p,a'_p))$$

$$+\frac{\lambda}{N_E}\sum_{j=1}^{N_E}k(s_j,a_j^*)-k(s_j,\pi_E(s_j)).$$

To maximize $\langle k,\partial_Q\overline{J}_{RLE}\rangle$, we have to find a classifier k such that $k(s_j,a_j^*)=1$, $k(s_i,a_i)=o_i=\text{sgn}(Q(s_i,a_i)-\overline{T}^*Q(s_i,a_i))$, $k(s_j,\pi_E(s_j))=-1$ and $k(s_p,a'_p)=-o_p=\text{sgn}(\sum_{i=1}^{N_{RL}}\left(Q(s_i,a_i)-\overline{T}^*Q(s_i,a_i)\right)\beta_p(s_i,a_i))$ for a maximum of inputs while taking into consideration the weight factors for each input. The weight factors are the following $w_i=\frac{2}{N_P}|Q(s_i,a_i)-\overline{T}^*Q(s_i,a_i)|$, $w_p=\frac{2\gamma}{N_P}|\sum_{i=1}^{N_{RL}}\left(Q(s_i,a_i)-\overline{T}^*Q(s_i,a_i)\right)\beta_p(s_i,a_i)|$, and $w_j=\frac{\lambda}{N_E}$ Thus, in order to obtain k^*, we train a classification tree with the following training set:

$$D_C = \left(((s_j,\pi_E(s_j)),w_j,-1)\cup((s_j,a_j^*),w_j,1)\right)_{\{1\le j\le N_E\}}$$
$$\cup\,((s_i,a_i),w_i,o_i)_{\{1\le i\le N_{RL}\}}\cup((s'_p,a'_p),w_p,-o_p)_{\{1\le p\le N_{RL}\}}.$$

5 Experiments

In this section, we compare our algorithms (boosted minimization of $\hat{J}_{RLE}$ noted Residual1 and boosted minimization of $\overline{J}_{RLE}$ noted Residual2) to APID, LSPI and a classification algorithm noted Classif which is the boosted minimization of J_E as done by [19]. The comparison is performed on two different tasks. The first task is a generic task, called the Garnet experiment, where the algorithms are tested on several randomly constructed finite MDPs where there is a specific topology that simulates the ones encountered on real continuous MDP. The second experiment is realized on an LfD benchmark called the Highway problem. As the MDP are finite in our experiment, we choose a tabular representation for the parametric algorithms (LSPI, APID). For the boosted algorithms (Residual1, Residual2 and Classif), the features are automatically chosen by the algorithm

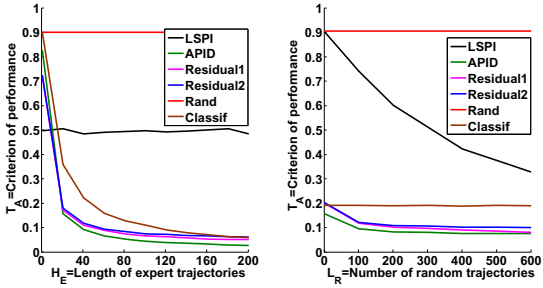
so there is no features choice but we fix the number of weak learners, which are classification trees, to $T = 30$. The regularization parameter λ is fixed at 1 (the expert data and the non expert data are supposed to be of an equal importance), the learning rates are $\xi_i = \frac{1}{i+1}, i \in \mathbb{N}$ and the discount factor is $\gamma = 0.99$ in all of our experiments. Finally, the margin function is $\forall 1 \leq j \leq N_E, \forall a \in A \backslash a_j, l(s_j, a_j, a_j) = 0, l(s_j, a_j, a) = 1$, the Kernels K ans L are the canonical dot products in $\mathbb{R}^{S \times A}$ and in $\mathbb{R}^S$ and $\lambda_K = 10^{-5}$.

5.1 The Garnet Experiment

This experiment focuses on stationary Garnet problems, which are a class of randomly constructed finite MDPs representative of the kind of finite MDPs that might be encountered in practice [3]. A stationary Garnet problem is characterized by 3 parameters: $Garnet(N_S, N_A, N_B)$. The parameters N_S and N_A are the number of states and actions respectively, and N_B is a branching factor specifying the number of next states for each state-action pair. In this experiment, we choose a particular type of Garnets which presents a topological structure relative to real dynamical systems. Those systems are generally multi-dimensional state spaces MDPs where an action leads to different next states close to each other. The fact that an action leads to close next states can model the noise in a real system for instance. Thus, problems such as the highway simulator [13], the mountain car or the inverted pendulum (possibly discretized) are particular cases of this type of Garnets. For those particular Garnets, the state space is composed of d dimensions ($d = 3$ in this particular experiment) and each dimension i has a finite number of elements x_i ($x_i = 5$). So, a state $s = [s^1, s^2, .., s^i, .., s^d]$ is a tuple where each composent s^i can take a finite value between 1 and x_i. In addition, the distance between two states s, s' is $\|s - s'\|^2 = \sum_{i=1}^{i=d} (s^i - s'^i)^2$. Thus, we obtain MDPs with a possible state space size of $\prod_{i=1}^{d} x_i$. The number of actions is $N_A = 5$. For each state action couple (s, a), we choose randomly N_B next states ($N_B = 5$) via a Gaussian distribution of d dimensions centered in s where the covariance matrix is the identity matrix of size d, I_d, multiply by a term σ (here $\sigma = 1$). σ allows handling the smoothness of the MDP: if σ is small the next states s' are close to s and if σ is large, the next states s' can be very far form each other and also from s. The probability of going to each next state s' is generated by partitioning the unit interval at $N_B - 1$ cut points selected randomly. We construct a sparse reward R by choosing $\frac{N_S}{10}$ states (uniform random choice without replacement) where $R(s, a) = 1$, elsewhere $R(s, a) = 0$. For each Garnet problem, it is possible to compute an expert policy $\pi_E = \pi^*$ which is optimal and the expert value function $V^{\pi_E} = f^*_{Q^*}$ via the policy iteration algorithm (as it is a finite MDP where the reward and the dynamics are perfectly known). In addition, we recall that the value function for a policy π is $V^\pi_R = f^\pi_{Q^\pi}$.

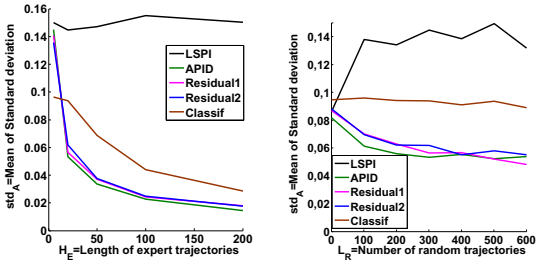
In this experiment, we construct 100 Garnets $\{G_p\}_{1 \leq p \leq 100}$ as explained before. For each Garnet G_p, we build 10 data sets $\{D^{p,q}_E\}_{1 \leq q \leq 10}$ composed of L_E trajectories of H_E expert demonstrations $(s_i, \pi_E(s_i))$ and 10 data sets $\{D^{p,q}_{RL}\}_{1 \leq q \leq 10}$ of L_R trajectories of H_R sampled transitions of the random policy (for each state, the action is uniformly chosen over the set of actions)

(s_i, a_i, s'_i, r_i). Each trajectory begins from a state chosen uniformly over the state space, this uniform distribution is noted ρ. Then the RLED algorithms (APID, Residual1 and Residual2) are fed with the data sets $D_E^{p,q}$ and $D_{RL}^{p,q}$, LSPI is fed with $D_{RL}^{p,q}$ and the Classif algorithm is fed with $D_E^{p,q}$. Each algorithm outputs a function $Q_A^{p,q} \in \mathbb{R}^{S\times A}$ and the policy associated to $Q_A^{p,q}$ is $\pi_A^{p,q}(s) = \text{argmax}_{a\in A} Q_A^{p,q}(s,a)$. In order to quantify the performance of a given algorithm, we calculate the criterion $T_A^{p,q} \frac{\mathbb{E}_\rho[V^{\pi_E} - V^{\pi_A^{p,q}}]}{\mathbb{E}_\rho[V^{\pi_E}]}$, where $V^{\pi_A^{p,q}}$ is calculated via the policy evaluation algorithm. The mean performance criterion T_A is $\frac{1}{1000}\sum_{p=1}^{100}\sum_{q=1}^{10} T_A^{p,q}$. We also calculate, for each algorithm, the variance criterion $\text{std}_A^p = \left(\frac{1}{10}\sum_{q=1}^{10}(T_A^{p,q} - \frac{1}{10}\sum_{q=1}^{10} T_A^{p,q})^2\right)^{\frac{1}{2}}$ and the resulting mean variance criterion is $\text{std}_A = \frac{1}{100}\sum_1^{100}\text{std}_A^p$. In Fig. 1(a), we plot the performance versus the length of the expert trajectories when $L_R = 300$, $H_R = 5$, $L_E = 5$ in order to see how the RLED algorithm manage to leverage the expert data. In Fig. 1(a), we see that the three RLED algorithms have quite the same performance. However, contrary to APID where the features are given by the user (the tabular representation has a size of 725 features), Residual1 and Residual2 manages to learn automatically 30 trees (which can be seen as features) in order to obtain the same performance as APID. The RLED algorithms outperforms the LSPI and Classif algorithm which was expected. We observe the same results in the experiments leaded by [12]. When the number of expert data grows, the RLED algorithms performance is getting better which shows that they are able to leverage those expert data. Besides, we observe that Residual1 and Residual2 have the same performance which shows that using an RKHS embedding is not critical in that case. In Fig. 1(b), we plot the performance versus the number of random trajectories when $H_R = 5$, $H_E = 50$, $L_E = 5$ in order to see the effects of adding non expert data on the RLED algorithms performance. In Fig. 1(b), we can observe that there is a gap between RLED algorithms and the Classif algorithm, and that LSPI does not manage to obtain the same results even when the number of data gets bigger. The gap between RLED and the Classif algorithm gets bigger as the number of RL data is growing which shows that RLED



(a) H_E is evolving (b) L_R is evolving

Fig. 1. Garnet Experiment



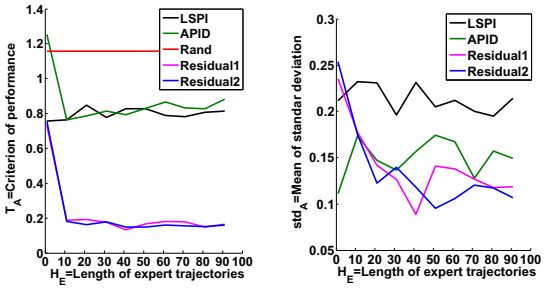
(a) H_E is evolving (b) L_R is evolving

Fig. 2. Garnet Experiment

methods are able to use those data to improve their performance independently of the optimization technique and the parametrization used. In Fig. 2(a) and Fig. 2(b), we plot the mean variance.

5.2 RLED for the Highway

The Highway problem is used as benchmark in the LfD literature [1,24,13]. In this problem, the goal is to drive a car on a busy three-lane highway with randomly generated traffic. The car can move left and right, accelerate, decelerate and keep a constant speed (5 actions). The expert optimizes a handcrafted reward R which favorises speed, punishes off-road, punishes even more collisions and is neutral otherwise. This reward is quite sparse. We have 729 states corresponding to: 9 horizontal positions for the car, 3 horizontal and 9 vertical positions for the closest traffic car and 3 possible speeds. We compute π_E via the policy iteration algorithm as the dynamics P and the reward R of the car driving simulator are known (but unknown for the algorithm user). Here, we build 100 data sets $\{D_E^q\}_{1\leq q\leq 100}$ composed of L_E trajectories of H_E expert demonstrations $(s_i, \pi_E(s_i))$ and 100 data sets $\{D_{RL}^q\}_{1\leq q\leq 100}$ of L_R trajectories of H_R sampled transitions of the random policy. Each trajectory begins from a state chosen uniformly over the state space and we use the same criterion of performance as in the Garnet experiment. We plot the performance versus the length of expert trajectories with $L_E = 5$, $H_R = 5$ and $L_R = 50$. In Fig. 3(a), we observe that Residual1 and Residual2 have clearly better performances than APID in that particular experiment. This can be explained by the fact that the tabular representation for the Highway problem is much bigger than in the in the Garnets experiments (3645 features) and only few features are important. As our algorithms are non-parametric, they are able to select the necessary features in order to find a good policy. Here, the number of data is too small compared to the size of the tabular representation and this can explained why parametric algorithms such as LSPI and APID can not obtain satisfying results. Thus, this observation makes us believe that our algorithms are suited to scale up.



(a) H_E is evolving (b) H_E is evolving

Fig. 3. Highway Experiment

6 Conclusion

In this paper, we present two new algorithms that minimize the empirical norm of the OBR guided by constraints define by expert demonstrations. These algorithms tackle the problem of RLED. Our algorithms are original in the sense that are not derived form the classical Approximate Dynamic Programming framework and manage to alleviate the difficulties inherent to the minimization of the OBR which are the non-convexity, the non-differentiability and the bias. Those drawbacks are overcome by the use of a distribution embbeding in an RKHS and a boosting technique which allows us to obtain non-parametric algorithms, avoiding the choice of features. We show also, in our experiments, that our algorithms perform well compared to the only state of the art algorithm APID, which is parametric. Finally, interesting perspective are to improve our algorithms by using better boosting algorithms, test our algorithms on large scale problems and to have a better understanding of our algorithm by a theoretical analysis.

Acknowledgements. The research leading to these results has received partial funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement nř270780.

References

1. Abbeel, P., Ng, A.: Apprenticeship learning via inverse reinforcement learning. In: Proc. of ICML (2004)
2. Antos, A., Szepesvári, C., Munos, R.: Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path. Machine Learning (2008)
3. Archibald, T., McKinnon, K., Thomas, L.: On the generation of markov decision processes. Journal of the Operational Research Society (1995)
4. Aronszajn, N.: Theory of reproducing kernels. Transactions of the American Mathematical Society (1950)

5. Bertsekas, D.: Dynamic programming and optimal control, vol. 1. Athena Scientific, Belmont (1995)
6. Bradtke, S., Barto, A.: Linear least-squares algorithms for temporal difference learning. Machine Learning (1996)
7. Breiman, L.: Classification and regression trees. CRC Press (1993)
8. Clarke, F.: Generalized gradients and applications. Transactions of the American Mathematical Society (1975)
9. Farahmand, A., Munos, R., Szepesvári, C.: Error propagation for approximate policy and value iteration. In: Proc. of NIPS (2010)
10. Grubb, A., Bagnell, J.: Generalized boosting algorithms for convex optimization. In: Proc. of ICML (2011)
11. Judah, K., Fern, A., Dietterich, T.: Active imitation learning via reduction to iid active learning. In: Proc. of UAI (2012)
12. Kim, B., Farahmand, A., Pineau, J., Precup, D.: Learning from limited demonstrations. In: Proc. of NIPS (2013)
13. Klein, E., Geist, M., Piot, B., Pietquin, O.: Inverse reinforcement learning through structured classification. In: Proc. of NIPS (2012)
14. Lagoudakis, M., Parr, R.: Least-squares policy iteration. Journal of Machine Learning Research (2003)
15. Lever, G., Baldassarre, L., Gretton, A., Pontil, M., Grünewälder, S.: Modelling transition dynamics in mdps with rkhs embeddings. In: Proc. of ICML (2012)
16. Munos, R.: Performance bounds in l_p-norm for approximate value iteration. SIAM Journal on Control and Optimization (2007)
17. Piot, B., Geist, M., Pietquin, O.: Learning from demonstrations: Is it worth estimating a reward function? In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) ECML PKDD 2013, Part I. LNCS, vol. 8188, pp. 17–32. Springer, Heidelberg (2013)
18. Puterman, M.: Markov decision processes: Discrete stochastic dynamic programming. John Wiley & Sons (1994)
19. Ratliff, N., Bagnell, J., Srinivasa, S.: Imitation learning for locomotion and manipulation. In: Proc. of IEEE-RAS International Conference on Humanoid Robots (2007)
20. Ratliff, N., Bagnell, J., Zinkevich, M.: Maximum margin planning. In: Proc. of ICML (2006)
21. Ross, S., Gordon, G., Bagnell, J.: A reduction of imitation learning and structured prediction to no-regret online learning. In: Proc. of AISTATS (2011)
22. Shor, N., Kiwiel, K., Ruszcaynski, A.: Minimization methods for non-differentiable functions. Springer (1985)
23. Sriperumbudur, B., Gretton, A., Fukumizu, K., Schölkopf, B., Lanckriet, G.: Hilbert space embeddings and metrics on probability measures. The Journal of Machine Learning Research (2010)
24. Syed, U., Bowling, M., Schapire, R.: Apprenticeship learning using linear programming. In: Proc. of ICML (2008)
25. Yu, B.: Rates of convergence for empirical processes of stationary mixing sequences. The Annals of Probability (1994)