Journal of Machine Learning Research 15 (2014) 289-333 Submitted 9/11; Revised 4/13; Published 1/14

Off-policy Learning With Eligibility Traces: A Survey

Matthieu Geist MATTHIEU.GEIST@SUPELEC.FR
IMS-MaLIS Research Group & UMI 2958 (GeorgiaTech-CNRS)
Supélec
2 rue Edouard Belin
57070 Metz, France

Bruno Scherrer BRUNO.SCHERRER@INRIA.FR
MAIA project-team
INRIA Lorraine
615 rue du Jardin Botanique
54600 Villers-lès-Nancy, France

Editor: Ronald Parr

Abstract

In the framework of Markov Decision Processes, we consider linear *off-policy* learning, that is the problem of learning a linear approximation of the value function of some fixed policy from one trajectory possibly generated by some other policy. We briefly review *on-policy* learning algorithms of the literature (gradient-based and least-squares-based), adopting a unified algorithmic view. Then, we highlight a systematic approach for adapting them to *off-policy* learning *with eligibility traces*. This leads to some known algorithms—off-policy LSTD(λ), LSPE(λ), TD(λ), TDC/GQ(λ)—and suggests new extensions—off-policy FPKF(λ), BRM(λ), gBRM(λ), GTD2(λ). We describe a comprehensive algorithmic derivation of all algorithms in a recursive and memory-efficent form, discuss their known convergence properties and illustrate their relative empirical behavior on Garnet problems. Our experiments suggest that the most standard algorithms on and off-policy LSTD(λ)/LSPE(λ)—and TD(λ) if the feature space dimension is too large for a least-squares approach—perform the best.

Keywords: reinforcement learning, value function estimation, off-policy learning, eligibility traces

1. Introduction

We consider the problem of learning a linear approximation of the value function of some fixed policy in a Markov Decision Process (MDP) framework. This study is performed in the most general situation where learning must be done from a single trajectory possibly generated by some other policy, also known as *off-policy* learning. Given samples, well-known methods for estimating a value function are temporal difference (TD) learning and Monte Carlo (Sutton and Barto, 1998). TD learning with eligibility traces (Sutton and Barto, 1998), known as TD(λ), constitutes a nice bridge between both approaches; by controlling the bias/variance trade-off (Kearns and Singh, 2000), their use can significantly speed up learning. When the value function is approximated through a linear architecture, the depth

λ of the eligibility traces is also known to control the quality of approximation (Tsitsiklis and Van Roy, 1997). Overall, the use of these traces often plays an important practical role.

	gradient-based	least-squares-based
bootstrapping	TD (Sutton and Barto, 1998)	FPKF (Choi and Van Roy, 2006)
residual	gBRM (Baird, 1995)	BRM (Engel, 2005) (Geist and Pietquin, 2010b)
projected fixed point	TDC/GTD2 (Sutton et al., 2009)	LSTD (Bradtke and Barto, 1996) LSPE (Nedić and Bertsekas, 2003)

Table 1: Taxonomy of linearly parameterized estimators for value function approximation (Geist and Pietquin, 2013).

There has been a significant amount of research on parametric linear approximation of the value function, *without eligibility traces* (in the on- or off-policy case). We follow the taxonomy proposed by Geist and Pietquin (2013), briefly recalled in Table 1 and further developed in Section 2. Value function approximators can be categorized depending on the cost function they minimize (based on bootstrapping, on a Bellman residual minimization or on a projected fixed point approach) and on how it is minimized (gradient descent or linear least-squares). Most of these algorithms have been extended to take into account eligibility traces, in the on-policy case. Works on extending these eligibility-trace approaches to off-policy learning are scarcer. They are summarized in Table 2 (algorithms in black). The first motivation of this article is to argue that it is conceptually simple to extend *all* the algorithms of Table 1 so that they can be applied to the off-policy setting *and* use eligibility traces. If this allows re-deriving existing algorithms (in black in Table 2), it also leads to new candidate algorithms (in red in Table 2). The second motivation of this work is to discuss the subtle differences between these intimately-related algorithms, and to provide some comparative insights on their empirical behavior (a topic that has to our knowledge not been considered in the literature, even in the simplest *on-policy* and *no-trace* situation).

	gradient-based	least-squares-based
bootstrapping	off-policy TD(λ) (Bertsekas and Yu, 2009)	off-policy FPKF(λ)
residual	off-policy gBRM(λ)	off-policy BRM(λ)
projected fixed point	GQ(λ) a.k.a. off-policy TDC(λ) (Maei and Sutton, 2010) off-policy GTD2(λ)	off-policy LSTD(λ) off-policy LSPE(λ) (Yu, 2010a)

Table 2: Surveyed off-policy and eligibility-traces approaches. Algorithms in black have been published before (provided references), algorithms in red are new.

The rest of this article is organized as follows. Section 2 introduces the background of Markov Decision Processes, describes the state-of-the-art algorithms for learning without eligibility traces, and gives the fundamental idea to extend the methods to the off-policy situation with eligibility traces. Section 3 details this extension for the least-squares approaches: the resulting algorithms are formalized, and we derive recursive and memory-efficient formula for their implementation (this allows online learning without loss of generality, all the more that half of these algorithms are recursive by their very definition), and we discuss their convergence properties. Section 4 does the same job for stochastic gradient approaches, which offers a smaller computational cost (linear per update, instead of quadratic). Last but not least, Section 5 describes an empirical comparison and Section 6 concludes.

2. Background

We consider a Markov Decision Process (MDP), that is a tuple $\{S, A, P, R, \gamma\}$ in which S is a finite state space identified with $\{1, 2, \dots, N\}$, A a finite action space, $P \in \mathcal{P}(S)^{S \times A}$ the set of transition probabilities, $R \in \mathbb{R}^{S \times A}$ the reward function and γ the discount factor. A mapping $\pi \in \mathcal{P}(A)^S$ is called a policy. For any policy π, let P^π be the corresponding stochastic transition matrix, and R^π the vector of mean reward when following π, that is of components $E_{a|\pi,s}[R(s,a)]$. The value $V^\pi(s)$ of state s for a policy π is the expected discounted cumulative reward starting in state s and then following the policy π:

$$V^\pi(s) = E_\pi \left[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s \right],$$

where E_π denotes the expectation over trajectories induced by policy π. The value function satisfies the (linear) Bellman equation:

$$\forall s, \quad V^\pi(s) = E_{s',a|s,\pi}[R(s,a) + \gamma V^\pi(s')].$$

It can be rewritten as the fixed-point of the Bellman evaluation operator: $V^\pi = T^\pi V^\pi$ where for all V, $T^\pi V = R^\pi + \gamma P^\pi V$.

In this article, we are interested in learning an approximation of this value function V^π under some constraints. First, we assume our approximation to be linearly parameterized:

$$\forall s, \quad \hat{V}_\theta(s) = \theta^T \phi(s)$$

with $\theta \in \mathbb{R}^p$ being the parameter vector and $\phi(s) \in \mathbb{R}^p$ the feature vector in state s. This encompasses notably the tabular case (exact representation of the value function). Also, we want to estimate the value function V^π (or equivalently the associated parameter θ) from a single finite trajectory generated using a possibly different behavioral policy π_0.[1] Let μ_0 be the stationary distribution of the stochastic matrix $P_0 = P^{\pi_0}$ of the *behavior policy* π_0 (we assume it exists and is unique). Let D_0 be the diagonal matrix of which the elements are $(\mu_0(s_i))_{1 \le i \le N}$. Let Φ be the matrix of feature vectors:

$$\Phi = [\phi(1) \dots \phi(N)]^T.$$

[1] This can be easily extended to multiple finite trajectories.

As we consider a linear approximation, the considered value functions belong to the space spanned by Φ. The projection Π_0 onto this hypothesis space with respect to the μ_0-quadratic norm, which will be central for the understanding of the algorithms, has the following closed-form:

$$\Pi_0 = \Phi(\Phi^T D_0 \Phi)^{-1}\Phi^T D_0.$$

If π_0 is different from π, it is called an off-policy setting. Notice that all algorithms considered in this paper use this Π_0 projection operator, that is the projection according to the observed data.[2] It would certainly be interesting to consider the projection according to the stationary distribution of π, the (unobserved) target policy: this would reduce off-policy learning to on-policy learning. However, this would require re-weighting samples according to the stationary distribution of the target policy π, which is unknown and probably as difficult to estimate as the value function itself.

2.1 Standard Algorithms For On-policy Learning Without Traces

We now review existing on-policy linearly parameterized temporal difference learning algorithms (see Table 1). In this case, the behavior and target policies are the same, so we omit the subscript 0 for the policy (π) and the projection (Π). We assume that a trajectory $(s_1, a_1, r_1, s_2, \dots, s_i, a_i, r_i, s_{i+1}, \dots, s_n, a_n, r_n, s_{n+1})$ sampled according to the policy π is available, and will explain how to compute the i^{th} iterate for several algorithms. For all $j \leq i$, let us introduce the empirical Bellman operator at step j:

$$\begin{aligned} \hat{T}_j \;:\; & \mathbb{R}^S \to \mathbb{R} \\ & V \mapsto r_j + \gamma V(s_{j+1}) \end{aligned}$$

so that $\hat{T}_j V$ is an unbiased estimate of $TV(s_j)$.

Projected fixed point approaches aim at finding the fixed-point of the operator being the composition of the projection onto the hypothesis space and the Bellman operator. In other words, they search for the fixed-point $\hat{V}_\theta = \Pi T \hat{V}_\theta$, Π being the just introduced projection operator. Solving the following fixed-point problem,

$$\theta_i = \operatorname*{argmin}_{\omega \in \mathbb{R}^p} \sum_{j=1}^{i} \left(\hat{T}_j \hat{V}_{\theta_i} - \hat{V}_\omega(s_j) \right)^2,$$

with a least-squares approach corresponds to the Least-Squares Temporal Differences (LSTD) algorithm of Bradtke and Barto (1996). Recently, Sutton et al. (2009) proposed two algorithms reaching the same objective, Temporal Difference with gradient Correction (TDC) and Gradient Temporal Difference 2 (GTD2), by performing a stochastic gradient descent of the function $\theta \mapsto \|\hat{V}_\theta - \Pi T \hat{V}_\theta\|^2$ which is minimal (and equal to 0) when $\hat{V}_\theta = \Pi T \hat{V}_\theta$.

[2] As far as we know, there are two notable exceptions. Precup et al. (2001) propose an algorithm that updates parameters according to full trajectories (not according to transitions, as all approaches to be reviewed next). Therefore, the distribution weighting the projection operator is the one of the starting states of these trajectories instead of the one involved by the behavioral policy. Another work to move in a different direction is the off-policy approach of Kolter (2011): samples are weighted such that the projection operator composed with the Bellman operator is non-expansive: this is weaker than finding the projection of the stationary distribution, but offers some guarantees. In this article, we consider only the Π_0 projection.

A related approach consists in building a recursive algorithm that repeatedly mimics the iteration $\hat{V}_{\theta_i} \simeq \Pi T \hat{V}_{\theta_{i-1}}$. In practice, we aim at minimizing

$$\omega \mapsto \sum_{j=1}^{i} \left(\hat{T}_j \hat{V}_{\theta_{i-1}} - \hat{V}_\omega(s_j) \right)^2 .$$

Performing the minimization exactly through a least-squares method leads to the Least-Squares Policy Evaluation (LSPE) algorithm of Bertsekas and Ioffe (1996). If this minimization is approximated by a stochastic gradient descent, this leads to the classical Temporal Difference (TD) algorithm (Sutton and Barto, 1998).

Bootstrapping approaches consist in treating value function approximation after seeing the i^{th} transition as a supervised learning problem, by replacing the unobserved values $V^\pi(s_j)$ at states s_j by some estimate computed from the trajectory until the transition (s_j, s_{j+1}), the best such estimate being $\hat{T}_j \hat{V}_{\theta_{j-1}}$. This amounts to minimizing the following function:

$$\omega \mapsto \sum_{j=1}^{i} \left(\hat{T}_j \hat{V}_{\theta_{j-1}} - \hat{V}_\omega(s_j) \right)^2 . \tag{1}$$

Choi and Van Roy (2006) proposed the Fixed-Point Kalman Filter (FPKF), a least-squares variation of TD that minimizes exactly the function of Equation (1). If the minimization is approximated by a stochastic gradient descent, this gives—again—the classical TD algorithm (Sutton and Barto, 1998).

Finally, **residual approaches** aim at minimizing the distance between the value function and its image through the Bellman operator, $\|V - TV\|^2_{\mu_0}$. Based on a trajectory, this suggests the following function to minimize

$$\omega \mapsto \sum_{j=1}^{i} \left(\hat{T}_j \hat{V}_{\omega} - \hat{V}_\omega(s_j) \right)^2 ,$$

which is a surrogate of the objective $\|V - TV\|^2_{\mu_0}$ that is biased (Antos et al., 2006). This cost function has originally been proposed by Baird (1995) who minimized it using a stochastic gradient approach (this algorithm being referred here as gBRM for gradient-based Bellman Residual Minimization). Both the *parametric* Gaussian Process Temporal Differences (GPTD) algorithm of Engel (2005) and the *linear* Kalman Temporal Differences (KTD) algorithm of Geist and Pietquin (2010b) can be shown to minimize the above cost using a least-squares approach, and are thus the very same algorithm, that we will refer to as BRM (for Bellman Residual Minimization) in the remaining of this paper.[3]

To sum up, it thus appears that after the i^{th} transition has been observed, the above mentioned algorithms behave according to the following pattern:

$$\text{move from } \theta_{i-1} \text{ to } \theta_i \text{ towards the minimum of } \omega \mapsto \sum_{j=1}^{i} \left(\hat{T}_j \hat{V}_{\xi} - \hat{V}_\omega(s_j) \right)^2 , \tag{2}$$

either through a least-squares approach or a stochastic gradient descent. Each of the algorithms mentioned above is obtained by substituting θ_i, θ_{i-1}, θ_{j-1} or ω for ξ.

[3] Note that this is only true in the linear case. GPTD and KTD were both introduced in a more general setting: GPTD is non-parametric and KTD is motivated by the goal of handling nonlinearities.

2.2 Towards Off-policy Learning With Traces

It is now easy to preview, at least at a high level, how one may extend the previously described algorithms so that they can deal with eligibility traces and off-policy learning.

Eligibility Traces. The idea of eligibility traces amounts to looking for the fixed-point of the following variation of the Bellman operator (Bertsekas and Tsitsiklis, 1996)

$$\forall V \in \mathbb{R}^S, \quad T^\lambda V = (1-\lambda)\sum_{k=0}^{\infty} \lambda^k T^{k+1} V$$

that makes a geometric average with parameter $\lambda \in (0,1)$ of the powers of the original Bellman operator T. Clearly, any fixed-point of T is a fixed-point of T^λ and vice-versa. After some simple algebra, one can see that:

$$\begin{aligned} T^\lambda V &= (I - \lambda\gamma P)^{-1}(R + (1-\lambda)\gamma P V) \\ &= V + (I - \lambda\gamma P)^{-1}(R + \gamma P V - V). \end{aligned} \tag{3}$$

This leads to the following well-known *temporal difference* expression in some state s

$$\begin{aligned} T^\lambda V(s) &= V(s) + E_\pi\left[\sum_{k=i}^{\infty}(\gamma\lambda)^{k-i}\Big(r_k + \gamma V(s_{k+1}) - V(s_k)\Big)\Big| s_i = s\right] \\ &= V(s) + \sum_{k=i}^{\infty}(\gamma\lambda)^{k-i}\delta_{ik}(s) \end{aligned}$$

where we recall that E_π means that the expectation is done according to the target policy π, and where $\delta_{ik}(s) = E_\pi\Big[r_k + \gamma V(s_{k+1}) - V(s_k)\Big| s_i = s\Big]$ is the expected temporal-difference (Sutton and Barto, 1998). With $\lambda = 0$, we recover the Bellman evaluation equation. With $\lambda = 1$, this is the definition of the value function as the expected and discounted cumulative reward: $T^1 V(s) = E_\pi[\sum_{k=i}^{\infty}\gamma^{k-i} r_k | s_i = s]$.

Off-policy Learning. As before, we assume that we are given a trajectory $(s_1, a_1, r_1, s_2, \dots, s_j, a_j, r_j, s_{j+1} \dots, s_n, a_n, r_n, s_{n+1})$, except now that it may be generated from some behavior policy possibly different from the target policy π of which we want to estimate the value. We are going to describe how to compute the i^{th} iterate for several algorithms. For any $i \leq k$, unbiased estimates of the temporal difference terms $\delta_{ik}(s_k)$ can be computed through importance sampling (Ripley, 1987). Indeed, for all s, a, let us introduce the following weight:

$$\rho(s,a) = \frac{\pi(a|s)}{\pi_0(a|s)}.$$

In our trajectory context, for any j and k, write

$$\rho_j^k = \prod_{l=j}^{k} \rho_l \text{ with } \rho_l = \rho(s_l, a_l)$$

with the convention that if $k < j$, $\rho_j^k = 1$. With these notations,

$$\hat{\delta}_{ik} = \rho_i^k \hat{T}_k V - \rho_i^{k-1} V(s_k)$$

is an unbiased estimate of $\delta_{ik}(s_i)$, from which we may build an estimate $\hat{T}_{j,i}^\lambda V$ of $T^\lambda V(s_j)$ (we will describe this very construction separately for the least-squares and the stochastic gradient as they slightly differ).

Then, by replacing the empirical operator $\hat{T}_j$ in Equation (2) by $\hat{T}_{j,i}^\lambda$, we get the general pattern for off-policy trace-based algorithms:

$$\text{move from } \theta_{i-1} \text{ to } \theta_i \text{ towards the minimum of } \omega \mapsto \sum_{j=1}^{i} \left(\hat{T}_{j,i}^\lambda \hat{V}_\xi - \hat{V}_\omega(s_j) \right)^2, \qquad (4)$$

either through a least-squares approach or a stochastic gradient descent after having instantiated $\xi = \theta_i, \theta_{i-1}, \theta_{j-1}$ or ω. This process, including in particular the precise definition of the empirical operator $\hat{T}_{j,i}^\lambda$, will be further developed in the next two sections.[4] Since they are easier to derive, we begin by focusing on least-squares algorithms (right column of Table 2) in Section 3. Then, Section 4 focuses on stochastic gradient-based algorithms (left column of Table 2).

3. Least-squares Extensions To Eligibility Traces And Off-policy Learning

First, we consider the least-squares solution to the problem described in Equation (4). At their i^{th} step, the algorithms that we are about to describe will compute the parameter θ_i by *exactly* solving the following problem:

$$\theta_i = \operatorname*{argmin}_{\omega \in \mathbb{R}^p} \sum_{j=1}^{i} \left(\hat{T}_{j,i}^\lambda \hat{V}_\xi - \hat{V}_\omega(s_j) \right)^2$$

where we define the following empirical *truncated* approximation of T_λ:

$$\hat{T}_{j,i}^\lambda : \mathbb{R}^S \to \mathbb{R}$$

$$V \mapsto V(s_j) + \sum_{k=j}^{i} (\gamma\lambda)^{k-j} \hat{\delta}_{jk} = V(s_j) + \sum_{k=j}^{i} (\gamma\lambda)^{k-j} \left(\rho_j^k \hat{T}_k V - \rho_j^{k-1} V(s_k) \right).$$

Though different definitions of this operator may lead to practical implementations, note that $\hat{T}_{j,i}^\lambda$ only uses samples seen before time i: this very feature—considered by all existing works in the literature—will enable us to derive recursive and low-memory algorithms.

Recall that a linear parameterization is chosen here, $\hat{V}_\xi(s_i) = \xi^T \phi(s_i)$. We adopt the following notations:

$$\phi_i = \phi(s_i),\ \Delta\phi_i = \phi_i - \gamma\rho_i\phi_{i+1} \text{ and } \tilde{\rho}_j^{k-1} = (\gamma\lambda)^{k-j}\rho_j^{k-1}.$$

[4] Note that we let the empirical operator $\hat{T}_{j,i}^\lambda$ depends on the index j of the sample (as before) but also on the step i of the algorithm. This will be particularly useful for the derivation of the recursive and memory-efficient least-squares algorithms that we present in the next section.

The generic cost function to be solved is therefore:

$$\theta_i = \operatorname*{argmin}_{\omega\in\mathbb{R}^p} J(\omega;\xi) \quad \text{with} \quad J(\omega;\xi) = \sum_{j=1}^{i}(\phi_j^T\xi + \sum_{k=j}^{i}\tilde{\rho}_j^{k-1}(\rho_k r_k - \Delta\phi_k^T\xi) - \phi_j^T\omega)^2. \tag{5}$$

Before deriving existing and new least-squares algorithms, as announced, some technical lemmas are required.

The first lemma allows computing directly the inverse of a rank-one perturbed matrix.

Lemma 1 (Sherman-Morrison) *Assume that A is an invertible $n\times n$ matrix and that $u,v\in\mathbb{R}^n$ are two vectors satisfying $1+v^TA^{-1}u\neq 0$. Then:*

$$(A+uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1+v^TA^{-1}u}.$$

The next lemma is simply a rewriting of imbricated sums. However, it is quite important here as it will allow stepping from the operator $\hat{T}_{j,i}^{\lambda}$ (operator which depends on future of s_j, so acasual)—*forward view* of eligibility traces—to the recursion over parameters using eligibility traces (dependence on only past samples)—*backward view* of eligibility traces. In other words, the forward view is a theoretical way of mixing backups that shifts parametrically (through the choice of λ) from the standard Bellman operator to the Monte Carlo one. However, it cannot be implemented easily, as it requires knowing the future states. On the other hand, the backward view, which is equivalent (see notably Lemma 2 and Proposition 6), is a more mechanistic and convenient viewpoint that allows performing the same updates using solely information gathered in the states encountered in the past. See Sutton and Barto (1998, Chapter 7) for further discussion on *backward/forward views*.

Lemma 2 *Let $f\in\mathbb{R}^{\mathbb{N}\times\mathbb{N}}$ and $n\in\mathbb{N}$. We have:*

$$\sum_{i=1}^{n}\sum_{j=i}^{n} f(i,j) = \sum_{i=1}^{n}\sum_{j=1}^{i} f(j,i)$$

We are now ready to mechanically derive the off-policy algorithms LSTD(λ), LSPE(λ), FPKF(λ) and BRM(λ). This is what we do in the following subsections.

3.1 Off-policy LSTD(λ)

The Least-Squares Temporal Difference algorithm, that computes directly a fixed-point of the projected Bellman operator, has originally been introduced in the no-trace and on-policy case by Bradtke and Barto (1996). It has been extended to eligibility traces by Boyan (1999), to off-policy (through state-action value function approximation) learning (without traces) by Lagoudakis and Parr (2003), and to off-policy learning with traces by Yu (2010a).

The off-policy LSTD(λ) algorithm actually corresponds to instantiating Problem (5) with $\xi = \theta_i$:

$$\theta_i = \operatorname*{argmin}_{\omega\in\mathbb{R}^p} \sum_{j=1}^{i}(\phi_j^T\theta_i + \sum_{k=j}^{i}\tilde{\rho}_j^{k-1}(\rho_k r_k - \Delta\phi_k^T\theta_i) - \phi_j^T\omega)^2.$$

This can be solved by zeroing the gradient respectively to ω:

$$\begin{aligned} \theta_i &= (\sum_{j=1}^{i} \phi_j \phi_j^T)^{-1} \sum_{j=1}^{i} \phi_j (\phi_j^T \theta_i + \sum_{k=j}^{i} \tilde{\rho}_j^{k-1} (\rho_k r_k - \Delta\phi_k^T \theta_i)) \\ \Leftrightarrow \qquad 0 &= \sum_{j=1}^{i} \sum_{k=j}^{i} \phi_j \tilde{\rho}_j^{k-1} (\rho_k r_k - \Delta\phi_k^T \theta_i), \end{aligned}$$

which, through Lemma 2, is equivalent to:

$$0 = \sum_{j=1}^{i} (\sum_{k=1}^{j} \phi_k \tilde{\rho}_k^{j-1}) (\rho_j r_j - \Delta\phi_j^T \theta_i).$$

Introducing the (importance-based) eligibility vector z_j:

$$z_j = \sum_{k=1}^{j} \phi_k \tilde{\rho}_k^{j-1} = \sum_{k=1}^{j} \phi_k (\gamma\lambda)^{j-k} \prod_{m=k}^{j-1} \rho_m = \gamma\lambda\rho_{j-1} z_{j-1} + \phi_j, \tag{6}$$

one obtains the following batch estimate:

$$\theta_i = (\sum_{j=1}^{i} z_j \Delta\phi_j^T)^{-1} \sum_{j=1}^{i} z_j \rho_j r_j = (A_i)^{-1} b_i \tag{7}$$

where

$$A_i = \sum_{j=1}^{i} z_j \Delta\phi_j^T \quad \text{and} \quad b_i = \sum_{j=1}^{i} z_j \rho_j r_j. \tag{8}$$

Thanks to Lemma 1, the inverse $M_i = (A_i)^{-1}$ can be computed recursively:

$$M_i = (\sum_{j=1}^{i} z_j \Delta\phi_j^T)^{-1} = M_{i-1} - \frac{M_{i-1} z_i \Delta\phi_i^T M_{i-1}}{1 + \Delta\phi_i^T M_{i-1} z_i}.$$

This can be used to derive a recursive estimate:

$$\begin{aligned} \theta_i &= (\sum_{j=1}^{i} z_j \Delta\phi_j^T)^{-1} \sum_{j=1}^{i} z_j \rho_j r_j = (M_{i-1} - \frac{M_{i-1} z_i \Delta\phi_i^T M_{i-1}}{1 + \Delta\phi_i^T M_{i-1} z_i})(\sum_{j=1}^{i-1} z_j r_j \rho_j + z_i \rho_i r_i) \\ &= \theta_{i-1} + \frac{M_{i-1} z_i}{1 + \Delta\phi_i^T M_{i-1} z_i} (\rho_i r_i - \Delta\phi_i^T \theta_{i-1}). \end{aligned}$$

Writing the gain $K_i = \frac{M_{i-1} z_i}{1 + \Delta\phi_i^T M_{i-1} z_i}$, this gives Algorithm 1.

This algorithm has been proposed and analyzed by Yu (2010a). The author proves the following result: if the *behavior* policy π_0 induces an irreducible Markov chain and chooses with positive probability any action that may be chosen by the *target* policy π, and if the compound (linear) operator $\Pi_0 T^\lambda$ has a unique fixed-point, then off-policy LSTD(λ)

Algorithm 1: Off-policy LSTD(λ)

```
Initialization;
Initialize vector $\theta_0$ and matrix $M_0$ ;
Set $z_0 = 0$;
for $i = 1, 2, \dots$ do
    Observe $\phi_i, r_i, \phi_{i+1}$ ;
    Update traces ;
    $z_i = \gamma\lambda\rho_{i-1}z_{i-1} + \phi_i$ ;
    Update parameters ;
    $K_i = \frac{M_{i-1}z_i}{1+\Delta\phi_i^T M_{i-1}z_i}$ ;
    $\theta_i = \theta_{i-1} + K_i(\rho_i r_i - \Delta\phi_i^T\theta_{i-1})$ ;
    $M_i = M_{i-1} - K_i(M_{i-1}^T\Delta\phi_i)^T$;
```

converges to it almost surely.[5] Formally, it converges to the solution θ^* of the so-called *projected fixed-point* equation:

$$V_{\theta^*} = \Pi_0 T^\lambda V_{\theta^*}. \tag{9}$$

Using the expression of the projection Π_0 and the form of the Bellman operator in Equation (3), it can be seen that θ^* satisfies (see Yu, 2010a for details)

$$\theta^* = A^{-1}b$$

where

$$A = \Phi^T D_0(I - \gamma P)(I - \lambda\gamma P)^{-1}\Phi \quad \text{and} \quad b = \Phi^T D_0(I - \lambda\gamma P)^{-1}R. \tag{10}$$

The core of the analysis of Yu (2010a) consists in showing that $\frac{1}{i}A_i$ and $\frac{1}{i}b_i$ defined in Equation (8) respectively converge to A and b almost surely. Through Equation (7), this implies the convergence of θ_i to θ^*.

3.2 Off-policy LSPE(λ)

The Least-Squares Policy Evaluation algorithm, that computes iteratively the fixed point of the projected Bellman operator, was originally introduced by Bertsekas and Ioffe (1996) and first analyzed in an on-policy context by Nedić and Bertsekas (2003). Its extension to off-policy learning with traces was briefly mentioned by Yu (2010a).

The off-policy LSPE(λ) algorithm corresponds to instantiate $\xi = \theta_{i-1}$ in Problem (5):

$$\theta_i = \operatorname*{argmin}_{\omega\in\mathbb{R}^p} \sum_{j=1}^{i}(\phi_j^T\theta_{i-1} + \sum_{k=j}^{i}\tilde{\rho}_j^{k-1}(\rho_k r_k - \Delta\phi_k^T\theta_{i-1}) - \phi_j^T\omega)^2.$$

[5] It is not always the case that $\Pi_0 T^\lambda$ has a unique fixed-point, see Tsitsiklis and Van Roy (1997) for a counter-example.

This can be solved by zeroing the gradient respectively to ω:

$$\begin{aligned}\theta_i &= (\sum_{j=1}^{i} \phi_j \phi_j^T)^{-1} \sum_{j=1}^{i} \phi_j (\phi_j^T \theta_{i-1} + \sum_{k=j}^{i} \tilde{\rho}_j^{k-1} (\rho_k r_k - \Delta\phi_k^T \theta_{i-1})) \\ &= \theta_{i-1} + (\sum_{j=1}^{i} \phi_j \phi_j^T)^{-1} \sum_{j=1}^{i} \sum_{k=j}^{i} \phi_j \tilde{\rho}_j^{k-1} (\rho_k r_k - \Delta\phi_k^T \theta_{i-1}).\end{aligned}$$

Using Lemma 2 and the definition of the eligibility vector z_j in Equation (6), we get:

$$\begin{aligned}\theta_i &= \theta_{i-1} + (\sum_{j=1}^{i} \phi_j \phi_j^T)^{-1} \sum_{j=1}^{i} \sum_{k=1}^{j} \phi_k \tilde{\rho}_k^{j-1} (\rho_j r_j - \Delta\phi_j^T \theta_{i-1}) \\ &= \theta_{i-1} + (\sum_{j=1}^{i} \phi_j \phi_j^T)^{-1} \sum_{j=1}^{i} z_j (\rho_j r_j - \Delta\phi_j^T \theta_{i-1}).\end{aligned}$$

Define the matrix N_i as follows:

$$N_i = (\sum_{j=1}^{i} \phi_j \phi_j^T)^{-1} = N_{i-1} - \frac{N_{i-1}\phi_i \phi_i^T N_{i-1}}{1 + \phi_i^T N_{i-1} \phi_i}, \tag{11}$$

where the second equality follows from Lemma 1. Let A_i and b_i be defined as in the LSTD description in Equation (8). For clarity, we restate their definition along with their recursive writing:

$$\begin{aligned}A_i &= \sum_{j=1}^{i} z_j \Delta\phi_j^T = A_{i-1} + z_i \Delta\phi_{i+1}^T \\ b_i &= \sum_{j=1}^{i} z_j \rho_j r_j = b_{i-1} + z_i \rho_i r_i.\end{aligned}$$

Then, it can be seen that the LSPE(λ) update is:

$$\theta_i = \theta_{i-1} + N_i (b_i - A_i \theta_{i-1}).$$

The overall computation is provided in Algorithm 2.

This algorithm, (briefly) mentioned by Yu (2010a), generalizes the LSPE(λ) algorithm of Bertsekas and Ioffe (1996) to off-policy learning. With respect to LSTD(λ), which computes $\theta_i = (A_i)^{-1} b_i$ at each iteration as stated in Equation (7), LSPE(λ) is fundamentally recursive (as it is based on an iterated fixed-point relation). Along with the almost sure convergence of $\frac{1}{i} A_i$ and $\frac{1}{i} b_i$ to A and b defined in Equation (10), it can be shown that $i N_i$ converges to $N = (\Phi^T D_0 \Phi)^{-1}$—see for instance Nedić and Bertsekas (2003)—so that, asymptotically, LSPE(λ) behaves as:

$$\theta_i = \theta_{i-1} + N(b - A\theta_{i-1}) = Nb + (I - NA)\theta_{i-1}$$

Algorithm 2: Off-policy LSPE(λ)

```
Initialization;
Initialize vector $\theta_0$ and matrix $N_0$ ;
Set $z_0 = 0$, $A_0 = 0$ and $b_0 = 0$;
for $i = 1, 2, \dots$ do
    Observe $\phi_i, r_i, \phi_{i+1}$;
    Update traces ;
    $z_i = \gamma\lambda\rho_{i-1}z_{i-1} + \phi_i$ ;
    Update parameters ;
    $N_i = N_{i-1} - \frac{N_{i-1}\phi_i\phi_i^T N_{i-1}}{1+\phi_i^T N_{i-1}\phi_i}$ ;
    $A_i = A_{i-1} + z_i\Delta\phi_i^T$;
    $b_i = b_{i-1} + \rho_i z_i r_i$;
    $\theta_i = \theta_{i-1} + N_i(b_i - A_i\theta_{i-1})$ ;
```

or using the definition of Π_0, A, b from Equation (10) and T^λ from Equation (3):

$$V_{\theta_i} = \Phi\theta_i = \Phi N b + \Phi(I - NA)\theta_{i-1} = \Pi_0 T^\lambda V_{\theta_{i-1}}. \tag{12}$$

The behavior of this sequence depends on whether the spectral radius of $\Pi_0 T^\lambda$ is smaller than 1 or not. Thus, the analyses of Yu (2010a) and Nedić and Bertsekas (2003) (for the convergence of N_i) imply the following convergence result: under the assumptions required for the convergence of off-policy LSTD(λ), and the additional assumption that the operator $\Pi_0 T^\lambda$ has a spectral radius smaller than 1 (so that it is contracting), LSPE(λ) also converges almost surely to the fixed-point of the compound $\Pi_0 T^\lambda$ operator.

There are two sufficient conditions that can ensure such a desired contraction property. The first one is when one considers on-policy learning, as Nedić and Bertsekas (2003) did when they derived the first convergence proof of (on-policy) LSPE(λ). When the behavior policy π_0 is different from the target policy π, a sufficient condition for contraction is that λ be close enough to 1; indeed, when λ tends to 1, the spectral radius of T^λ tends to zero and can potentially balance an expansion of the projection Π_0. In the off-policy case, when γ is sufficiently big, a small value of λ can make $\Pi_0 T^\lambda$ expansive (see Tsitsiklis and Van Roy 1997 for an example in the case $\lambda = 0$) and off-policy LSPE(λ) will then diverge. Eventually, Equations (9) and (12) show that when $\lambda = 1$, both LSTD(λ) and LSPE(λ) asymptotically coincide (as $T^1 V$ does not depend on V).

3.3 Off-policy FPKF(λ)

The Fixed Point Kalman Filter algorithm is a bootstrapped recursive least-squares approach to value function approximation originally introduced by Choi and Van Roy (2006). Its extensions to eligibility traces and to off-policy learning are new.

The off-policy FPKF(λ) algorithm corresponds to instantiate $\xi = \theta_{j-1}$ in Problem (5):

$$\theta_i = \operatorname*{argmin}_{\omega \in \mathbb{R}^p} \sum_{j=1}^{i} (\phi_j^T \theta_{j-1} + \sum_{k=j}^{i} \tilde{\rho}_j^{k-1} (\rho_k r_k - \Delta\phi_k^T \theta_{j-1}) - \phi_j^T \omega)^2.$$

This can be solved by zeroing the gradient respectively to ω:

$$\theta_i = N_i \sum_{j=1}^{i} \phi_j (\phi_j^T \theta_{j-1} + \sum_{k=j}^{i} \tilde{\rho}_j^{k-1} (\rho_k r_k - \Delta\phi_k^T \theta_{j-1})),$$

where N_i is the matrix introduced for LSPE(λ) in Equation (11). For clarity, we restate its definition here and its recursive writing:

$$N_i = (\sum_{j=1}^{i} \phi_j \phi_j^T)^{-1} = N_{i-1} - \frac{N_{i-1}\phi_i \phi_i^T N_{i-1}}{1 + \phi_i^T N_{i-1} \phi_i}. \tag{13}$$

Using Lemma 2, one obtains:

$$\theta_i = N_i (\sum_{j=1}^{i} \phi_j \phi_j^T \theta_{j-1} + \sum_{j=1}^{i} \sum_{k=1}^{j} \phi_k \tilde{\rho}_k^{j-1} (\rho_j r_j - \Delta\phi_j^T \theta_{k-1})).$$

With respect to the previously described algorithms, the difficulty here is that on the right side there is a dependence with all the previous terms θ_{k-1} for $1 \leq k \leq i$. Using the symmetry of the dot product $\Delta\phi_j^T \theta_{k-1} = \theta_{k-1}^T \Delta\phi_j$, it is possible to write a recursive algorithm by introducing the trace matrix Z_j that integrates the subsequent values of θ_k as follows:

$$Z_j = \sum_{k=1}^{j} \tilde{\rho}_k^{j-1} \phi_k \theta_{k-1}^T = Z_{j-1} + \gamma\lambda\rho_{j-1}\phi_j \theta_{j-1}^T.$$

With this notation we obtain:

$$\theta_i = N_i (\sum_{j=1}^{i} \phi_j \phi_j^T \theta_{j-1} + \sum_{j=1}^{i} (z_j \rho_j r_j - Z_j \Delta\phi_j)).$$

Using Equation (13) and a few algebraic manipulations, we end up with:

$$\theta_i = \theta_{i-1} + N_i (z_i \rho_i r_i - Z_i \Delta\phi_i).$$

This is the parameter update as provided in Algorithm 3.

As LSPE(λ), this algorithm is fundamentally recursive. However, its overall behavior is quite different. As we discussed for LSPE(λ), iN_i can be shown to tend asymptotically to $N = (\Phi^T D_0 \Phi)^{-1}$ and FPKF(λ) iterates eventually resemble:

$$\theta_i = \theta_{i-1} + \frac{1}{i} N (z_i \rho_i r_i - Z_i \Delta\phi_i).$$

Algorithm 3: Off-policy FPKF(λ)

```
Initialization;
Initialize vector $\theta_0$ and matrix $N_0$ ;
Set $z_0 = 0$ and $Z_0 = 0$;
for $i = 1, 2, \dots$ do
    Observe $\phi_i, r_i, \phi_{i+1}$;
    Update traces ;
    $z_i = \gamma\lambda\rho_{i-1} z_{i-1} + \phi_i$ ;
    $Z_i = \gamma\lambda\rho_{i-1} Z_{i-1} + \phi_i \theta_{i-1}^T$;
    Update parameters ;
    $N_i = N_{i-1} - \frac{N_{i-1}\phi_i\phi_i^T N_{i-1}}{1+\phi_i^T N_{i-1}\phi_i}$ ;
    $\theta_i = \theta_{i-1} + N_i(z_i \rho_i r_i - Z_i \Delta\phi_i)$ ;
```

The term in brackets is a random component (that only depends on the previous transitions) and $\frac{1}{i}$ acts as a learning coefficient that asymptotically tends to 0. In other words, FPKF(λ) has a *stochastic approximation* flavor. In particular, one can see FPKF(0) as a stochastic approximation of LSPE(0). Indeed, asymptotically, FPKF(0) does the following update

$$\theta_i = \theta_{i-1} + \frac{1}{i} N(\rho_i \phi_i r_i - \phi_i \Delta\phi_i^T \theta_{i-1}),$$

and one can notice that $\rho_i \phi_i r_i$ and $\phi_i \Delta\phi_i^T$ are samples of A and b to which A_i and b_i converge through LSPE(0). When $\lambda > 0$, the situation is less clear—up to the fact that since $T^1 V$ does not depend on V, we expect FPKF to asymptotically behave like LSTD and LSPE when λ tends to 1.

Due to its much more involved form (notably the matrix trace Z_j integrating the values of all the values θ_k from the start), it does not seem easy to provide a guarantee for FPKF(λ), even in the on-policy case. To our knowledge, there does not exist any *proof of convergence* for stochastic approximation algorithms in the off-policy case with traces, and a related result for FPKF(λ) thus seems difficult.[6] Based on the above-mentioned relation between FPKF(0) and LSPE(0) and the experiments we have run (see Section 5), we conjecture that off-policy FPKF(λ) has the same asymptotic behavior as LSPE(λ). We leave the formal study of this algorithm for future work.

3.4 Off-policy BRM(λ)

The Bellman Residual Minimization algorithm is a least-squares approach that minimizes directly the Bellman residual. The off-policy BRM(λ) algorithm corresponds to instantiate $\xi = \omega$ in Problem (5):

[6] An analysis of TD(λ), with a simplifying assumption that forces the algorithm to stay bounded is given by Yu (2010a). An analysis of GQ(λ) is provided by Maei and Sutton (2010), with an assumption on the second moment of the traces, which—as explained in Proposition 2 of Yu (2010a)—does not hold in general. A full analysis of these algorithms thus remains to be done. See also Sections 4.1 and 4.2.

$$\begin{aligned}\theta_i &= \operatorname*{argmin}_{\omega\in\mathbb{R}^p} \sum_{j=1}^{i}(\phi_j^T\omega + \sum_{k=j}^{i}\tilde{\rho}_j^{k-1}(\rho_k r_k - \Delta\phi_k^T\omega) - \phi_j^T\omega)^2\\ &= \operatorname*{argmin}_{\omega\in\mathbb{R}^p} \sum_{j=1}^{i}(\sum_{k=j}^{i}\tilde{\rho}_j^{k-1}(\rho_k r_k - \Delta\phi_k^T\omega))^2.\end{aligned}$$

Define

$$\psi_{j\to i} = \sum_{k=j}^{i}\tilde{\rho}_j^{k-1}\Delta\phi_k \text{ and } z_{j\to i} = \sum_{k=j}^{i}\tilde{\rho}_j^{k-1}\rho_k r_k.$$

This yields the following batch estimate:

$$\theta_i = \operatorname*{argmin}_{\omega\in\mathbb{R}^p}\sum_{j=1}^{i}(z_{j\to i} - \psi_{j\to i}^T\omega)^2 = (\tilde{A}_i)^{-1}\tilde{b}_i \tag{14}$$

where

$$\tilde{A}_i = \sum_{j=1}^{i}\psi_{j\to i}\psi_{j\to i}^T \quad\text{and}\quad \tilde{b}_i = \sum_{j=1}^{i}\psi_{j\to i}z_{j\to i}.$$

The transformation of this batch estimate into a recursive update rule is somewhat tedious (it involves three "trace" variables), and the details are deferred to Appendix A for clarity. The resulting BRM(λ) method is provided in Algorithm 4. Note that at each step, this algorithm involves the inversion of a 2×2 matrix (involving the 2×2 identity matrix I_2), inversion that admits a straightforward analytical solution. The computational complexity of an iteration of BRM(λ) is thus $O(p^2)$ (as for the preceding least-squares algorithms).

GPTD and KTD, which are close to BRM, have also been extended with some trace mechanism; however, GPTD(λ) (Engel, 2005), KTD(λ) (Geist and Pietquin, 2010a) and the just described BRM(λ) are different algorithms.[7] Briefly, GPTD(λ) is very close to LSTD(λ) and KTD(λ) uses a different Bellman operator.[8] As BRM(λ) builds a linear system whose solution is updated recursively, it resembles LSTD(λ). However, the system it builds is different. The following theorem, proved in Appendix B, partially characterizes the behavior of BRM(λ) and its potential limit.[9]

Theorem 3 *Assume that the stochastic matrix P_0 of the* behavior *policy is irreducible and has stationary distribution μ_0. Further assume that there exists a coefficient $\beta < 1$ such that*

$$\forall(s,a),\quad \lambda\gamma\rho(s,a)\le\beta. \tag{15}$$

[7] GPTD(λ) is not exactly a generalization of GPTD as it does not reduce to it when $\lambda = 0$. It is rather a technical variation that bridges a gap with the Monte Carlo approach.

[8] The corresponding loss is $(\hat{T}_{j,i}^0\hat{V}(\omega) - \hat{V}_\omega(s_j) + \gamma\lambda(\hat{T}_{j+1,i}^1\hat{V}(\omega) - \hat{V}_\omega(s_{j+1})))^2$. With $\lambda = 0$ it gives $\hat{T}_{j,i}^0$ and with $\lambda = 1$ it provides $\hat{T}_{j,i}^1$.

[9] Our proof is similar to that of Proposition 4 of Bertsekas and Yu (2009). The overall arguments are the following: Equation (15) implies that the traces can be truncated at some depth l, whose influence on the potential limit of the algorithm vanishes when l tends to ∞. For all l, the l-truncated version of the algorithm can easily be analyzed through the ergodic theorem for Markov chains. Making l tend to ∞ allows tying the convergence of the original arguments to that of the truncated version. Eventually, the formula for the limit of the truncated algorithm is computed and one derives the limit.

Algorithm 4: Off-policy BRM(λ)

```
Initialization;
Initialize vector $\theta_0$ and matrix $C_0$ ;
Set $y_0 = 0$, $\mathfrak{D}_0 = 0$ and $z_0 = 0$;
for $i = 1, 2, \dots$ do
    Observe $\phi_i, r_i, \phi_{i+1}$;
    Pre-update traces ;
    $y_i = (\gamma\lambda\rho_{i-1})^2 y_{i-1} + 1$ ;
    Compute ;
    $U_i = \left(\sqrt{y_i}\Delta\phi_i + \frac{\gamma\lambda\rho_{i-1}}{\sqrt{y_i}}\mathfrak{D}_{i-1} \quad \frac{\gamma\lambda\rho_{i-1}}{\sqrt{y_i}}\mathfrak{D}_{i-1}\right)^T$ ;
    $V_i = \left(\sqrt{y_i}\Delta\phi_i + \frac{\gamma\lambda\rho_{i-1}}{\sqrt{y_i}}\mathfrak{D}_{i-1} \quad -\frac{\gamma\lambda\rho_{i-1}}{\sqrt{y_i}}\mathfrak{D}_{i-1}\right)^T$ ;
    $W_i = \left(\sqrt{y_i}\rho r_i + \frac{\gamma\lambda\rho_{i-1}}{\sqrt{y_i}}z_{i-1} \quad -\frac{\gamma\lambda\rho_{i-1}}{\sqrt{y_i}}z_{i-1}\right)^T$;
    Update parameters ;
    $\theta_i = \theta_{i-1} + C_{i-1}U_i\left(I_2 + V_iC_{i-1}U_i\right)^{-1}\left(W_i - V_i\theta_{i-1}\right)$ ;
    $C_i = C_{i-1} - C_{i-1}U_i\left(I_2 + V_iC_{i-1}U_i\right)^{-1}V_iC_{i-1}$ ;
    Post-update traces ;
    $\mathfrak{D}_i = (\gamma\lambda\rho_{i-1})\mathfrak{D}_{i-1} + \Delta\phi_i y_i$ ;
    $z_i = (\gamma\lambda\rho_{i-1})z_{i-1} + r_i\rho_i y_i$ ;
```

Then $\frac{1}{i}\tilde{A}_i$ and $\frac{1}{i}\tilde{b}_i$ respectively converge almost surely to

$$\tilde{A} = \Phi^T\left[D - \gamma DP - \gamma P^T D + \gamma^2 D' + S(I - \gamma P) + (I - \gamma P^T)S^T\right]\Phi$$
$$\tilde{b} = \Phi^T\left[(I - \gamma P^T)Q^T D + S\right]R^\pi$$

where we wrote:

$$D = \operatorname{diag}\left((I - (\lambda\gamma)^2\tilde{P}^T)^{-1}\mu_0\right), \qquad Q = (I - \lambda\gamma P)^{-1},$$
$$D' = \operatorname{diag}\left(\tilde{P}^T(I - (\lambda\gamma)^2\tilde{P}^T)^{-1}\mu_0\right), \qquad S = \lambda\gamma(DP - \gamma D')Q,$$

and where $\tilde{P}$ is the matrix whose coordinates are $\tilde{p}_{ss'} = \sum_a \pi(a|s)\rho(s,a)P(s'|s,a)$. Then, the BRM($\lambda$) algorithm converges with probability 1 to $\tilde{A}^{-1}\tilde{b}$.

The assumption given by Equation (15) trivially holds in the on-policy case (in which $\rho(s,a) = 1$ for all (s,a)) and in the off-policy case when $\lambda\gamma$ is sufficiently small with respect to the mismatch between policies. Note in particular that this result implies the almost sure convergence of the GPTD/KTD algorithms in the on-policy and no-trace case, a question that was still open in the literature.[10] The matrix $\tilde{P}$, which is in general not a stochastic

[10] See for instance the conclusion of Engel (2005).

matrix, can have a spectral radius bigger than 1; Equation (15) ensures that $(\lambda\gamma)^2\tilde{P}$ has a spectral radius smaller than β so that D and D' are well defined. Removing assumption of Equation (15) does not seem easy, since by tuning $\lambda\gamma$ maliciously, one may force the spectral radius of $(\lambda\gamma)^2\tilde{P}$ to be as close to 1 as one may want, which would make $\tilde{A}$ and $\tilde{b}$ diverge. Though the quantity $\tilde{A}^{-1}\tilde{b}$ may compensate for these divergences, our current proof technique cannot account for this situation and a related analysis constitutes possible future work.

The fundamental idea behind the Bellman Residual approach is to address the computation of the fixed-point of T^λ differently from the previous methods. Instead of computing the projected fixed-point as in Equation (9), one considers the following over-determined system

$$\begin{aligned}
& \Phi\theta \simeq T^\lambda \Phi\theta \\
\Leftrightarrow \quad & \Phi\theta \simeq (I - \lambda\gamma P)^{-1}(R + (1-\lambda)\gamma P \Phi\theta) && \text{by Equation (3)} \\
\Leftrightarrow \quad & \Phi\theta \simeq QR + (1-\lambda)\gamma PQ\Phi\theta \\
\Leftrightarrow \quad & \Psi\theta \simeq QR
\end{aligned}$$

with $\Psi = \Phi - (1-\lambda)\gamma PQ\Phi$, and solves it in a least-squares sense, that is by computing $\theta^* = \bar{A}^{-1}\bar{b}$ with $\bar{A} = \Psi^T\Psi$ and $\bar{b} = \Psi^T QR$. One of the motivations for this approach is that, as opposed to the matrix A of LSTD/LSPE/FPKF, $\bar{A}$ is invertible for all values of λ, and one can always guarantee a finite error bound with respect to the best projection (Schoknecht, 2002; Yu and Bertsekas, 2008; Scherrer, 2010). If the goal of BRM(λ) is to compute $\bar{A}$ and $\bar{b}$ from samples, what it actually computes ($\tilde{A}$ and $\tilde{b}$ as characterized in Theorem 3) will in general be biased because the estimation is based on a single trajectory.[11] Such a bias adds an uncontrolled variance term (Antos et al., 2006) to $\bar{A}$ and $\bar{b}$; an interesting consequence is that $\tilde{A}$ is always non-singular.[12] More precisely, there are two sources of bias in the estimation: one results from the non Monte-Carlo evaluation (the fact that $\lambda < 1$) and the other from the use of the correlated importance sampling factors (as soon as one considers off-policy learning). The interested reader may check that in the on-policy case, and when λ tends to 1, $\tilde{A}$ and $\tilde{b}$ coincide with $\bar{A}$ and $\bar{b}$. However, in the strictly off-policy case, taking $\lambda = 1$ does not prevent the bias due to the correlated importance sampling factors. If we have argued that LSTD/LSPE/FPKF should asymptotically coincide when $\lambda = 1$, we see here that BRM should generally differ in an off-policy situation.

4. Stochastic Gradient Extensions To Eligibility Traces And Off-policy Learning

We have just provided a systematic derivation of all least-squares algorithms for learning with eligibility traces in an off-policy manner. When the number of features p is very large, the $O(p^2)$ complexity involved by a least-squares approach may be prohibitive. In such a situation, a natural alternative is to consider an approach based on a stochastic gradient

[11] It is possible to remove the bias when $\lambda = 0$ by using double samples. However, in the case where $\lambda > 0$, the possibility to remove the bias seems much more difficult.

[12] $\bar{A}$ is by construction positive definite, and $\tilde{A}$ equals $\bar{A}$ plus a positive term (the variance term), and is thus also positive definite.

descent of the objective function of interest (Bottou and Bousquet, 2011; Sutton et al., 2009; Maei and Sutton, 2010).

In this section, we will describe a systematic derivation of stochastic gradient algorithms for learning in an off-policy manner with eligibility traces. The principle followed is the same as for the least-squares approaches: we shall instantiate the algorithmic pattern of Equation (4) by choosing the value of ξ and update the parameter so as move towards the minimum of $J(\theta_i, \xi)$ using a stochastic gradient descent. To make the pattern of Equation (4) precise, we need to define the empirical approximate operator we use. We will consider the *untruncated* $\hat{T}^\lambda_{i,n}$ operators (written in the followings $\hat{T}^\lambda_i$, with a slight abuse of notation):

$$\hat{T}^\lambda_i V = V(s_i) + \sum_{j=i}^{n} (\gamma\lambda)^{j-i} \left(\rho_i^j \hat{T}_j V - \rho_i^{j-1} V(s_j)\right) \tag{16}$$

where n is the total length of the trajectory.

It should be noted that algorithmic derivations in this section will be a little bit more involved than in the least-squares case. First, by instantiating $\xi = \theta_i$, the pattern given in Equation (4) is actually a fixed-point problem onto which one cannot directly perform a stochastic gradient descent; this issue will be addressed in Section 4.2 through the introduction of an auxiliary objective function, following the approach originally proposed by Sutton et al. (2009). A second difficulty is the following: the just introduced empirical operator $\hat{T}^\lambda_i$ depends on the full trajectory after step i (on the future of the process), and is for this reason usually coined a *forward view* estimate. Though it would be possible, in principle, to implement a gradient descent based on this *forward view*, it would not be very memory nor time efficient. Thus, we will follow a usual trick of the literature by deriving recursive algorithms based on a *backward view* estimate that is equivalent to the *forward view* in expectation. To do so, we will repeatedly use the following identity that highlights the fact that the estimate $\hat{T}^\lambda_i V$ can be written as a forward recursion:

Lemma 4 *Let $\hat{T}^\lambda_i$ be the operator defined in Equation* (16) *and let $V \in \mathbb{R}^S$. We have*

$$\hat{T}^\lambda_i V = \rho_i r_i + \gamma\rho_i(1-\lambda)V(s_{i+1}) + \gamma\lambda\rho_i \hat{T}^\lambda_{i+1} V.$$

Proof Using notably the identity $\rho_i^j = \rho_i \rho_{i+1}^j$, we have:

$$\begin{aligned}\hat{T}^\lambda_i V &= V(s_i) + \sum_{j=i}^{n} (\gamma\lambda)^{j-i} \left(\rho_i^j \hat{T}_j V - \rho_i^{j-1} V(s_j)\right) \\ &= V(s_i) + \rho_i \hat{T}_i V - V(s_i) + \gamma\lambda\rho_i \sum_{j=i+1}^{n} (\rho_i^j \hat{T}_j V - \rho_i^{j-1} V(s_j)) \\ &= \rho_i \hat{T}_i V + \gamma\lambda\rho_i \left(\hat{T}^\lambda_{i+1} V - V(s_{i+1})\right). \qquad \blacksquare\end{aligned}$$

To sum up, the "recipe" that we are about to use to derive off-policy gradient learning algorithms based on eligibility traces will consist of the following steps:

1. write the empirical generic cost function of Equation (4) with the untruncated Bellman operator of Equation (16) ;

2. instantiate ξ and derive the gradient-based update rule (with some additional work for $\xi = \theta_i$, see Section 4.2);

3. turn the *forward view* into an equivalent (in expectation) *backward view*.

The next subsection details the precise derivation of the algorithms.

4.1 Off-policy TD(λ)

The Temporal-Difference algorithm (Sutton and Barto, 1998) is a gradient-based bootstrap approach for value function approximation. Because it is the simplest, we begin by considering this bootstrap approach, that is by instantiating $\xi = \theta_{j-1}$. The cost function to be minimized is therefore:

$$\sum_{j=1}^{i} \left(\hat{T}_j^\lambda \hat{V}_{\theta_{j-1}} - \hat{V}_\omega(s_j)\right)^2.$$

Minimized with a stochastic gradient descent, the related update rule is (α_i being a standard learning rate and recalling that $\hat{V}_\omega(s_i) = \omega^T \phi(s_i) = \omega^T \phi_i$):

$$\begin{aligned}\theta_i &= \theta_{i-1} - \frac{\alpha_i}{2} \nabla_\omega \left(\hat{T}_i^\lambda \hat{V}_{\theta_{i-1}} - \hat{V}_\omega(s_i)\right)^2 \Big|_{\omega=\theta_{i-1}} \\ &= \theta_{i-1} + \alpha_i \phi_i \left(\hat{T}_i^\lambda \hat{V}_{\theta_{i-1}} - \hat{V}_{\theta_{i-1}}(s_i)\right). \end{aligned} \tag{17}$$

At this point, one could notice that the exact same update rule would have been obtained by instantiating $\xi = \theta_{i-1}$. This was to be expected: as only the last term of the sum is considered for the update, we have $j = i$, and therefore $\xi = \theta_{i-1} = \theta_{j-1}$.

Equation (17) makes use of a λ-TD error defined as

$$\delta_i^\lambda(\omega) = \hat{T}_i^\lambda \hat{V}_\omega - \hat{V}_\omega(s_i).$$

For convenience, let also δ_i be the standard (off-policy) TD error defined as

$$\delta_i(\omega) = \delta_i^{\lambda=0}(\omega) = \rho_i \hat{T}_i \hat{V}_\omega - \hat{V}_\omega(s_i) = \rho_i \left(r_i + \gamma \hat{V}_\omega(s_{i+1})\right) - \hat{V}_\omega(s_i).$$

The λ-TD error can be expressed as a forward recursion:

Lemma 5 *Let δ_i^λ be the λ-TD error and δ_i be the standard TD error. Then for all ω,*

$$\delta_i^\lambda(\omega) = \delta_i(\omega) + \gamma\lambda\rho_i \delta_{i+1}^\lambda(\omega).$$

Proof This is a corollary of Lemma 4:

$$\begin{aligned} & \hat{T}_i^\lambda V_\omega = \rho_i r_i + \gamma\rho_i(1-\lambda)V_\omega(s_{i+1}) + \gamma\lambda\rho_i \hat{T}_{i+1}^\lambda V_\omega \\ \Leftrightarrow \quad & \hat{T}_i^\lambda V_\omega - V_\omega(s_i) = \rho_i r_i + \gamma\rho_i V_\omega(s_{i+1}) - V_\omega(s_i) + \gamma\lambda\rho_i(\hat{T}_{i+1}^\lambda V_\omega - V_\omega(s_{i+1})) \\ \Leftrightarrow \quad & \delta_i^\lambda(\omega) = \delta_i(\omega) + \gamma\lambda\rho_i \delta_{i+1}^\lambda(\omega). \end{aligned}$$

■

Therefore, we get the following update rule

$$\theta_i = \theta_{i-1} + \alpha_i \phi_i \delta_i^\lambda(\theta_{i-1})$$

with $\delta_i^\lambda(\theta_{i-1}) = \delta_i(\theta_{i-1}) + \gamma\lambda\delta_{i+1}^\lambda(\theta_{i-1})$. The key idea here is to find some backward recursion such that in expectation, when the Markov chain has reached its steady state distribution μ_0, it provides the same result as the forward recursion. Such a backward recursion is given by the following lemma.

Proposition 6 *Let z_i be the eligibility vector, defined by the following recursion:*

$$z_i = \phi_i + \gamma\lambda\rho_{i-1}z_{i-1}.$$

For all ω, we have

$$E_{\mu_0}[\phi_i\delta_i^\lambda(\omega)] = E_{\mu_0}[z_i\delta_i(\omega)].$$

Proof For clarity, we omit the dependence with respect to ω and write below δ_i (resp. δ_i^λ) for $\delta_i(\omega)$ (resp. $\delta_i^\lambda(\omega)$). The result relies on successive applications of Lemma 5. We have:

$$\begin{aligned} E_{\mu_0}[\phi_i\delta_i^\lambda] &= E_{\mu_0}[\phi_i(\delta_i + \gamma\lambda\rho_i\delta_{i+1}^\lambda)] \\ &= E_{\mu_0}[\phi_i\delta_i] + E_{\mu_0}[\phi_i\gamma\lambda\rho_i\delta_{i+1}^\lambda]. \end{aligned}$$

Moreover, we have that $E_{\mu_0}[\phi_i\rho_i\delta_{i+1}^\lambda] = E_{\mu_0}[\phi_{i-1}\rho_{i-1}\delta_i^\lambda]$, as expectation is done according to the stationary distribution, therefore:

$$\begin{aligned} E_{\mu_0}[\phi_i\delta_i^\lambda] &= E_{\mu_0}[\phi_i\delta_i] + \gamma\lambda E_{\mu_0}[\phi_{i-1}\rho_{i-1}\delta_i^\lambda] \\ &= E_{\mu_0}[\phi_i\delta_i] + \gamma\lambda E_{\mu_0}[\phi_{i-1}\rho_{i-1}(\delta_i + \gamma\lambda\rho_i\delta_{i+1}^\lambda)] \\ &= E_{\mu_0}[\delta_i(\phi_i + \gamma\lambda\rho_{i-1}\phi_{i-1} + (\gamma\lambda)^2\rho_{i-1}\rho_{i-2}\phi_{i-2} + \dots)] \\ &= E_{\mu_0}[\delta_i z_i]. \end{aligned}$$

■

This suggests to replace Equation (17) by the following update rule,

$$\theta_i = \theta_{i-1} + \alpha_i z_i \delta_i(\theta_{i-1}),$$

which is equivalent in expectation when the Markov chain has reached its steady state. This is summarized in Algorithm 5.

This algorithm was first proposed in the tabular case by Precup et al. (2000) (who call it per-decision importance sampling). An off-policy TD(λ) algorithm (with function approximation) was proposed by Precup et al. (2001), but it differs significantly from the algorithm just described, since it updates parameters based on full episodic trajectories rather than based on the current transition. Algorithm 5 was actually first proposed much more recently by Bertsekas and Yu (2009).

An important issue for the analysis of this algorithm is the fact that the trace z_i may have an infinite variance, due to importance sampling (Yu, 2010b, Section 3.1). As far as we know, the only existing analysis of off-policy TD(λ) (as provided in Algorithm 5) uses an additional constraint which forces the parameters to be bounded: after each parameter update, the

Algorithm 5: Off-policy TD(λ)

```
Initialization;
Initialize vector $\theta_0$;
Set $z_0 = 0$;
for $i = 1, 2, \dots$ do
    Observe $\phi_i, r_i, \phi_{i+1}$ ;
    Update traces ;
    $z_i = \gamma\lambda\rho_{i-1}z_{i-1} + \phi_i$ ;
    Update parameters ;
    $\theta_i = \theta_{i-1} + \alpha_i z_i(\rho_i r_i - \Delta\phi_i^T\theta_{i-1})$ ;
```

resulting parameter vector is projected onto some predefined compact set. This analysis is performed by Yu (2010b, Section 4.1). Under the standard assumptions of stochastic approximations and most of the assumptions required for the on-policy TD(λ) algorithm, assuming moreover that $\Pi_0 T^\lambda$ is a contraction (which we recall to hold for a big enough λ) and that the predefined compact set used to project the parameter vector is a large enough ball containing the fixed point of $\Pi_0 T^\lambda$, the constrained version of off-policy TD(λ) converges to this fixed-point —therefore, the same solution as off-policy LSTD(λ)—, LSPE(λ) and FPKF(λ). We refer to Yu (2010b, Section 4.1) for further details. An analysis of the unconstrained version of off-policy TD(λ) described in Algorithm 5 is an interesting topic for future research.

4.2 Off-policy TDC(λ) And Off-policy GTD2(λ)

The Temporal Difference with gradient Correction and Gradient Temporal Differences 2 algorithms have been introduced by Sutton et al. (2009) as gradient descent approaches to minimize the norm of the difference between the value function and its image through the projected Bellman operator (they are therefore projected fixed-point approaches). Maei and Sutton (2010) extended TDC to off-policy learning with traces, calling the resulting algorithm GQ(λ).

This corresponds (for all algorithms and extensions) to the case $\xi = \theta_i$, considered in this section. Following the general pattern, at step i, we would like to come up with a new parameter θ_i that moves (from θ_{i-1}) closer to the minimum of the function

$$\omega \mapsto J(\omega, \theta_i) = \left(\hat{T}_j^\lambda \hat{V}_{\theta_i} - \hat{V}_\omega(s_j)\right)^2.$$

This problem is tricky since the function to minimize contains what we want to compute—-θ_i—as a parameter. For this reason we cannot directly perform a stochastic gradient descent of the right hand side. Instead, we will consider an alternative (but equivalent) formulation of the projected fixed-point minimization $\theta = \arg\min_\omega \|V_\omega - \Pi_0 T^\lambda V_\omega\|^2$, and will move from θ_{i-1} to θ_i by making one step of gradient descent of an estimate of the function

$$\theta \mapsto \|V_\theta - \Pi_0 T^\lambda V_\theta\|^2.$$

With the following vectorial notations:

$$\begin{aligned}\hat{\mathbf{V}}_\omega &= \begin{pmatrix}\hat{V}_\omega(s_1) & \dots & \hat{V}_\omega(s_i)\end{pmatrix}^T,\\ \hat{\mathbf{T}}^\lambda\hat{\mathbf{V}}_\omega &= \begin{pmatrix}\hat{T}_1^\lambda\hat{V}_\omega & \dots & \hat{T}_i^\lambda\hat{V}_\omega\end{pmatrix}^T,\\ \tilde{\Phi} &= \begin{bmatrix}\phi(s_1) & \dots & \phi(s_i)\end{bmatrix}^T,\\ \tilde{\Pi}_0 &= \tilde{\Phi}(\tilde{\Phi}^T\tilde{\Phi})^{-1}\tilde{\Phi}^T,\end{aligned}$$

we consider the following objective function:

$$\begin{aligned}J(\theta) &= \left\|\hat{\mathbf{V}}_\theta - \tilde{\Pi}_0\hat{\mathbf{T}}^\lambda\hat{\mathbf{V}}_\theta\right\|^2\\ &= \left(\hat{\mathbf{V}}_\theta - \hat{\mathbf{T}}^\lambda\hat{\mathbf{V}}_\theta\right)^T\tilde{\Pi}_0\left(\hat{\mathbf{V}}_\theta - \hat{\mathbf{T}}^\lambda\hat{\mathbf{V}}_\theta\right)\\ &= \left(\sum_{j=1}^i \delta_j^\lambda(\theta)\phi_j\right)^T\left(\sum_{j=1}^i \phi_j\phi_j^T\right)^{-1}\left(\sum_{j=1}^i \delta_j^\lambda(\theta)\phi_j\right).\end{aligned}$$

This is the derivation followed by Sutton et al. (2009) in the case $\lambda = 0$ and by Maei and Sutton (2010) in the case $\lambda > 0$ (and off-policy learning). Let us introduce the following notation:

$$g_j^\lambda = \nabla\hat{T}_j^\lambda\hat{V}_\theta. \tag{18}$$

Note that since we consider a linear approximation this quantity does not depend on θ. Noticing that $\nabla\delta_j^\lambda(\theta) = \phi_j - g_j^\lambda$, we can compute $\nabla J(\theta)$:

$$\begin{aligned}-\frac{1}{2}\nabla J(\theta) &= -\frac{1}{2}\nabla\left(\sum_{j=1}^i \delta_j^\lambda(\theta)\phi_j\right)^T\left(\sum_{j=1}^i \phi_j\phi_j^T\right)^{-1}\left(\sum_{j=1}^i \delta_j^\lambda(\theta)\phi_j\right)\\ &= -\left(\nabla\sum_{j=1}^i \delta_j^\lambda(\theta)\phi_j\right)^T\left(\sum_{j=1}^i \phi_j\phi_j^T\right)^{-1}\left(\sum_{j=1}^i \delta_j^\lambda(\theta)\phi_j\right)\\ &= \left(\sum_{j=1}^i (\phi_j - g_j^\lambda)\phi_j^T\right)\left(\sum_{j=1}^i \phi_j\phi_j^T\right)^{-1}\left(\sum_{j=1}^i \delta_j^\lambda(\theta)\phi_j\right) \qquad (19)\\ &= \left(\sum_{j=1}^i \delta_j^\lambda(\theta)\phi_j\right) - \left(\sum_{j=1}^i g_j^\lambda\phi_j^T\right)\left(\sum_{j=1}^i \phi_j\phi_j^T\right)^{-1}\left(\sum_{j=1}^i \delta_j^\lambda(\theta)\phi_j\right).\end{aligned}$$

Let $w_i(\theta)$ be a quasi-stationary estimate of the last part, that can be recognized as the solution of a least-squares problem (regression of λ-TD errors δ_j^λ on features ϕ_j):

$$w_i(\theta) \approx \left(\sum_{j=1}^i \phi_j\phi_j^T\right)^{-1}\left(\sum_{j=1}^i \delta_j^\lambda(\theta)\phi_j\right) = \operatorname*{argmin}_\omega \sum_{j=1}^i \left(\phi_j^T\omega - \delta_j^\lambda(\theta)\right)^2.$$

The identification with the above least-squares solution suggests to use the following stochastic gradient descent to form the quasi-stationary estimate:

$$w_i = w_{i-1} + \beta_i \phi_i \left(\delta_i^\lambda(\theta_{i-1}) - \phi_i^T w_{i-1} \right).$$

This update rule makes use of the λ-TD error, defined through a *forward view*. As for the previous algorithm, we can use Proposition 6 to obtain the following *backward view* update rule that is equivalent (in expectation when the Markov chain reaches its steady state):

$$w_i = w_{i-1} + \beta_i \left(z_i \delta_i(\theta_{i-1}) - \phi_i(\phi_i^T w_{i-1}) \right). \tag{20}$$

Using this quasi-stationary estimate, the gradient can be approximated as:

$$-\frac{1}{2}\nabla J(\theta) \approx \left(\sum_{j=1}^{i} \delta_j^\lambda(\theta)\phi_j \right) - \left(\sum_{j=1}^{i} g_j^\lambda \phi_j^T \right) w_i.$$

Therefore, a stochastic gradient descent gives the following update rule for the parameter vector θ:

$$\theta_i = \theta_{i-1} + \alpha_i \left(\delta_i^\lambda(\theta_{i-1})\phi_i - g_i^\lambda \phi_i^T w_i \right). \tag{21}$$

Once again the *forward view* term $\delta_i^\lambda(\theta_{i-1})\phi_i$ can be turned into a *backward view* by using Proposition 6. There remains to work on the term $g_i^\lambda \phi_i^T$.

First, one can notice that the term g_i^λ satisfies a forward recursion.

Lemma 7 *We have*

$$g_i^\lambda = \gamma\rho_i(1-\lambda)\phi_{i+1} + \gamma\lambda\rho_i g_{i+1}^\lambda.$$

Proof This result is simply obtained by applying the gradient to the forward recursion of $\hat{T}_i^\lambda V_\theta$ provided in Lemma 4 (according to θ). ■

Using this, the term $g_i^\lambda \phi_i^T$ can be worked out similarly to the term $\delta_i^\lambda(\theta_{i-1})\phi_i$.

Proposition 8 *Let z_i be the eligibility vector defined in Proposition 6. We have*

$$E_{\mu_0}[g_i^\lambda \phi_i^T] = E_{\mu_0}[\gamma\rho_i(1-\lambda)\phi_{i+1} z_i^T].$$

Proof The proof is similar to that of Proposition 6. Writing $b_i = \gamma\rho_i(1-\lambda)\phi_{i+1}$ and $\eta_i = \gamma\lambda\rho_i$, we have

$$\begin{aligned} E_{\mu_0}[g_i^\lambda \phi_i^T] &= E_{\mu_0}[(b_i + \eta_i g_{i+1}^\lambda)\phi_i^T] \\ &= E_{\mu_0}[b_i\phi_i^T] + E_{\mu_0}[\eta_{i-1}(b_i + \eta_i g_{i+1}^\lambda)\phi_{i-1}^T] \\ &= E_{\mu_0}[b_i z_i^T]. \end{aligned}$$

■

Using this result and Proposition 6, it is natural to replace Equation (21) by an update based on a backward recursion:

$$\theta_i = \theta_{i-1} + \alpha_i \left(z_i\delta_i - \gamma\rho_i(1-\lambda)\phi_{i+1}(z_i^T w_{i-1}) \right). \tag{22}$$

Last but not least, for the estimate w_i to be indeed quasi-stationary, the learning rates should satisfy the following condition (in addition to the classical conditions):

$$\lim_{i\to\infty} \frac{\alpha_i}{\beta_i} = 0.$$

Equations. (22) and (20) define the off-policy TDC(λ) algorithm, summarized in Algorithm 6. It was originally proposed by Maei and Sutton (2010) under the name GQ(λ). We call it off-policy TDC(λ) to highlight the fact that it is the extension of the original TDC algorithm of Sutton et al. (2009) to off-policy learning with traces. One can observe—to our knowledge, this was never mentioned in the literature before—that when $\lambda = 1$, the learning rule of TDC(1) reduces to that of TD(1).

Maei and Sutton (2010) show that the algorithm converges with probability 1 to the same solution as the LSTD(λ) algorithm (that is, to $\theta^* = A^{-1}b$) under some technical assumptions. Contrary to off-policy TD(λ), this algorithm does not requires $\Pi_0 T^\lambda$ to be a contraction in order to be convergent. Unfortunately, one of the assumptions made in the analysis, requiring that the traces z_i have uniformly bounded second moments, is restrictive since in an off-policy setting the traces z_i may easily have an infinite variance (unless the behavior policy is really close to the target policy), as noted by Yu (2010a).[13] A full proof of convergence thus still remains to be done.

Algorithm 6: Off-policy TDC(λ), also known as GQ(λ)

```
Initialization;
Initialize vector $\theta_0$ and $w_0$;
Set $z_0 = 0$;
for $i = 1, 2, \dots$ do
    Observe $\phi_i, r_i, \phi_{i+1}$ ;
    Update traces ;
    $z_i = \gamma\lambda\rho_{i-1} z_{i-1} + \phi_i$ ;
    Update parameters ;
    $\theta_i = \theta_{i-1} + \alpha_i \left( z_i(\rho_i r_i - \Delta\phi_i^T \theta_{i-1}) - \gamma\rho_i(1-\lambda)\phi_{i+1}(z_i^T w_{i-1}) \right)$ ;
    $w_i = w_{i-1} + \beta_i \left( z_i(\rho_i r_i - \Delta\phi_i^T \theta_i) - \phi_i(\phi_i^T w_{i-1}) \right)$ ;
```

Using the same principle—performing a stochastic gradient descent to minimize $J(\theta)$)—, an alternative to TDC, the GTD2 algorithm, was derived by Sutton et al. (2009) in the $\lambda = 0$ case. As far as we know, it has never been extended to off-policy learning with traces; we do it now. Notice that, given the derivation of GQ(λ), obtaining this algorithm is pretty straightforward.

[13] See also Randhawa and Juneja (2004).

To do so, we can start back from Equation (19):

$$\begin{aligned}
-\frac{1}{2}\nabla J(\theta) &= \left(\sum_{j=1}^{i}(\phi_j - g_j^\lambda)\phi_j^T\right)\left(\sum_{j=1}^{i}\phi_j\phi_j^T\right)^{-1}\left(\sum_{j=1}^{i}\delta_j^\lambda(\theta)\phi_j\right) \\
&\approx \left(\sum_{j=1}^{i}(\phi_j - g_j^\lambda)\phi_j^T\right) w_i.
\end{aligned}$$

This suggests the following alternative update rule (based on forward recursion):

$$\theta_i = \theta_{i-1} + \alpha_i(\phi_i - g_i^\lambda)\phi_i^T w_i.$$

Using Proposition 8, it is natural to use the following alternative update rule, based on a backward recursion:

$$\theta_i = \theta_{i-1} + \alpha_i\left(\phi_i(\phi_i^T w_{i-1}) - \gamma\rho_i(1-\lambda)\phi_{i+1}(z_i^T w_{i-1})\right).$$

The update of w_i remains the same, and put together it gives off-policy GTD2(λ), summarized in Algorithm 7. The analysis of this new algorithm constitutes a potential topic for future research.

Algorithm 7: Off-policy GTD2(λ)

```
Initialization;
Initialize vector $\theta_0$ and $w_0$;
Set $z_0 = 0$;
for $i = 1, 2, \dots$ do
    Observe $\phi_i, r_i, \phi_{i+1}$ ;
    Update traces ;
    $z_i = \gamma\lambda\rho_{i-1}z_{i-1} + \phi_i$ ;
    Update parameters ;
    $\theta_i = \theta_{i-1} + \alpha_i\left(\phi_i(\phi_i^T w_{i-1}) - \gamma\rho_i(1-\lambda)\phi_{i+1}(z_i^T w_{i-1})\right)$ ;
    $w_i = w_{i-1} + \beta_i\left(z_i(\rho_i r_i - \Delta\phi_i^T\theta_i) - \phi_i(\phi_i^T w_{i-1})\right)$ ;
```

4.3 Off-policy gBRM(λ)

The algorithm proposed by Baird (1995) minimizes the Bellman residual using a gradient-based approach, in the no-trace and on-policy case. We extend it to eligibility traces and to off-policy learning, which corresponds to instantiate $\xi = \omega$. The cost function to be minimized is then:

$$\sum_{j=1}^{i}\left(\hat{T}_j^\lambda \hat{V}_\omega - \hat{V}_\omega(s_j)\right)^2.$$

Following the negative of the gradient of the last term leads to the following update rule:

$$\begin{aligned}\theta_i &= \theta_{i-1} - \alpha_i \nabla_\omega \left(\hat{T}_i^\lambda \hat{V}_\omega - \hat{V}_\omega(s_i)\right)^2 \Big|_{\omega=\theta_{i-1}}\\ &= \theta_{i-1} - \alpha_i \nabla_\omega \left(\hat{T}_i^\lambda \hat{V}_\omega - \hat{V}_\omega(s_i)\right) \Big|_{\omega=\theta_{i-1}} \left(\hat{T}_i^\lambda \hat{V}_{\theta_{i-1}} - \hat{V}_{\theta_{i-1}}(s_i)\right)\\ &= \theta_{i-1} + \alpha_i \left(\phi_i - g_i^\lambda\right) \delta_i^\lambda(\theta_{i-1}),\end{aligned}$$

recalling the notation $g_i^\lambda = \nabla \hat{T}_i^\lambda \hat{V}_\omega$ first defined in Equation (18).

As usual, this update involves a *forward view*, which we are going to turn into a *backward view*. The term $\phi_i \delta_i^\lambda$ can be worked thanks to Proposition 6. The term $g_i^\lambda \delta_i^\lambda$ is more difficult to handle, as it is the product of two *forward views* (until now, we only considered the product of a *forward view* with a non-recursive term). This can be done thanks to the following original relation (the proof being somewhat tedious, it is deferred to Appendix C):

Proposition 9 *Write* $g_i^\lambda = \nabla_\omega \hat{T}_i^\lambda$ *and define*

$$\begin{aligned} c_i &= 1 + (\gamma\lambda\rho_{i-1})^2 c_{i-1},\\ \zeta_i &= \gamma\rho_i(1-\lambda)\phi_{i+1}c_i + \gamma\lambda\rho_{i-1}\zeta_{i-1}\\ \text{and } d_i &= \delta_i c_i + \gamma\lambda\rho_{i-1} d_{i-1}.\end{aligned}$$

We have that

$$E_{\mu_0}[\delta_i^\lambda g_i^\lambda] = E_{\mu_0}[\delta_i\zeta_i + d_i\gamma\rho_i(1-\lambda)\phi_{i+1} - \delta_i\gamma\rho_i(1-\lambda)\phi_{i+1}c_i].$$

This result (together with Proposition 6) suggests to update parameters as follows:

$$\theta_i = \theta_{i-1} + \alpha_i \left(\delta_i(z_i + \gamma\rho_i(1-\lambda)\phi_{i+1}c_i - \zeta_i) - d_i\gamma\rho_i(1-\lambda)\phi_{i+1}\right).$$

This gives the off-policy gBRM(λ) algorithm, depicted in Algorithm 8. One can observe that gBRM(1) is equivalent to TD(1) (and thus also TDC(1), *cf.* the comment before the description of Algorithm 6). The analysis of this new algorithm is left for future research.

5. Empirical Study

This section aims at empirically comparing the surveyed algorithms. As they only address the policy evaluation problem, we compare the algorithms in their ability to perform policy evaluation (no control, no policy optimization); however, they may straightforwardly be used in an approximate policy iteration approach (Bertsekas and Tsitsiklis, 1996; Munos, 2003). In order to assess their quality, we consider finite problems where the exact value function can be computed.

More precisely, we consider Garnet problems (Archibald et al., 1995), which are a class of randomly constructed finite MDPs. They do not correspond to any specific application, but are totally abstract while remaining representative of the kind of MDP that might be encountered in practice. In our experiments, a Garnet is parameterized by 4 parameters and is written $\mathcal{G}(n_S, n_A, b, p)$: n_S is the number of states, n_A is the number of actions, b

Algorithm 8: Off-policy gBRM(λ)

```
Initialization;
Initialize vector $\theta_0$;
Set $z_0 = 0$, $d_0 = 0$, $c_0 = 0$, $\zeta_0 = 0$;
for $i = 1, 2, \dots$ do
    Observe $\phi_i, r_i, \phi_{i+1}$ ;
    Update traces ;
    $z_i = \phi_i + \gamma\lambda\rho_{i-1}z_{i-1}$ ;
    $c_i = 1 + (\gamma\lambda\rho_{i-1})^2 c_{i-1}$ ;
    $\zeta_i = \gamma\rho_i(1-\lambda)\phi_{i+1}c_i + \gamma\lambda\rho_{i-1}\zeta_{i-1}$ ;
    $d_i = (\rho_i r_i - \Delta\phi_i^T\theta_{i-1})c_i + \gamma\lambda\rho_{i-1}d_{i-1}$ ;
    Update parameters ;
    $\theta_i = \theta_{i-1} + \alpha_i\big((\rho_i r_i - \Delta\phi_i^T\theta_{i-1})(z_i + \gamma\rho_i(1-\lambda)\phi_{i+1}c_i - \zeta_i) - d_i\gamma\rho_i(1-\lambda)\phi_{i+1}\big)$ ;
```

is a branching factor specifying how many possible next states are possible for each state-action pair (b states are chosen uniformly at random and transition probabilities are set by sampling uniform random $b-1$ cut points between 0 and 1) and p is the number of features (for function approximation). The reward is state-dependent: for a given randomly generated Garnet problem, the reward for each state is uniformly sampled between 0 and 1. Features are chosen randomly: Φ is a $n_S \times p$ feature matrix of which each component is randomly and uniformly sampled between 0 and 1. The discount factor γ is set to 0.95 in all experiments.

We consider two types of problems, "small" and "big", respectively corresponding to instances $\mathcal{G}(30, 2, 2, 8)$ and $\mathcal{G}(100, 4, 3, 20)$. We also consider two types of learning: on-policy and off-policy. In the on-policy setting, for each Garnet a policy π to be evaluated is randomly generated (by sampling randomly $n_A - 1$ cut points between 0 and 1 for each state), and trajectories (to be used for learning) are sampled according to this same policy. In the off-policy setting, the policy π to be evaluated is randomly generated the same way, but trajectories are sampled according to a different (similarly randomly generated) behavior policy π_0.

For all algorithms, we choose $\theta_0 = 0$. For least-squares algorithms (LSTD, LSPE, FPKF and BRM), we set the initial matrices (M_0, N_0, C_0) to $10^3 I$ (the higher this value, the more negligible its effect on estimates; we observed that this parameter did not play a crucial role in practice). We run a first set of experiments in order to set all other parameters (eligibility factor and learning rates). We use the following schedule for the learning rates:

$$\alpha_i = \alpha_0 \frac{\alpha_c}{\alpha_c + i} \text{ and } \beta_i = \beta_0 \frac{\beta_c}{\beta_c + i^{\frac{2}{3}}}.$$

More precisely, we generate 30 problems (MDPs and policies) for each possible combination small/big on-policy/off-policy (leading to four cases). For each problem, we generate one trajectory of length 10^4 using the behavioral policy (which is the randomly generated target policy in the on-policy case and the behavior policy in the off-policy case), to be used by all algorithms. For each meta-parameter, we consider the following ranges of values: $\lambda \in \{0, 0.4, 0.7, 0.9, 1\}$, $\alpha_0 \in \{10^{-2}, 10^{-1}, 10^0\}$, $\alpha_c \in \{10^1, 10^2, 10^3\}$, $\beta_0 \in \{10^{-2}, 10^{-1}, 10^0\}$ and $\beta_c \in \{10^1, 10^2, 10^3\}$. Then, we compute the parameter estimates considering all algorithms instantiated with each possible combination of the meta-parameters. This gives for each combination a family $\theta_{i,d}$ with i the number of transitions encountered in the trajectory of the d^{th} problem. Finally, for each case, for all problems and each algorithm, we choose the combination of meta-parameters which minimizes the average error on the last one-tenth of the averaged (over all problems) learning curves (we do this to reduce the sensitivity to the initialization and the transient behavior). Formally, we pick the set of parameters that minimizes the following quantity:

$$\text{err} = \frac{1}{30}\sum_{d=1}^{30}\frac{1}{10^3}\sum_{i=9.10^3}^{10^4}\|\Phi\theta_{i,d} - V^{\pi_d}\|_2.$$

We provide the empirical results of this first set of experiments in Tables 3 to 6. As a complement, we detail in Figure 1 the sensitivity of all algorithms with respect to the *main* parameter λ that controls the eligibility traces (averaged over the 30 problems, with the best global meta-parameters for each choice of λ). We comment these results below.

	λ	α_0	α_c	β_0	β_c	err
LSTD	1.0					2.07
LSPE	1.0					2.07
FPKF	1.0					2.07
BRM	1.0					2.07
TD	1.0	10^{-2}	10^3			2.06
gBRM	1.0	10^{-2}	10^3			2.06
TDC	1.0	10^{-2}	10^3	10^{-2}	10^1	2.06
GTD2	1.0	10^{-2}	10^3	10^{-1}	10^2	2.05

Table 3: Small problem ($\mathcal{G}(30, 2, 2, 8)$), on-policy learning ($\pi = \pi_0$).

Table 3 shows the best global meta-parameters over the 30 considered instances (one trajectory per instance) of a small Garnet problem in an on-policy setting, as well as related efficiency. Numerically, all methods provide equivalent performance (the slight difference of error is not statistically significant, provided the variance of the estimates). All methods use the same eligibility factor ($\lambda = 1$), leading to a Monte Carlo estimate, to reach their best performance. Figure 1 (top, left) shows that this choice of λ does matter and that BRM, gBRM and FPKF are more sensitive to a good choice of the eligibility factor.

Table 4 shows the best global meta-parameters over the 30 considered instances (one trajectory per instance) of a big Garnet problem in an on-policy setting, as well as related performance. These results are consistent with those of the small problem, in the on-policy setting (with rather different meta-parameters, apart from the eligibility factor). Here again, the algorithms need the highest value of λ to perform the best, except TDC and GTD2 that

	λ	α_0	α_c	β_0	β_c	err
LSTD	1.0					1.20
LSPE	1.0					1.20
FPKF	1.0					1.20
BRM	1.0					1.20
TD	1.0	10^{-1}	10^{1}			1.25
gBRM	1.0	10^{-1}	10^{1}			1.25
TDC	0.9	10^{-1}	10^{2}	10^{-1}	10^{2}	1.21
GTD2	0.9	10^{-1}	10^{2}	10^{-2}	10^{3}	1.22

Table 4: Big problem ($\mathcal{G}(100, 4, 3, 20)$), on-policy learning ($\pi = \pi_0$).

take nevertheless a high value of λ. Figure 1 (top, right) suggests that as the problem's size grows, the role of the eligibility factor gets more prominent (but with a similar behavior).

	λ	α_0	α_c	β_0	β_c	err
LSTD	0.4					3.69
LSPE	0.4					3.69
FPKF	0.7					4.74
BRM	0.0					4.42
TD	0.4	10^{-1}	10^{2}			3.85
gBRM	0.0	10^{-2}	10^{1}			10.42
TDC	0.4	10^{-1}	10^{1}	10^{-2}	10^{1}	7.81
GTD2	0.4	10^{-1}	10^{3}	10^{-2}	10^{1}	4.53

Table 5: Small problem ($\mathcal{G}(30, 2, 2, 8)$), off-policy learning ($\pi \neq \pi_0$).

Table 5 reports the best meta-parameters in an off-policy setting for a small problem (still for 30 instances). Regarding the least-squares methods, LSTD and LSPE get the best results, whereas FPKF and BRM suffer more from the off-policy aspect. Regarding gradient methods, TD's performance is good (it is close to that of LSTD/LSPE and better than BRM/FPKF), followed closely by GTD2. TDC and gBRM lead to the worse results. All algorithms use a small or intermediate value of the eligibility factor. Increasing λ would reduce the bias, but the performance would suffer from the variance due to importance sampling, as shown also in Figure 1 (bottom, left).

Eventually, Table 6 shows the meta-parameters and performance in the most difficult situation: the off-policy setting of the big problem. These results are consistent with the off-policy results of the small problem, summarized in Table 5. LSTD and LSPE are the most efficient least-squares algorithms and choose the smallest possible value $\lambda = 0$. FPKF and BRM's performance deteriorate (significantly for the latter). TD behaves very well and GTD2 follows closely. The performance of TDC and gBRM are the worse. Figure 1 (bottom, right) is similar to that of the small problem. It shows that TD (with a good learning rate) is quite stable, in particular more than LSTD/LSPE.

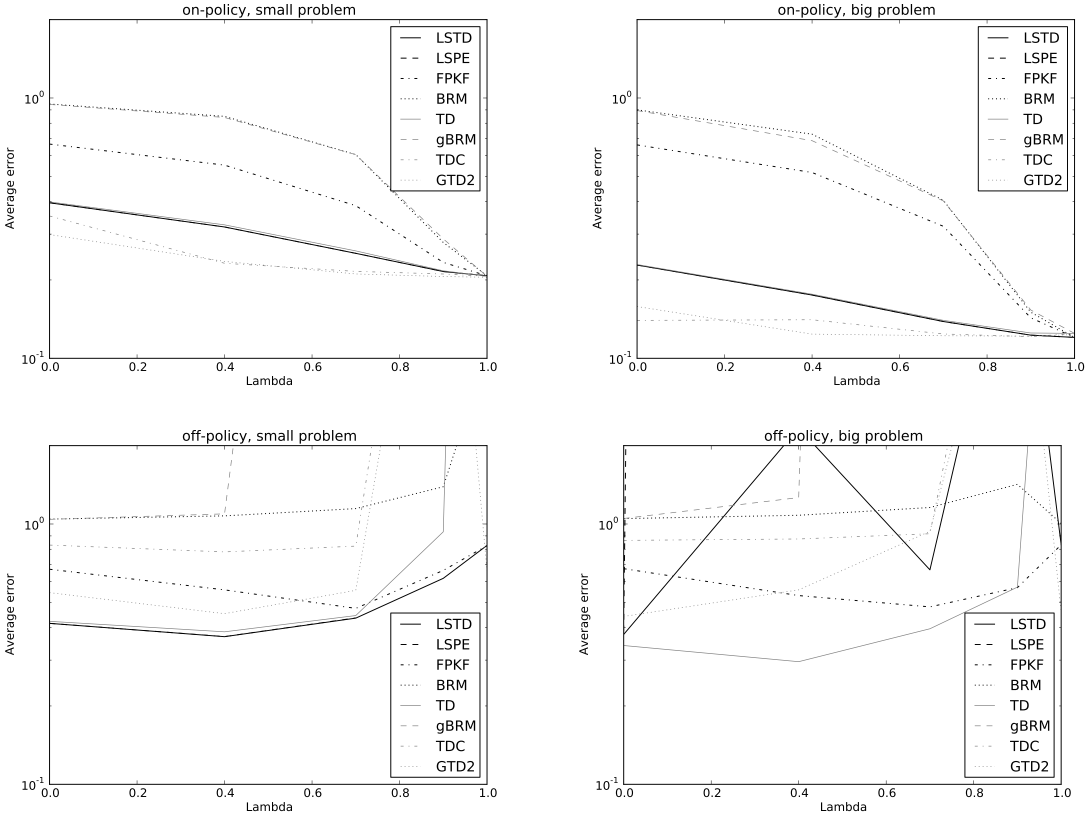

Figure 1: Sensitivity of performance of the algorithms (y-axis, in logarithmic scale) with respect to the eligibility trace parameter λ (x-axis). Left: Small problem ($\mathcal{G}(30, 2, 2, 8)$), right: Big problem ($\mathcal{G}(100, 4, 3, 20)$). Top: on-policy learning ($\pi = \pi_0$), bottom: off-policy learning ($\pi \neq \pi_0$).

The main goal of the series of experiments we have just described was to choose reasonable values for the meta-parameters. We have also used these experiments to quickly comment the relative performance of the algorithms, but this is not statistically significant as this was based on a few (random) problems, onto which meta-parameters have been optimized. Though we will see that the general behavior of the algorithm is globally consistent with what we have seen so far, the series of experiments that we are about to describe aims at providing such a statistically significant performance comparison. For each situation (small and big problems, on- and off-policy), we fix the meta-parameters to the previously reported values and we compare the algorithms on several new instances of the problems. These results are reported on Figures 2 to 5. For each of the 4 problems, we randomly generate 100 instances (MDP and policy to be evaluated). For each such problem, we generate a trajectory of length 10^5. Then, all algorithms learn using this very trajectory. On each figure, we report the average performance (left), measured as the difference between the true

	λ	α_0	α_c	β_0	β_c	err
LSTD	0					3.76
LSPE	0					3.86
FPKF	0.7					4.80
BRM	1.0					10.05
TD	0.4	10^{-1}	10^{1}			2.96
gBRM	0.0	10^{-2}	10^{1}			10.50
TDC	0.0	10^{-1}	10^{1}	10^{-2}	10^{1}	8.65
GTD2	0.0	10^{-1}	10^{3}	10^{-2}	10^{1}	4.41

Table 6: Big problem ($\mathcal{G}(100, 4, 3, 20)$), off-policy learning ($\pi \neq \pi_0$).

value function (computed from the model) and the currently estimated one, $\|V^\pi - \Phi\theta\|_2$, as well as the associated standard deviation (right).

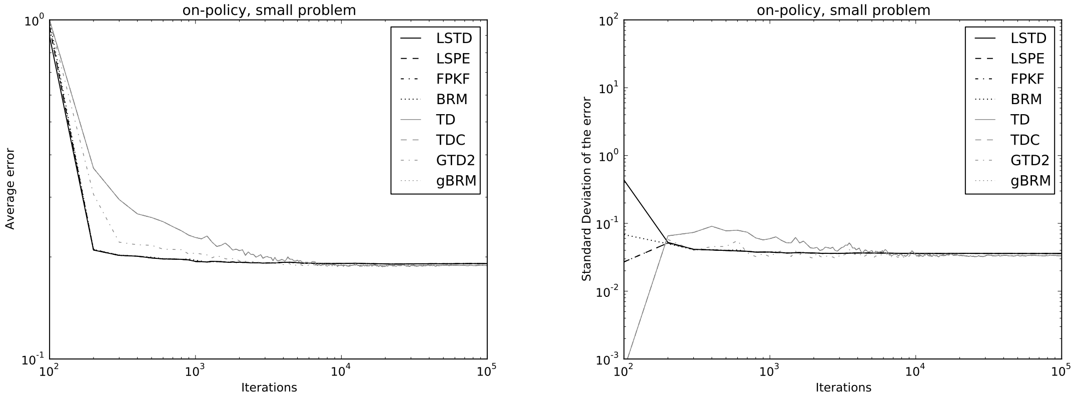

Figure 2: Performance for small problems ($\mathcal{G}(30, 2, 2, 8)$), on-policy learning ($\pi = \pi_0$) (left: average error, right: standard deviation).

We begin by discussing the results in the on-policy setting. Figure 2 compares all algorithms for 100 randomly generated small problems (that is, each run corresponds to different dynamics, reward function, features and evaluated policy), the meta-parameters being those provided in Table 3. All least-squares approaches provide the best results and are bunched together; this was to be expected, as all algorithms use λ equal to 1. The gBRM, TD and TDC algorithms provide the same results (being equivalent with the choice $\lambda = 1$), they are slower than GTD2, which is slower than the least-squares algorithms. Figure 3 compares the algorithms for 100 randomly generated big problems, the meta-parameters being those provided in Table 4. These result are similar to those of the small problem in an off-policy setting, except that TDC has now a different (and slower) behavior, due to the different choice of the eligibility factor ($\lambda = 0.9$). GTD2 is still the better gradient-based algorithm.

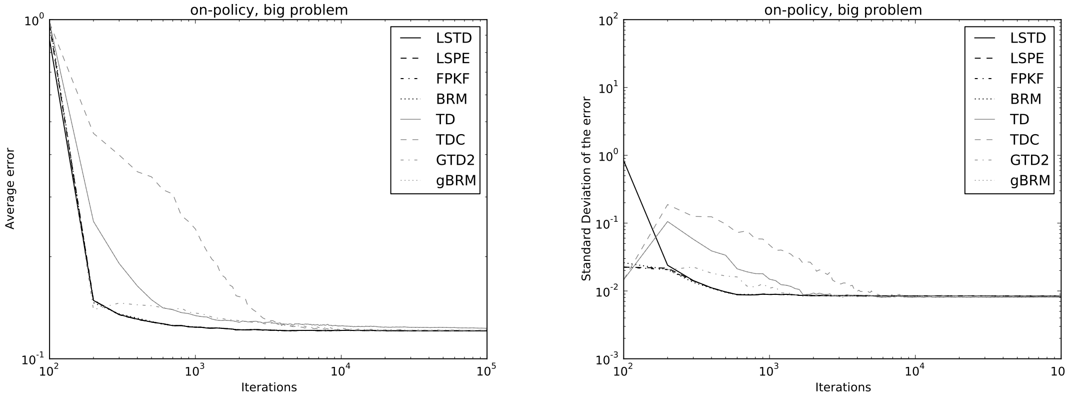

Figure 3: Performance for big problems ($\mathcal{G}(100, 4, 3, 20)$), on-policy learning ($\pi = \pi_0$) (left: average error, right: standard deviation).

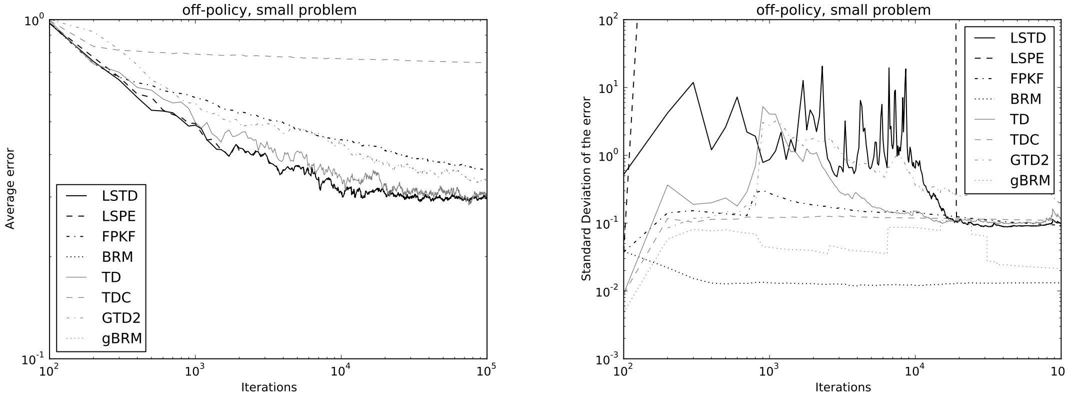

Figure 4: Performance for small problems ($\mathcal{G}(30, 2, 2, 8)$), off-policy learning ($\pi \neq \pi_0$) (left: average error, right: standard deviation).

We now consider the off-policy setting. Figure 4 provides the average performance and standard deviation of the algorithms (meta-parameters being those of Table 5) on 100 small problems. Once again, we can see that LSTD/LSPE provide the best results. The two other least-squares methods (FPKF and BRM) are overtaken by the gradient-based TD algorithm, that follows closely LSTD/LSPE. GTD2 is a little bit slower and TDC is the slowest algorithm. Figure 5 provides the same data for the big problems (with the meta-parameters of Table 6). These results are similar to those of the small problems in an off-policy setting, except that TD is even closer to LSTD/LSPE (but requires the choice of a learning rate).

Summary. Overall, our experiments suggest that the two best algorithms are LSTD and LSPE, since they converge much faster in all situations with less parameter tuning. The gradient-based TD algorithm globally displays a good behavior and constitutes a good alter-

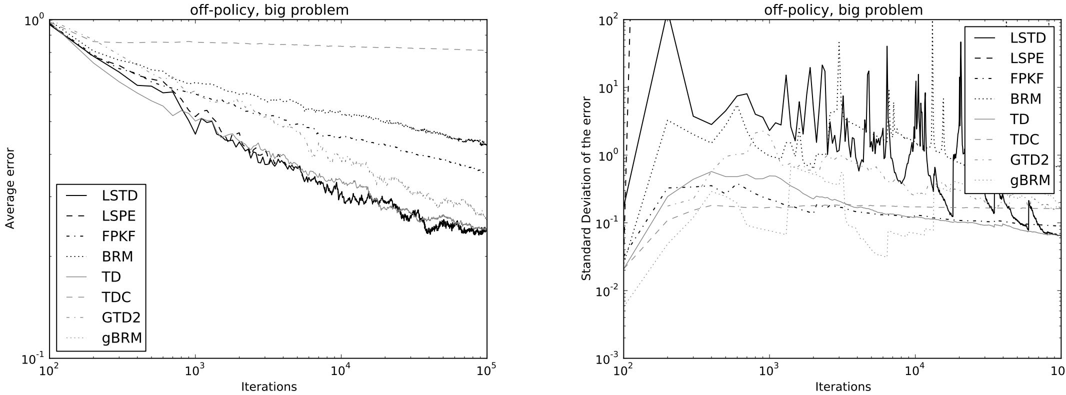

Figure 5: Performance for big problems ($\mathcal{G}(100, 4, 3, 20)$), on-policy learning ($\pi \neq \pi_0$) (left: average error, right: standard deviation).

native when the number p of features is too big for least-squares methods to be implemented. Though some new algorithms/extensions show interesting results (FPKF(λ) is consistently better that the state-of-the-art FPKF by Choi and Van Roy 2006, gBRM works well in the on-policy setting) most of the other algorithms do not seem to be empirically competitive with the *trio* LSTD/LSPE/TD, especially in off-policy situations. In particular, the algorithm introduced specifically for the off-policy setting (TDC/GTD2) are much slower than TD in the off-policy case (but GTD2 is faster in the on-policy experiments, yet with more parameter tuning). Moreover, the condition required for the good behavior of LSPE, FPKF and TD—the contraction of $\Pi_0 T^\lambda$—does not seem to be very restrictive in practice (at least for the Garnet problems we considered): though it is possible to build specific pathological examples where these algorithms diverge, this never happened in our experiments.[14]

6. Conclusion And Future Work

We have considered least-squares and gradient-based algorithms for value estimation in an MDP context. Starting from the on-policy case with no trace, we have recalled that several algorithms (LSTD, LSPE, FPKF and BRM for least-squares approaches, TD, gBRM and TDC/GTD2 for gradient-based approaches) fall in a common algorithmic pattern: Equation (2). Substituting the original Bellman operator by an operator that deals with traces and off-policy samples naturally leads to the state-of-the-art off-policy trace-based versions of LSTD, LSPE, TD and TDC, and suggests natural extensions of FPKF, BRM, gBRM and GTD2. This way, we surveyed many known and new off-policy eligibility trace-based algorithms for policy evaluation.

We have explained how to derive recursive (memory and time-efficient) implementations of all these algorithms and discussed their known convergence properties, including an original analysis of BRM(λ) for sufficiently small λ, that implies the so far not known convergence

[14] A preliminary version of this article (Scherrer and Geist, 2011) contains such examples, and also an example where an adversarial choice of λ leads to the divergence of LSTD(λ).

of GPTD/KTD. Interestingly, it appears that the analysis of off-policy trace-based stochastic gradient algorithms under mild assumptions is still an open problem: the only currently known analysis of TD (Yu, 2010a) only applies to a constrained version of the algorithm, and that of TDC (Maei and Sutton, 2010) relies on an assumption on the boundedness of the second moment traces that is restrictive (Yu, 2010a). Filling this theoretical gap, as well as providing complete analyses for the other gradient algorithms and FPFK(λ) and BRM(λ) constitute important future work.

Finally, we have illustrated and compared the behavior of these algorithms; this constitutes the first exhaustive empirical comparison of linear methods.[15] Overall, our study suggests that even if the use of eligibility traces generally improves the efficiency of all algorithms, LSTD and LSPE consistently provide the best estimates; and in situations where the computational cost is prohibitive for a least-squares approach (when the number p of features is large), TD probably constitutes the best alternative.

Appendix A. Derivation Of The Recursive Formulas For BRM(λ)

We here detail the derivation of off-policy BRM(λ). We will need two technical lemmas. The first one is the Woodbury matrix identity which generalizes the Sherman-Morrison formula (given in Lemma 1).

Lemma 10 (Woodbury) *Let A, U, C and V be matrices of correct sizes, then:*

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}.$$

The second lemma is a rewriting of imbricated sums:

Lemma 11 *Let $f \in \mathbb{R}^{\mathbb{N}\times\mathbb{N}\times\mathbb{N}}$ and $n \in \mathbb{N}$. We have:*

$$\sum_{i=1}^{n}\sum_{j=i}^{n}\sum_{k=i}^{n} f(i,j,k) = \sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=1}^{j} f(k,i,j) + \sum_{i=2}^{n}\sum_{j=1}^{i-1}\sum_{k=1}^{j} f(k,j,i).$$

As stated in Equation (14), we have the following batch estimate for BRM(λ):

$$\theta_i = \operatorname*{argmin}_{\omega\in\mathbb{R}^p} \sum_{j=1}^{i} (z_{j\to i} - \psi_{j\to i}^T \omega)^2 = (\tilde{A}_i)^{-1}\tilde{b}_i,$$

where

$$\psi_{j\to i} = \sum_{k=j}^{i} \tilde{\rho}_j^{k-1} \Delta\phi_k \text{ and } z_{j\to i} = \sum_{k=j}^{i} \tilde{\rho}_j^{k-1}\rho_k r_k$$

and

$$\tilde{A}_i = \sum_{j=1}^{i} \psi_{j\to i}\psi_{j\to i}^T \quad\text{and}\quad \tilde{b}_i = \sum_{j=1}^{i} \psi_{j\to i} z_{j\to i}.$$

[15] To our knowledge, there does not even exist any work reporting and comparing empirical results of LSTD(0) and FPKF(0).

To obtain a recursive formula, these two sums have to be reworked through Lemma 11. Let us first focus on the latter:

$$\begin{aligned}\sum_{j=1}^{i}\psi_{j\to i}z_{j\to i} &= \sum_{j=1}^{i}\sum_{k=j}^{i}\sum_{m=j}^{i}\tilde{\rho}_j^{k-1}\Delta\phi_k\tilde{\rho}_j^{m-1}\rho_m r_m\\ &= \sum_{j=1}^{i}\sum_{k=1}^{j}\sum_{m=1}^{k}\tilde{\rho}_m^{j-1}\Delta\phi_j\tilde{\rho}_m^{k-1}\rho_k r_k + \sum_{j=2}^{i}\sum_{k=1}^{j-1}\sum_{m=1}^{k}\tilde{\rho}_m^{k-1}\Delta\phi_k\tilde{\rho}_m^{j-1}\rho_j r_j.\end{aligned}$$

Writing

$$y_k = \sum_{m=1}^{k}(\tilde{\rho}_m^{k-1})^2 = 1 + (\gamma\lambda\rho_{k-1})^2 y_{k-1},$$

we have that:

$$\sum_{m=1}^{k}\tilde{\rho}_m^{j-1}\tilde{\rho}_m^{k-1} = \tilde{\rho}_k^{j-1}y_k.$$

Therefore:

$$\sum_{j=1}^{i}\psi_{j\to i}z_{j\to i} = \sum_{j=1}^{i}\sum_{k=1}^{j}\tilde{\rho}_k^{j-1}y_k\Delta\phi_j\rho_k r_k + \sum_{j=2}^{i}\sum_{k=1}^{j-1}\tilde{\rho}_k^{j-1}y_k\Delta\phi_k\rho_j r_j.$$

With the following notations:

$$\begin{aligned} z_j &= \sum_{k=1}^{j}\tilde{\rho}_k^{j-1}y_k\rho_k r_k = \gamma\lambda\rho_{j-1}z_{j-1} + \rho_j r_j y_j\\ \text{and}\quad \mathfrak{D}_j &= \sum_{k=1}^{j}\tilde{\rho}_k^{j-1}y_k\Delta\phi_k = \gamma\lambda\rho_{j-1}\mathfrak{D}_{j-1} + y_j\Delta\phi_j,\end{aligned}$$

and with the convention that $z_0 = 0$ and $\mathfrak{D}_0 = 0$, one can write:

$$\sum_{j=1}^{i}\psi_{j\to i}z_{j\to i} = \sum_{j=1}^{i}(\Delta\phi_j\rho_j r_j y_j + \gamma\lambda\rho_{j-1}(\Delta\phi_j z_{j-1} + \rho_j r_j\mathfrak{D}_{j-1})).$$

Similarly, on can show that:

$$\sum_{j=1}^{i}\psi_{j\to i}\psi_{j\to i}^T = \sum_{j=1}^{i}(\Delta\phi_j\Delta\phi_j^T y_j + \gamma\lambda\rho_{j-1}(\Delta\phi_j\mathfrak{D}_{j-1}^T + \mathfrak{D}_{j-1}\Delta\phi_j^T)).$$

Denoting

$$\begin{aligned} u_j &= \sqrt{y_j}\Delta\phi_j,\\ v_j &= \frac{\gamma\lambda\rho_{j-1}}{\sqrt{y_j}}\mathfrak{D}_{j-1},\end{aligned}$$

and I_2 the 2×2 identity matrix, we have:

$$\begin{aligned}\sum_{j=1}^{i} \psi_{j\to i}\psi_{j\to i}^T &= \sum_{j=1}^{i}((u_j+v_j)(u_j+v_j)^T - v_j v_j^T)\\ &= \sum_{j=1}^{i-1} \psi_{j\to i}\psi_{j\to i}^T + \underbrace{\begin{pmatrix} u_i+v_i & v_i\end{pmatrix}}_{=U_i} I_2 \underbrace{\begin{pmatrix}(u_i+v_i)^T \\ -v_i^T\end{pmatrix}}_{=V_i}.\end{aligned}$$

We can apply the Woodbury identity given in Lemma 10:

$$\begin{aligned}C_i = \left(\sum_{j=1}^{i}\psi_{j\to i}\psi_{j\to i}^T\right)^{-1} &= \left(\sum_{j=1}^{i-1}\psi_{j\to i}z_{j\to i} + U_i I_2 V_i\right)^{-1}\\ = C_{i-1} - C_{i-1}U_i\left(I_2 + V_i C_{i-1}U_i\right)^{-1} V_i C_{i-1}.\end{aligned}$$

The other sum can also be reworked:

$$\begin{aligned}\tilde b_i &= \sum_{j=1}^{i}\psi_{j\to i}z_{j\to i} = \sum_{j=1}^{i}\Delta\phi_j r_j y_j + \gamma\lambda\left(\mathfrak{D}_{j-1}r_j + \Delta\phi_j z_{j-1}\right)\\ &= \tilde b_{i-1} + \Delta\phi_i r_i y_i + \gamma\lambda\left(\mathfrak{D}_{i-1}r_i + \Delta\phi_i z_{i-1}\right) = \tilde b_{i-1} + U_i \underbrace{\begin{pmatrix}\sqrt{y_i}r_i + \frac{\gamma\lambda}{\sqrt{y_i}}z_{i-1}\\ -\frac{\gamma\lambda}{\sqrt{y_i}}z_{i-1}\end{pmatrix}}_{=W_i}.\end{aligned}$$

Finally, the recursive BRM(λ) estimate can be computed as follows:

$$\theta_i = C_i\tilde b_i = \theta_{i-1} + C_{i-1}U_i\left(I_2 + V_iC_{i-1}U_i\right)^{-1}\left(W_i - V_i\theta_{i-1}\right).$$

This gives BRM(λ) as provided in Algorithm 4.

Appendix B. Proof Of Theorem 3: Convergence Of BRM(λ)

The proof of Theorem 3 follows the general idea of that of Proposition 4 of Bertsekas and Yu (2009). It is done in 2 steps. First we argue that the limit of the sequence is linked to that of an alternative algorithm for which one cuts the traces at a certain depth l. Then, we show that for all depth l, this alternative algorithm converges almost surely, we explicitly compute its limit and make l tend to infinity to obtain the limit of BRM(λ).

We will only show that $\frac{1}{i}\tilde A_i$ tends to $\tilde A$. The argument is similar for $\frac{1}{i}b_i \to \tilde b$. Consider the following l-truncated version of the algorithm based on the following alternative traces (we here limit the “memory” of the traces to a size l):

$$\begin{aligned}y_{k,l} &= \sum_{m=\max(1,k-l+1)}^{k}(\tilde\rho_m^{k-1})^2,\\ \mathfrak{D}_{j,l} &= \sum_{k=\max(1,j-l+1)}^{j}\tilde\rho_k^{j-1}y_{k,l}\Delta\phi_k,\end{aligned}$$

and update the following matrix:

$$\tilde{A}_{i,l} = \tilde{A}_{i-1,l} + \Delta\phi_i \Delta\phi_i^T y_{i,l} + \tilde{\rho}_{i-1}(\Delta\phi_i \mathfrak{D}_{i-1,l}^T + \mathfrak{D}_{i-1,l}\Delta\phi_i^T).$$

The assumption in Equation (15) implies that $\tilde{\rho}_i^{j-1} \leq \beta^{j-i}$, therefore it can be seen that for all k,

$$|y_{k,l} - y_k| = \sum_{m=1}^{\max(0,k-l)} (\tilde{\rho}_m^{k-1})^2 \leq \sum_{m=1}^{\max(0,k-l)} \beta^{2(k-m)} \leq \frac{\beta^{2l}}{1-\beta^2} = \epsilon_1(l)$$

where $\epsilon_1(l)$ tends to 0 when l tends to infinity. Similarly, using the fact that $y_k \leq \frac{1}{1-\beta^2}$ and writing $K = \max_{s,s'} \|\phi(s) - \gamma\phi(s')\|_\infty$, one has for all j,

$$\begin{aligned}
\|\mathfrak{D}_{j,l} - \mathfrak{D}_j\|_\infty &\leq \sum_{k=1}^{\max(0,j-l)} \tilde{\rho}_k^{j-1} \|y_k \Delta\phi_k\|_\infty + \sum_{k=\max(1,j-l+1)}^{j} \tilde{\rho}_k^{j-1} |y_{k,l} - y_k| \|\Delta\phi_k\|_\infty \\
&\leq \sum_{k=1}^{\max(0,j-l)} \tilde{\rho}_k^{j-1} \frac{1}{1-\beta^2} K + \sum_{k=\max(1,j-l+1)}^{j} \tilde{\rho}_k^{j-1} \frac{\beta^{2l}}{1-\beta^2} K \\
&\leq \frac{\beta^l}{1-\beta} \frac{1}{1-\beta^2} K + \frac{1}{1-\beta} \frac{\beta^{2l}}{1-\beta^2} K = \epsilon_2(l)
\end{aligned}$$

where $\epsilon_2(l)$ also tends to 0. Then, it can be seen that:

$$\begin{aligned}
\|\tilde{A}_{i,l} - \tilde{A}_i\|_\infty = \Big\| & \tilde{A}_{i-1,l} - \tilde{A}_{i-1} + \Delta\phi_i \Delta\phi_i^T (y_{i,l} - y_i) \\
& + \tilde{\rho}_{i-1}(\Delta\phi_i(\mathfrak{D}_{i-1,l}^T - \mathfrak{D}_{i-1}^T) + (\mathfrak{D}_{i-1,l} - \mathfrak{D}_{i-1})\Delta\phi_i^T) \Big\|_\infty \\
\leq{} & \|\tilde{A}_{i-1,l} - \tilde{A}_{i-1}\|_\infty + \|\Delta\phi_i \Delta\phi_i^T\|_\infty |y_{k,l} - y_k| + 2\beta \|\Delta\phi_i\|_\infty \|\mathfrak{D}_{i-1,l} - \mathfrak{D}_i\|_\infty \\
\leq{} & \|\tilde{A}_{i-1,l} - \tilde{A}_{i-1}\|_\infty + K^2 \epsilon_1(l) + 2\beta K \epsilon_2(l)
\end{aligned}$$

and, by a recurrence on i, one obtains

$$\left\| \frac{\tilde{A}_{i,l}}{i} - \frac{\tilde{A}_i}{i} \right\|_\infty \leq \epsilon(l)$$

where $\epsilon(l)$ tends to 0 when l tends to infinity. This implies that:

$$\liminf_{l\to\infty} \frac{\tilde{A}_{i,l}}{i} - \epsilon(l) \leq \liminf_{l\to\infty} \frac{\tilde{A}_i}{i} \leq \limsup_{l\to\infty} \frac{\tilde{A}_i}{i} \leq \limsup_{l\to\infty} \frac{\tilde{A}_{i,l}}{i} + \epsilon(l).$$

In other words, one can see that $\lim_{i\to\infty} \frac{\tilde{A}_i}{i}$ and $\lim_{l\to\infty} \lim_{i\to\infty} \frac{\tilde{A}_{i,l}}{i}$ are equal if the latter exists. In the remaining of the proof, we show that the latter limit indeed exists and we compute it explicitly.

Let us fix some l and let us consider the sequence $(\frac{\tilde{A}_{i,l}}{i})$. At some index i, $y_{i,l}$ depends only on the last l samples, while $\mathfrak{D}_{i,l}$ depends on the same samples and the last l values of $y_{j,l}$, thus on the last $2l$ samples. It is then natural to view the computation of $\tilde{A}_{i,l}$, which is based on $y_{i,l}$, $\mathfrak{D}_{i-1,l}$ and $\Delta\phi_i = \phi_i - \gamma\rho_i\phi_{i+1}$, as being related to a Markov chain of which the states are the $2l+1$ consecutive states of the original chain $(s_{i-2l}, \dots, s_i, s_{i+1})$. Write E_0 the expectation with respect to its stationary distribution. By the Markov chain Ergodic Theorem, we have with probability 1:

$$\lim_{i\to\infty} \frac{\tilde{A}_{i,l}}{i} = E_0\left[\Delta\phi_{2l}\Delta\phi_{2l}^T y_{2l,l} + \lambda\gamma\rho_{2l-1}(\Delta\phi_{2l}\mathfrak{D}_{2l-1,l}^T + \mathfrak{D}_{2l-1,l}\Delta\phi_{2l}^T)\right]. \tag{23}$$

Let us now explicitly compute this expectation. Write x_i the indicator vector (of which the k^{th} coordinate equals 1 when the state at time i is k and 0 otherwise). One has the following relations: $\phi_i = \Phi^T x_i$. Let us first look at the left part of the above limit:

$$\begin{aligned}
&E_0\left[\Delta\phi_{2l}\Delta\phi_{2l}^T y_{2l,l}\right] \\
&= E_0\left[(\phi_{2l} - \gamma\rho_{2l}\phi_{2l+1})(\phi_{2l} - \gamma\rho_{2l}\phi_{2l+1})^T y_{2l,l}\right] \\
&= E_0\left[\Phi^T(x_{2l} - \gamma\rho_{2l}x_{2l+1})(x_{2l} - \gamma\rho_{2l}x_{2l+1})^T\Phi\left(\sum_{m=l+1}^{2l}(\lambda\gamma)^{2(2l-m)}(\rho_m^{2l-1})^2\right)\right] \\
&= \Phi^T\left\{\sum_{m=l+1}^{2l}(\lambda\gamma)^{2(2l-m)}E_0\left[(\rho_m^{2l-1})^2(x_{2l} - \gamma\rho_{2l}x_{2l+1})(x_{2l} - \gamma\rho_{2l}x_{2l+1})^T\right]\right\}\Phi \\
&= \Phi^T\left\{\sum_{m=l+1}^{2l}(\lambda\gamma)^{2(2l-m)}E_0\left[(X_{m,2l,2l} - \gamma X_{m,2l,2l+1} - \gamma X_{m,2l+1,2l} + \gamma^2 X_{m,2l+1,2l+1})\right]\right\}\Phi
\end{aligned}$$

where we used the definition $\tilde{\rho}_j^{k-1} = (\lambda\gamma)^{k-j}\rho_j^{k-1}$ and the notation $X_{m,i,j} = \rho_m^{i-1}\rho_m^{j-1}x_i x_j^T$. To finish the computation, we will mainly rely on the following Lemma:

Lemma 12 (Some Identities) *Let $\tilde{P}$ be the matrix of which the coordinates are $\tilde{p}_{ss'} = \sum_a \pi(s,a)\rho(s,a)T(s,a,s')$, which is in general not a stochastic matrix. Let μ_0 be the stationary distribution of the behavior policy π_0. Write $\tilde{D}_i = \operatorname{diag}\left((\tilde{P}^T)^i\mu_0\right)$. Then*

$$\begin{aligned}
\forall m \le i,\ & E_0[X_{m,i,i}] = \tilde{D}_{i-m}, \\
\forall m \le i \le j,\ & E_0[X_{m,i,j}] = \tilde{D}_{i-m}P^{j-i}, \\
\forall m \le j \le i,\ & E_0[X_{m,i,j}] = (P^T)^{j-i}\tilde{D}_{i-m}.
\end{aligned}$$

Proof We first observe that:

$$\begin{aligned}
E_0[X_{m,i,i}] &= E_0[(\rho_m^{i-1})^2 x_i x_i^T] \\
&= E_0[(\rho_m^{i-1})^2 \operatorname{diag}(x_i)] \\
&= \operatorname{diag}\left(E_0[(\rho_m^{i-1})^2 x_i\right).
\end{aligned}$$

To provide the identity, we will thus simply provide a proof by recurrence that $E_0[(\rho_m^{i-1})^2 x_i] = (\tilde{P}^T)^{m-i}\mu_0$. For $i = m$, we have $E_0[x_m] = \mu_0$. Now suppose the relation holds for i and let us prove it for $i+1$.

$$\begin{aligned}
E_0[(\rho_m^i)^2 x_{i+1}] &= E_0\left[E_0[(\rho_m^i)^2 x_{i+1}|\mathcal{F}_i]\right] \\
&= E_0\left[E_0[(\rho_m^{i-1})^2(\rho_i)^2 x_{i+1}|\mathcal{F}_i]\right] \\
&= E_0\left[(\rho_m^{i-1})^2 E_0[(\rho_i)^2 x_{i+1}|\mathcal{F}_i]\right].
\end{aligned}$$

Write $\mathcal{F}_i$ the realization of the process until time i. Recalling that s_i is the state at time i and x_i is the indicator vector corresponding to s_i, one has for all s':

$$\begin{aligned} E_0[(\rho_i)^2 x_{i+1}(s')|\mathcal{F}_i] &= \sum_a \pi_0(s_i,a)\rho(s_i,a)^2 T(s_i,a,s') \\ &= \sum_a \pi(s_i,a)\rho(s_i,a)T(s_i,a,s') \\ &= \tilde{p}_{s_i,s'} \\ &= [\tilde{P}^T x_i](s'). \end{aligned}$$

As this is true for all s', we deduce that $E_0[(\rho_i)^2 x_{i+1}|\mathcal{F}_i] = \tilde{P}^T x_i$ and

$$\begin{aligned} E_0[(\rho_m^i)^2 x_{i+1}] &= E_0[(\rho_m^{i-1})^2 \tilde{P}^T x_i] \\ &= \tilde{P}^T E_0[(\rho_m^{i-1})^2 \tilde{P}^T x_i] \\ &= \tilde{P}^T (\tilde{P}^T)^i \mu_0 \\ &= (\tilde{P}^T)^{i+1}\mu_0 \end{aligned}$$

which concludes the proof by recurrence.

Let us consider the next identity. For $i \leq j$,

$$\begin{aligned} E_0[\rho_m^{i-1}\rho_m^{j-1} x_i x_j^T] &= E_0[E_0[\rho_m^{i-1}\rho_m^{j-1} x_i x_j^T|\mathcal{F}_i]] \\ &= E_0[(\rho_m^{i-1})^2 x_i E_0[\rho_i^{j-1} x_j^T|\mathcal{F}_i]] \\ &= E_0[(\rho_m^{i-1})^2 x_i x_i^T P^{j-i}] \\ &= \operatorname{diag}\left((\tilde{P}^T)^{m-i}\mu_0\right) P^{j-i}. \end{aligned}$$

Eventually, the last identity is obtained by considering $Y_{m,i,j} = X_{m,j,i}^T$. ■

Thus, coming back to our calculus,

$$E_0\left[\Delta\phi_{2l}\Delta\phi_{2l}^T y_{2l,l}\right] = \Phi^T\left\{\sum_{m=l+1}^{2l}(\lambda\gamma)^{2(2l-m)}\left(\tilde{D}_{2l-m} - \gamma\tilde{D}_{2l-m}P - \gamma P^T\tilde{D}_{2l-m} + \gamma^2\tilde{D}_{2l+1-m}\right)\right\}\Phi$$

$$= \Phi^T(D_l - \gamma D_l P - \gamma P^T D_l + \gamma^2 D_l')\Phi \tag{24}$$

$$\text{with}\quad D_l = \sum_{j=0}^{l-1}(\lambda\gamma)^{2j}\tilde{D}_j, \quad\text{and}\quad D_l' = \sum_{j=0}^{l-1}(\lambda\gamma)^{2j}\tilde{D}_{j+1}.$$

Similarly, the second term on the right side of Equation (23) satisfies:

$$
\begin{aligned}
&E_0\left[\rho_{2l-1}\mathfrak{D}_{2l-1,l}\Delta\phi_{2l}^T\right]\\
&= E_0\left[\rho_{2l-1}\sum_{k=l}^{2l-1}\tilde{\rho}_k^{2l-2}y_{k,l}\Delta\phi_k\Delta\phi_{2l}^T\right]\\
&= E_0\left[\sum_{k=l}^{2l-1}(\lambda\gamma)^{2l-1-k}\rho_k^{2l-1}\left(\sum_{m=k-l+1}^{k}(\tilde{\rho}_m^{k-1})^2\right)\Phi^T(x_k-\gamma\rho_k x_{k+1})(x_{2l}-\gamma\rho_{2l}x_{2l+1})^T\Phi\Delta\phi_{2l}^T\right]\\
&= \Phi^T\Bigg(\sum_{k=l}^{2l-1}(\lambda\gamma)^{2l-1-k}\\
&\qquad\sum_{m=k-l+1}^{k}(\lambda\gamma)^{2(k-m)}E_0\left[\rho_m^{2l-1}\rho_m^{k-1}(x_k-\gamma\rho_k x_{k+1})(x_{2l}-\gamma\rho_{2l}x_{2l+1})^T\right]\Bigg)\Phi\\
&= \Phi^T\Bigg(\sum_{k=l}^{2l-1}(\lambda\gamma)^{2l-1-k}\sum_{m=k-l+1}^{k}(\lambda\gamma)^{2(k-m)}\\
&\qquad E_0\left[X_{m,k,2l}-\gamma X_{m,k+1,2l}-\gamma X_{m,k,2l+1}+\gamma^2 X_{m,k+1,2l+1}\right]\Bigg)\Phi\\
&= \Phi^T\Bigg(\sum_{k=l}^{2l-1}(\lambda\gamma)^{2l-1-k}\sum_{m=k-l+1}^{k}(\lambda\gamma)^{2(k-m)}\\
&\qquad\left(\tilde{D}_{k-m}P^{2l-k}-\gamma\tilde{D}_{k+1-m}P^{2l-k-1}-\gamma\tilde{D}_{k-m}P^{2l+1-k}+\gamma^2\tilde{D}_{k+1-m}P^{2l-k}\right)\Bigg)\Phi\\
&= \Phi^T\Bigg(\sum_{k=l}^{2l-1}(\lambda\gamma)^{2l-1-k}\sum_{m=k-l+1}^{k}(\lambda\gamma)^{2(k-m)}\\
&\qquad\left(\tilde{D}_{k-m}P^{2l-k}(I-\gamma P)-\gamma\tilde{D}_{k+1-m}P^{2l-1-k}(I-\gamma P)\right)\Bigg)\Phi\\
&= \Phi^T\left(\sum_{k=l}^{2l-1}(\lambda\gamma)^{2l-1-k}\sum_{m=k-l+1}^{k}(\lambda\gamma)^{2(k-m)}\left(\tilde{D}_{k-m}P-\gamma\tilde{D}_{k+1-m}\right)P^{2l-1-k}(I-\gamma P)\right)\Phi\\
&= \Phi^T\left(\sum_{k=l}^{2l-1}(\lambda\gamma)^{2l-1-k}\left(D_lP-\gamma D_l'\right)P^{2l-1-k}(I-\gamma P)\right)\Phi\\
&= \Phi^T\left(D_lP-\gamma D_l'\right)Q_l(I-\gamma P)\Phi
\end{aligned}
$$

with $Q_l=\sum_{j=0}^{l-1}(\lambda\gamma P)^j$.

Gathering this and Equation (24), we see that the limit of $\frac{A_{i,l}}{i}$ expressed in Equation (23) equals:

$$\Phi^T \big[D_l - \gamma D_l P - \gamma P^T D_l + \gamma^2 D'_l \\ +\lambda\gamma \left((D_l P - \gamma D'_l) Q_l (I - \gamma P) + (I - \gamma P^T) Q_l^T (P^T D_l - \gamma D'_l)\right)\big] \Phi.$$

When l tends to infinity, Q_l tends to $Q = (I - \lambda\gamma P)^{-1}$. The assumption of Equation (15) ensures that $(\lambda\gamma)\tilde{P}$ has spectral radius smaller than 1, and thus when l tends to infinity, D_l tends to $D = \text{diag}\left((I - (\lambda\gamma)^2 \tilde{P}^T)^{-1}\mu_0\right)$ and D'_l to $D' = \text{diag}\left(\tilde{P}^T (I - (\lambda\gamma)^2 \tilde{P}^T)^{-1}\mu_0\right)$. In other words, $\lim_{l\to\infty} \lim_{i\to\infty} \frac{\tilde{A}_{i,l}}{i}$ exists with probability 1 and equals:

$$\Phi^T \big[D - \gamma D P - \gamma P^T D + \gamma^2 D' \\ +\lambda\gamma \left((D P - \gamma D') Q (I - \gamma P) + (I - \gamma P^T) Q^T (P^T D - \gamma D')\right)\big] \Phi.$$

Eventually, this shows that $\lim_{i\to\infty} \frac{\tilde{A}_i}{i}$ exists with probability 1 and shares the same value.

A similar reasoning allows to show that $\lim_{i\to\infty} \frac{\tilde{b}_i}{i}$ exists and equals

$$\Phi^T \left[(I - \gamma P^T) Q^T D + \lambda\gamma (D P - \gamma D') Q\right] R^\pi. \quad \blacksquare$$

Appendix C. Proof Of Proposition 9

To prove Proposition 9, we need the following technical lemma.

Lemma 13 *Let α_i and β_i be two forward recursions defined as*

$$\alpha_i = a_i + \eta_i \alpha_{i+1}$$
$$\text{and } \beta_i = b_i + \eta_i \beta_{i+1}.$$

Assume that for any function f we have that (this is typically true if the index i refers to a state sampled according to some stationary distribution, which is the case we are interested in)

$$E[f(a_i, b_i, \eta_i)] = E[f(a_{i-1}, b_{i-1}, \eta_{i-1})].$$

Let also u_i, v_i and w_i be the backward recursions defined as

$$\begin{aligned} w_i &= 1 + \eta_{i-1}^2 w_{i-1}, \\ u_i &= a_i w_i + \eta_{i-1} u_{i-1}, \\ v_i &= b_i w_i + \eta_{i-1} v_{i-1}. \end{aligned}$$

Then, we have

$$E[\alpha_i \beta_i] = E[a_i v_i + b_i u_i - a_i b_i w_i].$$

Proof The proof looks like the one of Proposition 6, but is a little bit more complicated. A key equality, to be applied repeatedly, is:

$$\begin{aligned} \alpha_i \beta_i &= (a_i + \eta_i \alpha_{i+1})(b_i + \eta_i \beta_{i+1}) \\ &= a_i \beta_i + b_i \alpha_i + \eta_i^2 \alpha_{i+1} \beta_{i+1} - a_i b_i. \end{aligned}$$

Another equality to be used repeatedly makes use of the "stationarity" assumption. For any $k \geq 0$ we have:

$$E[(\prod_{j=0}^{k} \eta_{i-j}^2)\alpha_{i+1}\beta_{i+1}] = E[(\prod_{j=1}^{k+1} \eta_{i-j}^2)\alpha_i\beta_i].$$

These two identities can be used to work the term of interest:

$$\begin{aligned} E[\alpha_i\beta_i] &= E[(a_i + \eta_i\alpha_{i+1})(b_i + \eta_i\beta_{i+1})] \\ &= E[a_i\beta_i] + E[b_i\alpha_i] + E[\eta_i^2\alpha_{i+1}\beta_{i+1}] - E[a_ib_i] \\ &= E[a_i\beta_i] + E[b_i\alpha_i] - E[a_ib_i] + E[\eta_{i-1}^2(a_i + \eta_i\alpha_{i+1})(b_i + \eta_i\beta_{i+1})] \\ &= E[a_i(1 + \eta_{i-1}^2)\beta_i] + E[b_i(1 + \eta_{i-1}^2)\alpha_i] - E[a_ib_i(1 + \eta_{i-1}^2)] + E[(\eta_{i-1}\eta_i)^2\alpha_{i+1}\beta_{i+1}]. \end{aligned}$$

This process can be repeated, giving

$$E[\alpha_i\beta_i] = E[(a_i\beta_i + b_i\alpha_i - a_ib_i)(1 + \eta_{i-1}^2 + (\eta_{i-1}\eta_{i-2})^2 + \dots)].$$

We have that

$$w_i = 1 + \eta_{i-1}^2 w_{i-1} = 1 + \eta_{i-1}^2 + (\eta_{i-1}\eta_{i-2})^2 + \dots,$$

therefore

$$E[\alpha_i\beta_i] = E[a_iw_i\beta_i] + E[b_iw_i\alpha_i] - E[a_ib_iw_i].$$

We can work on the first term:

$$\begin{aligned} E[a_iw_i\beta_i] &= E[a_iw_i(b_i + \eta_i\beta_{i+1})] \\ &= E[a_iw_ib_i] + E[a_{i-1}w_{i-1}\eta_{i-1}(b_i + \eta_i\beta_{i+1})] \\ &= E[b_i(a_iw_i + \eta_{i-1}(a_{i-1}w_{i-1}) + \eta_{i-1}\eta_{i-2}(a_{i-2}w_{i-2}) + \dots)] \\ &= E[b_iu_i]. \end{aligned}$$

The work on the second term is symmetric:

$$E[b_iw_i\alpha_i] = E[a_iv_i].$$

This finishes proving the result. ■

The proof of Proposition 9 is a simple application of the preceding technical lemma. By lemma 5, we have that

$$\underbrace{\delta_i^\lambda}_{\doteq\alpha_i} = \underbrace{\delta_i}_{\doteq a_i} + \underbrace{\gamma\lambda\rho_i}_{\doteq\eta_i} \underbrace{\delta_{i+1}^\lambda}_{\doteq\alpha_{i+1}}.$$

By lemma 7, we have that

$$\underbrace{g_i^\lambda}_{\doteq\beta_i} = \underbrace{\gamma\rho_i(1-\lambda)\phi_{i+1}}_{\doteq b_i} + \underbrace{\gamma\lambda\rho_i}_{\doteq\eta_i} \underbrace{g_{i+1}^\lambda}_{\doteq\beta_{i+1}}.$$

The result is then a direct application of Lemma 13.

References

A. Antos, Cs. Szepesvári, and R. Munos. Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. In *Conference on Learning Theory (COLT)*, 2006.

T. Archibald, K. McKinnon, and L. Thomas. On the generation of Markov decision processes. *Journal of the Operational Research Society*, 46:354–361, 1995.

L.C. Baird. Residual algorithms: Reinforcement learning with function approximation. In *International Conference on Machine Learning (ICML)*, 1995.

D.P. Bertsekas and S. Ioffe. Temporal differences-based policy iteration and applications in neuro-dynamic programming. Technical report, MIT, 1996.

D.P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, 1996.

D.P. Bertsekas and H. Yu. Projected equation methods for approximate solution of large linear systems. *Journal of Computational and Applied Mathematics*, 227:27–50, 2009.

L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In S. Sra, S. Nowozin, and S.J. Wright, editors, *Optimization for Machine Learning*, pages 351–368. MIT Press, 2011.

J.A. Boyan. Technical update: Least-squares temporal difference learning. *Machine Learning*, 49(2-3):233–246, 1999.

S.J. Bradtke and A.G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1-3):33–57, 1996.

D. Choi and B. Van Roy. A generalized Kalman filter for fixed point approximation and efficient temporal-difference learning. *Discrete Event Dynamic Systems*, 16:207–239, 2006.

Y. Engel. *Algorithms and Representations for Reinforcement Learning.* PhD thesis, Hebrew University, 2005.

M. Geist and O. Pietquin. Eligibility traces through colored noises. In *IEEE International Conference on Ultra Modern Control Systems (ICUMT)*, 2010a.

M. Geist and O. Pietquin. Kalman temporal differences. *Journal of Artifical Intelligence Research (JAIR)*, 39:483–532, 2010b.

M. Geist and O. Pietquin. Algorithmic survey of parametric value function approximation. *IEEE Transactions on Neural Networks and Learning Systems*, 24(6):845 – 867, 2013.

M. Kearns and S. Singh. Bias-variance error bounds for temporal difference updates. In *Conference on Learning Theory (COLT)*, 2000.

J.Z. Kolter. The fixed points of off-policy TD. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

M.G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003. ISSN 1533-7928.

H.R. Maei and R.S. Sutton. GQ(λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In *Conference on Artificial General Intelligence (AGI)*, 2010.

R. Munos. Error bounds for approximate policy iteration. In *International Conference on Machine Learning (ICML)*, 2003.

A. Nedić and D.P. Bertsekas. Least squares policy evaluation algorithms with linear function approximation. *Discrete Event Dynamic Systems*, 13:79–110, 2003.

D. Precup, R.S. Sutton, and S.P. Singh. Eligibility traces for off-policy policy evaluation. In *International Conference on Machine Learning (ICML)*, 2000.

D. Precup, R.S. Sutton, and S. Dasgupta. Off-policy temporal-difference learning with function approximation. In *International Conference on Machine Learning (ICML)*, 2001.

R.S. Randhawa and S. Juneja. Combining importance sampling and temporal difference control variates to simulate Markov chains. *ACM Transactions on Modeling and Computer Simulation*, 14(1):1–30, 2004.

B.D. Ripley. *Stochastic Simulation.* Wiley & Sons, 1987.

B. Scherrer. Should one compute the temporal difference fix point or minimize the Bellman residual? The unified oblique projection view. In *International Conference on Machine Learning (ICML)*, 2010.

B. Scherrer and M. Geist. Recursive least-squares learning with eligibility traces. In *European Workshop on Reinforcement Learning (EWRL)*, 2011.

R. Schoknecht. Optimality of reinforcement learning algorithms with linear function approximation. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning).* MIT Press, 3rd edition, 1998.

R.S. Sutton, H.R. Maei, D. Precup, S. Bhatnagar, D. Silver, Cs. Szepesvári, and E. Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *International Conference on Machine Learning (ICML)*, 2009.

J.N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, 1997.

H. Yu. Convergence of least-squares temporal difference methods under general conditions. In *International Conference on Machine Learning (ICML)*, 2010a.

H. Yu. Least squares temporal difference methods: An analysis under general condtions. Technical Report C-2010-39, University of Helsinki, September 2010b.

H. Yu and D.P. Bertsekas. New error bounds for approximations from projected linear equations. Technical Report C-2008-43, Dept. Computer Science, Univ. of Helsinki, July 2008.