Least-Squares Methods for Policy Iteration

Lucian Buşoniu, Alessandro Lazaric, Mohammad Ghavamzadeh, Rémi Munos, Robert Babuška, and Bart De Schutter

Abstract Approximate reinforcement learning deals with the essential problem of applying reinforcement learning in large and continuous state-action spaces, by using function approximators to represent the solution. This chapter reviews least-squares methods for policy iteration, an important class of algorithms for approximate reinforcement learning. We discuss three techniques for solving the core, policy evaluation component of policy iteration, called: least-squares temporal difference, least-squares policy evaluation, and Bellman residual minimization. We introduce these techniques starting from their general mathematical principles and detailing them down to fully specified algorithms. We pay attention to online variants of policy iteration, and provide a numerical example highlighting the behavior of representative offline and online methods. For the policy evaluation component as well as for the overall resulting approximate policy iteration, we provide guarantees on the performance obtained asymptotically, as the number of samples processed and iterations executed grows to infinity. We also provide finite-sample results, which apply when a finite number of samples and iterations are considered. Finally, we outline several extensions and improvements to the techniques and methods reviewed.

Lucian Buşoniu, Alessandro Lazaric, Mohammad Ghavamzadeh, Rémi Munos
Team SequeL, INRIA Lille-Nord Europe, France,
{ion-lucian.busoniu,alessandro.lazaric, mohammad.ghavamzadeh, remi.munos}@inria.fr
Robert Babuška, Bart De Schutter
Delft Center for Systems and Control, Delft University of Technology, The Netherlands,
{r.babuska, b.deschutter}@tudelft.nl
This work was performed in part while Lucian Buşoniu was with the Delft Center for Systems and Control.

1 Introduction

Policy iteration is a core procedure for solving reinforcement learning problems, which evaluates policies by estimating their value functions, and then uses these value functions to find new, improved policies. In Chapter 2, the classical policy iteration was introduced, which employs tabular, exact representations of the value functions and policies. However, most problems of practical interest have state and action spaces with a very large or even infinite number of elements, which precludes tabular representations and exact policy iteration. Instead, *approximate policy iteration* must be used. In particular, approximate policy evaluation – constructing approximate value functions for the policies considered – is the central, most challenging component of approximate policy iteration. While representing the policy can also be challenging, an explicit representation is often avoided, by computing policy actions on-demand from the approximate value function.

Some of the most powerful state-of-the-art algorithms for approximate policy evaluation represent the value function using a linear parameterization, and obtain a linear system of equations in the parameters, by exploiting the linearity of the Bellman equation satisfied by the value function. Then, in order to obtain parameters approximating the value function, this system is solved in a least-squares sample-based sense, either in one shot or iteratively.

Since highly efficient numerical methods are available to solve such systems, least-squares methods for policy evaluation are computationally efficient. Additionally taking advantage of the generally fast convergence of policy iteration methods, an overall fast policy iteration algorithm is obtained. More importantly, least-squares methods are sample-efficient, i.e., they approach their solution quickly as the number of samples they consider increases This is a crucial property in reinforcement learning for real-life systems, as data obtained from such systems are very expensive (in terms of time taken for designing and running data collection experiments, of system wear-and-tear, and possibly even of economic costs).

In this chapter, we review least-squares methods for policy iteration: the class of approximate policy iteration methods that employ least-squares techniques at the policy evaluation step. The review is organized as follows. Section 2 provides a quick recapitulation of classical policy iteration, and also serves to further clarify the technical focus of the chapter. Section 3 forms the chapter's core, thoroughly introducing the family of least-squares methods for policy evaluation. Section 4 describes an application of a particular algorithm, called simply least-squares policy iteration, to online learning. Section 5 illustrates the behavior of offline and online least-squares policy iteration in an example. Section 6 reviews available theoretical guarantees about least-squares policy evaluation and the resulting, approximate policy iteration. Finally, Section 7 outlines several important extensions and improvements to least-squares methods, and mentions other reviews of reinforcement learning that provide different perspectives on least-squares methods.

2 Preliminaries: Classical Policy Iteration

In this section, we revisit the classical policy iteration algorithm and some relevant theoretical results from Chapter 2, adapting their presentation for the purposes of the present chapter.

Recall first some notation. A Markov decision process with states $s \in S$ and actions $a \in A$ is given, governed by the stochastic dynamics $s' \sim T(s,a,\cdot)$ and with the immediate performance described by the rewards $r = R(s,a,s')$, where T is the transition function and R the reward function. The goal is to find an optimal policy $\pi^* : S \to A$ that maximizes the value function $V^\pi(s)$ or $Q^\pi(s,a)$. For clarity, in this chapter we refer to state value functions V as "V-functions", thus achieving consistency with the name "Q-functions" traditionally used for state-action value functions Q. We use the name "value function" to refer collectively to V-functions and Q-functions.

Policy iteration works by iteratively evaluating and improving policies. At the *policy evaluation* step, the V-function or Q-function of the current policy is found. Then, at the *policy improvement* step, a new, better policy is computed based on this V-function or Q-function. The procedure continues afterward with the next iteration. When the state-action space is finite and exact representations of the value function and policy are used, policy improvement obtains a strictly better policy than the previous one, unless the policy is already optimal. Since additionally the number of possible policies is finite, policy iteration is guaranteed to find the optimal policy in a finite number of iterations. Algorithm 1 shows the classical, offline policy iteration for the case when Q-functions are used.

1: **input** initial policy π_0
2: $k \leftarrow 0$
3: **repeat**
4: find Q^{π_k} {policy evaluation}
5: $\pi_{k+1}(s) \leftarrow \arg\max_{a \in A} Q^{\pi_k}(s,a) \quad \forall s$ {policy improvement}
6: $k \leftarrow k+1$
7: **until** $\pi_k = \pi_{k-1}$
8: **output** $\pi^* = \pi_k$, $Q^* = Q^{\pi_k}$

Algorithm 1: Policy iteration with Q-functions.

At the policy evaluation step, the Q-function Q^π of policy π can be found using the fact that it satisfies the Bellman equation:

$$Q^\pi = B_Q^\pi(Q^\pi) \tag{1}$$

where the Bellman mapping (also called backup operator) B_Q^π is defined for any Q-function as follows:

$$[B_Q^\pi(Q)](s,a) = \mathrm{E}_{s' \sim T(s,a,\cdot)} \left\{ R(s,a,s') + \gamma Q(s', \pi(s')) \right\} \tag{2}$$

Similarly, the V-function V^π of policy π satisfies the Bellman equation:

$$V^\pi = B_V^\pi(V^\pi) \tag{3}$$

where the Bellman mapping B_V^π is defined by:

$$[B_V^\pi(V)](s) = \mathrm{E}_{s' \sim T(s,\pi(s),\cdot)} \left\{ R(s,\pi(s),s') + \gamma V(s') \right\} \tag{4}$$

Note that both the Q-function and the V-function are bounded in absolute value by $V_{\max} = \frac{\|R\|_\infty}{1-\gamma}$, where $\|R\|_\infty$ is the maximum absolute reward. A number of algorithms are available for computing Q^π or V^π, based, e.g., on directly solving the linear system of equations (1) or (3) to obtain the Q-values (or V-values), on turning the Bellman equation into an iterative assignment, or on temporal-difference, model-free estimation – see Chapter 2 for details.

Once Q^π or V^π is available, policy improvement can be performed. In this context, an important difference between using Q-functions and V-functions arises. When Q-functions are used, policy improvement involves only a maximization over the action space:

$$\pi_{k+1}(s) \leftarrow \arg\max_{a \in A} Q^{\pi_k}(s,a) \tag{5}$$

(see again Algorithm 1), whereas policy improvement with V-functions additionally requires a model, in the form of T and R, to investigate the transitions generated by each action:

$$\pi_{k+1}(s) \leftarrow \arg\max_{a \in A} \mathrm{E}_{s' \sim T(s,a,\cdot)} \left\{ R(s,a,s') + \gamma V(s') \right\} \tag{6}$$

This is an important point in favor of using Q-functions in practice, especially for model-free algorithms.

Classical policy iteration requires exact representations of the value functions, which can generally only be achieved by storing distinct values for every state-action pair (in the case of Q-functions) or for every state (V-functions). When some of the variables have a very large or infinite number of possible values, e.g., when they are continuous, exact representations become impossible and value functions must be *approximated*. This chapter focuses on *approximate policy iteration*, and more specifically, on algorithms for approximate policy iteration that use a class of *least-squares* methods for policy evaluation.

While representing policies in large state spaces is also challenging, an explicit representation of the policy can fortunately often be avoided. Instead, improved policies can be computed on-demand, by applying (5) or (6) at every state where an action is needed. This means that an implicit policy representation – via the value function – is used. In the sequel, we focus on this setting, additionally requiring that *exact* policy improvements are performed, i.e., that the action returned is always an exact maximizer in (5) or (6). This requirement can be satisfied, e.g., when the action space is discrete and contains not too large a number of actions. In this case,

policy improvement can be performed by computing the value function for all the discrete actions, and finding the maximum among these values using enumeration.[1]

In what follows, wherever possible, we will introduce the results and algorithms in terms of Q-functions, motivated by the practical advantages they provide in the context of policy improvement. However, most of these results and algorithms directly extend to the case of V-functions.

3 Least-Squares Methods for Approximate Policy Evaluation

In this section we consider the problem of policy evaluation, and we introduce least-squares methods to solve this problem. First, in Section 3.1, we discuss the high-level principles behind these methods. Then, we progressively move towards the methods' practical implementation. In particular, in Section 3.2 we derive idealized, model-based versions of the algorithms for the case of linearly parameterized approximation, and in Section 3.3 we outline their realistic, model-free implementations. To avoid detracting from the main line, we postpone the discussion of most literature references until Section 3.4.

3.1 Main Principles and Taxonomy

In problems with large or continuous state-action spaces, value functions cannot be represented exactly, but must be approximated. Since the solution of the Bellman equation (1) will typically not be representable by the chosen approximator, the Bellman equation must be solved approximately instead. Two main classes of least-squares methods for policy evaluation can be distinguished by the approach they take to approximately solve the Bellman equation, as shown in Figure 1: projected policy evaluation, and Bellman residual minimization.

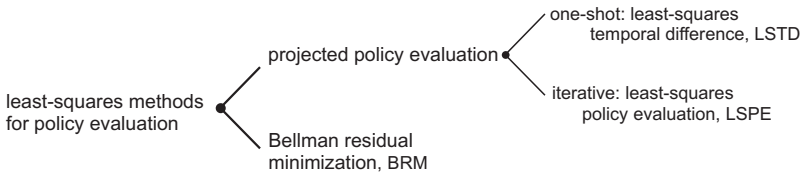

Fig. 1 Taxonomy of the methods covered.

[1] In general, when the action space is large or continuous, the maximization problems (5) or (6) may be too involved to solve exactly, in which case only *approximate* policy improvements can be made.

Methods for *projected policy evaluation* look for a $\widehat{Q}$ that is approximately equal to the projection of its backed-up version $B_Q^\pi(\widehat{Q})$ on the space of representable Q-functions:

$$\widehat{Q} \approx \Pi(B_Q^\pi(\widehat{Q})) \tag{7}$$

where $\Pi : \mathscr{Q} \to \widehat{\mathscr{Q}}$ denotes projection from the space $\mathscr{Q}$ of all Q-functions onto the space $\widehat{\mathscr{Q}}$ of representable Q-functions. Solving this equation is the same as minimizing the distance between $\widehat{Q}$ and $\Pi(B_Q^\pi(\widehat{Q}))$: [2]

$$\min_{\widehat{Q}\in\widehat{\mathscr{Q}}} \left\| \widehat{Q} - \Pi(B_Q^\pi(\widehat{Q})) \right\| \tag{8}$$

where $\|\cdot\|$ denotes a generic norm / measure of magnitude. An example of such a norm will be given in Section 3.2. The relation (7) is called the *projected Bellman equation*, hence the name projected policy evaluation. As will be seen in Section 3.2, in the case of linear approximation (7) can in fact be solved exactly, i.e. with equality (whereas in general, there may not exist a function $\widehat{Q}$ that when passed through B_Q^π and Π leads exactly back to itself).

Two subclasses of methods employing the principle of projected policy evaluation can be further identified: *one-shot* methods that aim directly for a solution to the projected Bellman equation; and methods that find a solution in an *iterative* fashion. Such methods are called, respectively, least-squares temporal difference (LSTD) (Bradtke and Barto, 1996; Boyan, 2002) and least-squares policy evaluation (LSPE) (Bertsekas and Ioffe, 1996) in the literature. Note that the LSPE iterations are internal to the policy evaluation step, contained within the larger iterations of the overall policy iteration algorithm.

Methods for *Bellman residual minimization* (BRM) do not employ projection, but try to directly solve the Bellman equation in an approximate sense:

$$\widehat{Q} \approx B_Q^\pi(\widehat{Q})$$

by performing the minimization:

$$\min_{\widehat{Q}\in\widehat{\mathscr{Q}}} \left\| \widehat{Q} - B_Q^\pi(\widehat{Q}) \right\| \tag{9}$$

The difference $\widehat{Q} - B_Q^\pi(\widehat{Q})$ is called Bellman residual, hence the name BRM.

In the sequel, we will append the suffix "Q" to the acronym of the methods that find Q-functions (e.g., LSTD-Q, LSPE-Q, BRM-Q), and the suffix "V" in the case of V-functions. We will use the unqualified acronym to refer to both types of methods collectively, when the distinction between them is not important.

[2] In this equation as well as the others, a multistep version of the Bellman mapping B_Q^π (or B_V^π) can also be used, parameterized by a scalar $\lambda \in [0,1)$. In this chapter, we mainly consider the single-step case, for $\lambda = 0$, but we briefly discuss the multistep case in Section 7.

3.2 The Linear Case and Matrix Form of the Equations

Until now, the approximator of the value function, as well as the norms in the minimizations (8) and (9) have been left unspecified. The most common choices are, respectively, approximators linear in the parameters and (squared, weighted) Euclidean norms, as defined below. With these choices, the minimization problems solved by projected and BRM policy evaluation can be written in terms of matrices and vectors, and have closed-form solutions that eventually lead to efficient algorithms. The name "least-squares" for this class of methods comes from minimizing the squared Euclidean norm.

To formally introduce these choices and the solutions they lead to, the state and action spaces will be assumed finite, $S = \{s_1, \ldots, s_N\}$, $A = \{a_1, \ldots, a_M\}$. Nevertheless, the final, practical algorithms obtained in Section 3.3 below can also be applied to infinite and continuous state-action spaces.

A linearly parameterized representation of the Q-function has the following form:

$$\widehat{Q}(s,a) = \sum_{l=1}^{d} \varphi_l(s,a)\theta_l = \phi^\top(s,a)\theta \tag{10}$$

where $\theta \in \mathbb{R}^d$ is the parameter vector and $\phi(s,a) = [\varphi_1(s,a), \ldots, \varphi_d(s,a)]^\top$ is a vector of basis functions (BFs), also known as features (Bertsekas and Tsitsiklis, 1996).[3] Thus, the space of representable Q-functions is the span of the BFs, $\widehat{\mathcal{Q}} = \left\{\phi^\top(\cdot,\cdot)\theta \;\middle|\; \theta \in \mathbb{R}^d\right\}$.

Given a weight function $\rho : S \times A \to [0,1]$, the (squared) weighted Euclidean norm of a Q-function is defined by:

$$\|Q\|_\rho^2 = \sum_{\substack{i=1,\ldots,N \\ j=1,\ldots,M}} \rho(s_i,a_j)\left|Q(s_i,a_j)\right|^2$$

Note the norm itself, $\|Q\|_\rho$, is the square root of this expression; we will mostly use the squared variant in the sequel.

The corresponding weighted least-squares projection operator, used in projected policy evaluation, is:

$$\Pi^\rho(Q) = \arg\min_{\widehat{Q}} \left\|\widehat{Q} - Q\right\|_\rho^2$$

The weight function is interpreted as a probability distribution over the state-action space, so it must sum up to 1. The distribution given by ρ will be used to generate samples used by the model-free policy evaluation algorithms of Section 3.3 below.

[3] Throughout the chapter, column vectors are employed.

Matrix Form of the Bellman Mapping

To specialize projection-based and BRM methods to the linear approximation, Euclidean-norm case, we will write the Bellman mapping (2) in a matrix form, and then in terms of the parameter vector.

Because the state-action space is discrete, the Bellman mapping can be written as a sum:

$$[B_Q^\pi(Q)](s_i,a_j) = \sum_{i'=1}^{N} T(s_i,a_j,s_{i'})\left[R(s_i,a_j,s_{i'}) + \gamma Q(s_{i'},\pi(s_{i'}))\right] \tag{11}$$

$$= \sum_{i'=1}^{N} T(s_i,a_j,s_{i'})R(s_i,a_j,s_{i'}) + \gamma \sum_{i'=1}^{N} T(s_i,a_j,s_{i'})Q(s_{i'},\pi(s_{i'})) \tag{12}$$

for any i, j. The two-sum expression leads us to a matrix form of this mapping:

$$\boldsymbol{B}_Q^\pi(\boldsymbol{Q}) = \boldsymbol{R} + \gamma \boldsymbol{T}^\pi \boldsymbol{Q} \tag{13}$$

where $\boldsymbol{B}_Q^\pi : \mathbb{R}^{NM} \to \mathbb{R}^{NM}$. Denoting by $[i,j] = i + (j-1)N$ the scalar index corresponding to i and j,[4] the vectors and matrices on the right hand side of (13) are defined as follows:[5]

- $\boldsymbol{Q} \in \mathbb{R}^{NM}$ is a vector representation of the Q-function Q, with $\boldsymbol{Q}_{[i,j]} = Q(s_i,a_j)$.
- $\boldsymbol{R} \in \mathbb{R}^{NM}$ is a vector representation of the *expectation* of the reward function R, where the element $\boldsymbol{R}_{[i,j]}$ is the expected reward after taking action a_j in state s_i, i.e., $\boldsymbol{R}_{[i,j]} = \sum_{i'=1}^{N} T(s_i,a_j,s_{i'})R(s_i,a_j,s_{i'})$. So, for the first sum in (12), the transition function T has been integrated into $\boldsymbol{R}$ (unlike for the second sum).
- $\boldsymbol{T}^\pi \in \mathbb{R}^{NM \times NM}$ is a matrix representation of the transition function combined with the policy, with $\boldsymbol{T}_{[i,j],[i',j']} = T(s_i,a_j,s_{i'})$ if $\pi(s_{i'}) = a_{j'}$, and 0 otherwise. A useful way to think about $\boldsymbol{T}^\pi$ is as containing transition probabilities between *state-action pairs*, rather than just states. In this interpretation, $\boldsymbol{T}^\pi_{[i,j],[i',j']}$ is the probability of moving from an arbitrary state-action pair (s_i,a_j) to a next state-action pair $(s_{i'},a_{j'})$ that follows the current policy. Thus, if $a_{j'} \neq \pi(s_{i'})$, then the probability is zero, which is what the definition says. This interpretation also indicates that stochastic policies can be represented with a simple modification: in that case, $\boldsymbol{T}^\pi_{[i,j],[i',j']} = T(s_i,a_j,s_{i'}) \cdot \pi(s_{i'},a_{j'})$, where $\pi(s,a)$ is the probability of taking a in s.

The next step is to rewrite the Bellman mapping in terms of the parameter vector, by replacing the generic Q-vector in (13) by an approximate, parameterized Q-vector. This is useful because in all the methods, the Bellman mapping is always

[4] If the d elements of the BF vector were arranged into an $N \times M$ matrix, by first filling in the first column with the first N elements, then the second column with the subsequent N elements, etc., then the element at index $[i,j]$ of the vector would be placed at row i and column j of the matrix.

[5] Boldface notation is used for vector or matrix representations of functions and mappings. Ordinary vectors and matrices are displayed in normal font.

applied to approximate Q-functions. Using the following matrix representation of the BFs:

$$\boldsymbol{\Phi}_{[i,j],l} = \varphi_l(s_i, a_j), \quad \boldsymbol{\Phi} \in \mathbb{R}^{NM \times d} \tag{14}$$

an approximate Q-vector is written $\widehat{\boldsymbol{Q}} = \boldsymbol{\Phi}\theta$. Plugging this into (13), we get:

$$\boldsymbol{B}_Q^\pi(\boldsymbol{\Phi}\theta) = \boldsymbol{R} + \gamma \boldsymbol{T}^\pi \boldsymbol{\Phi}\theta \tag{15}$$

We are now ready to describe projected policy evaluation and BRM in the linear case. Note that the matrices and vectors in (15) are too large to be used directly in an implementation; however, we will see that starting from these large matrices and vectors, both the projected policy evaluation and the BRM solutions can be written in terms of smaller matrices and vectors, which can be stored and manipulated in practical algorithms.

Projected Policy Evaluation

Under appropriate conditions on the BFs and the weights ρ (see Section 6.1 for a discussion), the projected Bellman equation can be exactly solved in the linear case:

$$\widehat{Q} = \Pi^\rho(B_Q^\pi(\widehat{Q}))$$

so that a minimum of 0 is attained in the problem $\min_{\widehat{Q} \in \widehat{\mathcal{Q}}} \left\| \widehat{Q} - \Pi(B_Q^\pi(\widehat{Q})) \right\|$. In matrix form, the projected Bellman equation is:

$$\begin{aligned} \boldsymbol{\Phi}\theta &= \boldsymbol{\Pi}^\rho \boldsymbol{B}_Q^\pi(\boldsymbol{\Phi}\theta) \\ &= \boldsymbol{\Pi}^\rho(\boldsymbol{R} + \gamma \boldsymbol{T}^\pi \boldsymbol{\Phi}\theta) \end{aligned} \tag{16}$$

where $\boldsymbol{\Pi}^\rho$ is a closed-form, matrix representation of the weighted least-squares projection Π^ρ:

$$\boldsymbol{\Pi}^\rho = \boldsymbol{\Phi}(\boldsymbol{\Phi}^\top \boldsymbol{\rho} \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^\top \boldsymbol{\rho} \tag{17}$$

The weight matrix $\boldsymbol{\rho}$ collects the weights of each state-action pair on its main diagonal:

$$\boldsymbol{\rho}_{[i,j],[i,j]} = \rho(s_i, a_j), \quad \boldsymbol{\rho} \in \mathbb{R}^{NM \times NM} \tag{18}$$

After substituting (17) into (16), a left-multiplication by $\boldsymbol{\Phi}^\top \boldsymbol{\rho}$, and a rearrangement of terms, we obtain:

$$\boldsymbol{\Phi}^\top \boldsymbol{\rho} \boldsymbol{\Phi}\, \theta = \gamma \boldsymbol{\Phi}^\top \boldsymbol{\rho} \boldsymbol{T}^\pi \boldsymbol{\Phi}\, \theta + \boldsymbol{\Phi}^\top \boldsymbol{\rho} \boldsymbol{R}$$

or in condensed form:

$$A\theta = \gamma B\theta + b \tag{19}$$

with the notations $A = \boldsymbol{\Phi}^\top \boldsymbol{\rho} \boldsymbol{\Phi}$, $B = \boldsymbol{\Phi}^\top \boldsymbol{\rho} \boldsymbol{T}^\pi \boldsymbol{\Phi}$, and $b = \boldsymbol{\Phi}^\top \boldsymbol{\rho} \boldsymbol{R}$. The matrices A and B are in $\mathbb{R}^{d \times d}$, while b is a vector in $\mathbb{R}^d$. This is a crucial expression, highlighting

that the projected Bellman equation can be represented and solved using only small matrices and vectors (of size $d \times d$ and d), instead of the large matrices and vectors (of sizes up to $NM \times NM$) that originally appeared in the formulas.

Next, two idealized algorithms for projected policy evaluation are introduced, which assume knowledge of A, B, and b. The next section will show how this assumption can be removed.

The idealized LSTD-Q belongs to the first ("one-shot") subclass of methods for projected policy evaluation in Figure 1. It simply solves the system (19) to arrive at the parameter vector θ. This parameter vector provides an approximate Q-function $\widehat{Q}^{\pi}(s,a) = \phi^{\top}(s,a)\theta$ of the considered policy π. Note that because θ appears on both sides of (19), this equation can be simplified to:

$$(A - \gamma B)\theta = b$$

The idealized LSPE-Q is an iterative algorithm, thus belonging to the second subclass of projected policy evaluation methods in Figure 1. It also relies on (19), but updates the parameter vector incrementally:

$$\begin{aligned} \theta_{\tau+1} &= \theta_{\tau} + \alpha(\theta'_{\tau+1} - \theta_{\tau}) \\ \text{where } A\theta'_{\tau+1} &= \gamma B\theta_{\tau} + b \end{aligned} \tag{20}$$

starting from some initial value θ_0. In this update, α is a positive step size parameter.

Bellman Residual Minimization

Consider the problem solved by BRM-Q (9), specialized to the weighted Euclidean norm used in this section:

$$\min_{\widehat{Q} \in \widehat{\mathcal{Q}}} \left\| \widehat{Q} - B^{\pi}_{Q}(\widehat{Q}) \right\|^2_{\rho} \tag{21}$$

Using the matrix expression (15) for B^{π}, this minimization problem can be rewritten in terms of the parameter vector as follows:

$$\begin{aligned} &\min_{\theta \in \mathbb{R}^d} \|\boldsymbol{\Phi}\theta - \boldsymbol{R} - \gamma \boldsymbol{T}^{\pi}\boldsymbol{\Phi}\theta\|^2_{\rho} \\ &\quad = \min_{\theta \in \mathbb{R}^d} \|(\boldsymbol{\Phi} - \gamma \boldsymbol{T}^{\pi}\boldsymbol{\Phi})\theta - \boldsymbol{R}\|^2_{\rho} \\ &\quad = \min_{\theta \in \mathbb{R}^d} \|C\theta - \boldsymbol{R}\|^2_{\rho} \end{aligned}$$

where $C = \boldsymbol{\Phi} - \gamma \boldsymbol{T}^{\pi}\boldsymbol{\Phi}$. By linear-algebra arguments, the minimizer of this problem can be found by solving the equation:

$$C^{\top}\boldsymbol{\rho}C\,\theta = C^{\top}\boldsymbol{\rho}\boldsymbol{R} \tag{22}$$

The idealized BRM-Q algorithm consists of solving this equation to arrive at θ, and thus to an approximate Q-function of the policy considered.

It is useful to note that BRM is closely related to projected policy evaluation, as it can be interpreted as solving a similar projected equation (Bertsekas, 2010b). Specifically, finding a minimizer in (21) is equivalent to solving:

$$\boldsymbol{\Phi}\theta = \boldsymbol{\Pi}^{\rho}(\bar{\boldsymbol{B}}_Q^{\pi}(\boldsymbol{\Phi}\theta))$$

which is similar to the projected Bellman equation (16), except that the modified mapping $\bar{\boldsymbol{B}}_Q^{\pi}$ is used:

$$\bar{\boldsymbol{B}}_Q^{\pi}(\boldsymbol{\Phi}\theta) = \boldsymbol{B}_Q^{\pi}(\boldsymbol{\Phi}\theta) + \gamma(\boldsymbol{T}^{\pi})^{\top}[\boldsymbol{\Phi}\theta - \boldsymbol{B}_Q^{\pi}(\boldsymbol{\Phi}\theta)]$$

3.3 Model-Free Implementations

To obtain the matrices and vectors appearing in their equations, the idealized algorithms given above would require the transition and reward functions of the Markov decision process, which are unknown in a reinforcement learning context. Moreover, the algorithms would need to iterate over all the state-action pairs, which is impossible in the large state-action spaces they are intended for.

This means that, in practical reinforcement learning, *sample-based* versions of these algorithms should be employed. Fortunately, thanks to the special structure of the matrices and vectors involved, they can be estimated from samples. In this section, we derive practically implementable LSTD-Q, LSPE-Q, and BRM-Q algorithms.

Projected Policy Evaluation

Consider a set of n transition samples of the form (s_i, a_i, s_i', r_i), $i = 1, \ldots, n$, where s_i' is drawn from the distribution $T(s_i, a_i, \cdot)$ and $r_i = R(s_i, a_i, s_i')$. The state-action pairs (s_i, a_i) must be drawn from the distribution given by the weight function ρ. For example, when a generative model is available, samples can be drawn independently from ρ and the model can be used to find corresponding next states and rewards. From an opposite perspective, it can also be said that the distribution of the state-action pairs implies the weights used in the projected Bellman equation. For example, collecting samples along a set of trajectories of the system, or along a single trajectory, implicitly leads to a distribution (weights) ρ.

Estimates of the matrices A and B and of the vector b appearing in (19) and (20) can be constructed from the samples as follows:

$$\begin{aligned}\widehat{A}_i &= \widehat{A}_{i-1} + \phi(s_i,a_i)\phi^\top(s_i,a_i)\\ \widehat{B}_i &= \widehat{B}_{i-1} + \phi(s_i,a_i)\phi^\top(s_i',\pi(s_i'))\\ \widehat{b}_i &= \widehat{b}_{i-1} + \phi(s_i,a_i)r_i\end{aligned} \tag{23}$$

starting from zero initial values ($\widehat{A}_0 = 0, \widehat{B}_0 = 0, \widehat{b}_0 = 0$).

LSTD-Q processes the n samples using (23) and then solves the equation:

$$\frac{1}{n}\widehat{A}_n\theta = \gamma\frac{1}{n}\widehat{B}_n\theta + \frac{1}{n}\widehat{b}_n \tag{24}$$

or, equivalently:

$$\frac{1}{n}(\widehat{A}_n - \gamma\widehat{B}_n)\theta = \frac{1}{n}\widehat{b}_n$$

to find a parameter vector θ. Note that, because this equation is an approximation of (19), the parameter vector θ is only an approximation to the solution of (19) (however, for notation simplicity we denote it in the same way). The divisions by n, while not mathematically necessary, increase the numerical stability of the algorithm, by preventing the coefficients from growing too large as more samples are processed. The composite matrix $(\widehat{A} - \gamma\widehat{B})$ can also be updated as a single entity $\widehat{(A-\gamma B)}$, thereby eliminating the need of storing in memory two potentially large matrices $\widehat{A}$ and $\widehat{B}$.

Algorithm 2 summarizes this more memory-efficient variant of LSTD-Q. Note that the update of $\widehat{(A-\gamma B)}$ simply accumulates the updates of $\widehat{A}$ and $\widehat{B}$ in (23), where the term from $\widehat{B}$ is properly multiplied by $-\gamma$.

1: **input** policy to evaluate π, BFs ϕ, samples (s_i,a_i,s_i',r_i), $i=1,\ldots,n$
2: $\widehat{(A-\gamma B)}_0 \leftarrow 0$, $\widehat{b}_0 \leftarrow 0$
3: **for** $i=1,\ldots,n$ **do**
4: $\quad\widehat{(A-\gamma B)}_i \leftarrow \widehat{(A-\gamma B)}_{i-1} + \phi(s_i,a_i)\phi^\top(s_i,a_i) - \gamma\phi(s_i,a_i)\phi^\top(s_i',\pi(s_i'))$
5: $\quad\widehat{b}_i \leftarrow \widehat{b}_{i-1} + \phi(s_i,a_i)r_i$
6: **end for**
7: solve $\frac{1}{n}\widehat{(A-\gamma B)}_n\theta = \frac{1}{n}\widehat{b}_n$
8: **output** Q-function $\widehat{Q}^\pi(s,a) = \phi^\top(s,a)\theta$

Algorithm 2: LSTD-Q.

Combining LSTD-Q policy evaluation with exact policy improvements leads to what is perhaps the most widely used policy iteration algorithm from the least-squares class, called simply *least-squares policy iteration* (LSPI).

LSPE-Q uses the same estimates $\widehat{A}$, $\widehat{B}$, and $\widehat{b}$ as LSTD-Q, but updates the parameter vector iteratively. In its basic variant, LSPE-Q starts with an arbitrary initial parameter vector θ_0 and updates using:

$$\begin{aligned} \theta_i &= \theta_{i-1} + \alpha(\theta'_i - \theta_{i-1}) \\ \text{where } \frac{1}{i}\widehat{A}_i\theta'_i &= \gamma\frac{1}{i}\widehat{B}_i\theta_{i-1} + \frac{1}{i}\widehat{b}_i \end{aligned} \tag{25}$$

This update is an approximate, sample-based version of the idealized version (20). Similarly to LSTD-Q, the divisions by i increase the numerical stability of the updates. The estimate $\widehat{A}$ may not be invertible at the start of the learning process, when only a few samples have been processed. A practical solution to this issue is to initialize $\widehat{A}$ to a small multiple of the identity matrix.

A more flexible algorithm than (25) can be obtained by (i) processing more than one sample in-between consecutive updates of the parameter vector, as well as by (ii) performing more than one parameter update after processing each (batch of) samples, while holding the coefficient estimates constant. The former modification may increase the stability of the algorithm, particularly in the early stages of learning, while the latter may accelerate its convergence, particularly in later stages as the estimates $\widehat{A}$, $\widehat{B}$, and $\widehat{b}$ become more precise.

Algorithm 3 shows this more flexible variant of LSPE-Q, which (i) processes batches of $\bar{n}$ samples in-between parameter update episodes ($\bar{n}$ should preferably be a divisor of n). At each such episode, (ii) the parameter vector is updated N_{upd} times. Notice that unlike in (25), the parameter update index τ is different from the sample index i, since the two indices are no longer advancing synchronously.

input policy to evaluate π, BFs ϕ, samples (s_i, a_i, s'_i, r_i), $i = 1, \ldots, n$
  step size α, small constant $\delta_A > 0$
  batch length $\bar{n}$, number of consecutive parameter updates N_{upd}
$\widehat{A}_0 \leftarrow \delta_A I$, $\widehat{B}_0 \leftarrow 0$, $\widehat{b}_0 \leftarrow 0$
$\tau = 0$
for $i = 1, \ldots, n$ **do**
  $\widehat{A}_i \leftarrow \widehat{A}_{i-1} + \phi(s_i, a_i)\phi^\top(s_i, a_i)$
  $\widehat{B}_i \leftarrow \widehat{B}_{i-1} + \phi(s_i, a_i)\phi^\top(s'_i, \pi(s'_i))$
  $\widehat{b}_i \leftarrow \widehat{b}_{i-1} + \phi(s_i, a_i)r_i$
  if i is a multiple of $\bar{n}$ **then**
    for $\tau = \tau, \ldots, \tau + N_{\text{upd}} - 1$ **do**
      $\theta_{\tau+1} \leftarrow \theta_\tau + \alpha(\theta'_{\tau+1} - \theta_\tau)$, where $\frac{1}{i}\widehat{A}_i\theta'_{\tau+1} = \gamma\frac{1}{i}\widehat{B}_i\theta_\tau + \frac{1}{i}\widehat{b}_i$
    end for
  end if
end for
output Q-function $\widehat{Q}^\pi(s,a) = \phi^\top(s,a)\theta_{\tau+1}$

Algorithm 3: LSPE-Q.

Because LSPE-Q must solve the system in (25) multiple times, it will require more computational effort than LSTD-Q, which solves the similar system (24) only once. On the other hand, the incremental nature of LSPE-Q can offer it advantages over LSTD-Q. For instance, LSPE can benefit from a good initial value of the parameter vector, and better flexibility can be achieved by controlling the step size.

Bellman Residual Minimization

Next, sample-based BRM is briefly discussed. The matrix $C^\top \boldsymbol{\rho} C$ and the vector $C^\top \boldsymbol{\rho} \boldsymbol{R}$ appearing in the idealized BRM equation (22) can be estimated from samples. The estimation procedure requires *double* transition samples, that is, for each state-action sample, two independent transitions to next states must be available. These double transition samples have the form $(s_i, a_i, s'_{i,1}, r_{i,1}, s'_{i,2}, r_{i,2})$, $i = 1, \ldots, n$, where $s'_{i,1}$ and $s'_{i,2}$ are independently drawn from the distribution $T(s_i, a_i, \cdot)$, while $r_{i,1} = R(s_i, a_i, s'_{i,1})$ and $r_{i,2} = R(s_i, a_i, s'_{i,2})$. Using these samples, estimates can be built as follows:

$$\begin{aligned}
(\widehat{C^\top \boldsymbol{\rho} C})_i &= (\widehat{C^\top \boldsymbol{\rho} C})_{i-1} + \\
&\quad [\phi(s_i, a_i) - \gamma \phi(s'_{i,1}, \pi(s'_{i,1}))] \cdot [\phi^\top(s_i, a_i) - \gamma \phi^\top(s'_{i,2}, \pi(s'_{i,2}))] \qquad (26) \\
(\widehat{C^\top \boldsymbol{\rho} \boldsymbol{R}})_i &= (\widehat{C^\top \boldsymbol{\rho} \boldsymbol{R}})_{i-1} + \phi(s_i, a_i) - \gamma \phi(s'_{i,2}, \pi(s'_{i,2}))
\end{aligned}$$

starting from zero initial values: $(\widehat{C^\top \boldsymbol{\rho} C})_0 = 0, (\widehat{C^\top \boldsymbol{\rho} \boldsymbol{R}})_0 = 0$. Once all the samples have been processed, the estimates can be used to approximately solve (22), obtaining a parameter vector and thereby an approximate Q-function.

The reason for using double samples is that, if a single sample (s_i, a_i, s'_i, r_i) were used to build sample products of the form $[\phi(s_i, a_i) - \gamma \phi(s'_i, \pi(s'_i))] \cdot [\phi^\top(s_i, a_i) - \gamma \phi^\top(s'_i, \pi(s'_i))]$, such samples would be biased, which in turn would lead to a biased estimate of $C^\top \boldsymbol{\rho} C$ and would make the algorithm unreliable (Baird, 1995; Bertsekas, 1995).

The Need for Exploration

A crucial issue that arises in all the algorithms above is *exploration*: ensuring that the state-action space is sufficiently covered by the available samples. Consider first the exploration of the action space. The algorithms are typically used to evaluate deterministic policies. If samples were only collected according to the current policy π, i.e., if all samples were of the form $(s, \pi(s))$, no information about pairs (s, a) with $a \neq \pi(s)$ would be available. Therefore, the approximate Q-values of such pairs would be poorly estimated and unreliable for policy improvement. To alleviate this problem, exploration is necessary: sometimes, actions different from $\pi(s)$ have to be selected, e.g., in a random fashion. Looking now at the state space, exploration plays another helpful role when samples are collected along trajectories of the system. In the absence of exploration, areas of the state space that are not visited under the current policy would not be represented in the sample set, and the value function would therefore be poorly estimated in these areas, even though they may be important in solving the problem. Instead, exploration drives the system along larger areas of the state space.

Computational Considerations

The computational expense of least-squares algorithms for policy evaluation is typically dominated by solving the linear systems appearing in all of them. However, for one-shot algorithms such as LSTD and BRM, which only need to solve the system once, the time needed to process samples may become dominant if the number of samples is very large.

The linear systems can be solved in several ways, e.g., by matrix inversion, by Gaussian elimination, or by incrementally computing the inverse with the Sherman-Morrison formula (see Golub and Van Loan, 1996, Chapters 2 and 3). Note also that, when the BF vector ϕ is sparse, as in the often-encountered case of localized BFs, this sparsity can be exploited to greatly improve the computational efficiency of the matrix and vector updates in all the least-squares algorithms.

3.4 Bibliographical notes

The high-level introduction of Section 3.1 followed the line of (Farahmand et al, 2009). After that, we followed at places the derivations in Chapter 3 of (Buşoniu et al, 2010a).

LSTD was introduced in the context of V-functions (LSTD-V) by Bradtke and Barto (1996), and theoretically studied by, e.g., Boyan (2002); Konda (2002); Nedić and Bertsekas (2003); Lazaric et al (2010b); Yu (2010). LSTD-Q, the extension to the Q-function case, was introduced by Lagoudakis et al (2002); Lagoudakis and Parr (2003a), who also used it to develop the LSPI algorithm. LSTD-Q was then used and extended in various ways, e.g., by Xu et al (2007); Li et al (2009); Kolter and Ng (2009); Buşoniu et al (2010d,b); Thiery and Scherrer (2010).

LSPE-V was introduced by Bertsekas and Ioffe (1996) and theoretically studied by Nedić and Bertsekas (2003); Bertsekas et al (2004); Yu and Bertsekas (2009). Its extension to Q-functions, LSPE-Q, was employed by, e.g., Jung and Polani (2007a,b); Buşoniu et al (2010d).

The idea of minimizing the Bellman residual was proposed as early as in (Schweitzer and Seidmann, 1985). BRM-Q and variants were studied for instance by Lagoudakis and Parr (2003a); Antos et al (2008); Farahmand et al (2009), while Scherrer (2010) recently compared BRM approaches with projected approaches. It should be noted that the variants of Antos et al (2008) and Farahmand et al (2009), called "modified BRM" by Farahmand et al (2009), eliminate the need for double sampling by introducing a change in the minimization problem (9).

4 Online least-squares policy iteration

An important topic in practice is the application of least-squares methods to *online learning*. Unlike in the offline case, where only the final performance matters, in online learning the performance should improve once every few transition samples. Policy iteration can take this requirement into account by performing policy improvements once every few transition samples, before an accurate evaluation of the current policy can be completed. Such a variant is called *optimistic* policy iteration (Bertsekas and Tsitsiklis, 1996; Sutton, 1988; Tsitsiklis, 2002). In the extreme, fully optimistic case, the policy is improved after every single transition. Optimistic policy updates were combined with LSTD-Q – thereby obtaining optimistic LSPI – in our works (Buşoniu et al, 2010d,b) and with LSPE-Q in (Jung and Polani, 2007a,b). Li et al (2009) explored a non-optimistic, more computationally involved approach to online policy iteration, in which LSPI is fully executed between consecutive sample-collection episodes.

Next, we briefly discuss the online, optimistic LSPI method we introduced in (Buşoniu et al, 2010d), shown here as Algorithm 4. The same matrix and vector

input BFs ϕ, policy improvement interval L, exploration schedule,
  initial policy π_0, small constant $\delta_A > 0$
$k \leftarrow 0, t = 0$
$\widehat{(A-\gamma B)}_0 \leftarrow \delta_A I, \widehat{b}_0 \leftarrow 0$
observe initial state s_0
repeat
  $a_t \leftarrow \pi_k(s_t)$ + exploration
  apply a_t, observe next state s_{t+1} and reward r_{t+1}
  $\widehat{(A-\gamma B)}_{t+1} \leftarrow \widehat{(A-\gamma B)}_t + \phi(s_t,a_t)\phi^\top(s_t,a_t) - \gamma\phi(s_t,a_t)\phi^\top(s_{t+1},\pi(s_{t+1}))$
  $\widehat{b}_{t+1} \leftarrow \widehat{b}_t + \phi(s_t,a_t)r_{t+1}$
  if $t = (k+1)L$ **then**
    solve $\frac{1}{t}\widehat{(A-\gamma B)}_{t+1}\theta_k = \frac{1}{t}\widehat{b}_{t+1}$ to find θ_k
    $\pi_{k+1}(s) \leftarrow \arg\max_{a\in A} \phi^\top(s,a)\theta_k \quad \forall s$
    $k \leftarrow k+1$
  end if
  $t \leftarrow t+1$
until experiment finished

Algorithm 4: Online LSPI.

estimates are used as in offline LSTD-Q and LSPI, but there are important differences. First, online LSPI collects its own samples, by using its current policy to interact with the system. This immediately implies that exploration must explicitly be added on top of the (deterministic) policy. Second, optimistic policy improvements are performed, that is, the algorithm improves the policy without waiting for the estimates $\widehat{(A-\gamma B)}$ and $\widehat{b}$ to get close to their asymptotic values for the current policy. Moreover, these estimates continue to be updated without being reset after the policy changes – so in fact they correspond to multiple policies. The underlying

assumption here is that $A - \gamma B$ and b are similar for subsequent policies. A more computationally costly alternative would be to store the samples and rebuild the estimates from scratch before every policy update, but as will be illustrated in the example of Section 5, this may not be necessary in practice.

The number L of transitions between consecutive policy improvements is a crucial parameter of the algorithm. For instance, when $L = 1$, online LSPI is fully optimistic. In general, L should not be too large, to avoid potentially bad policies from being used too long. Note that, as in the offline case, improved policies do not have to be explicitly computed in online LSPI, but can be computed on demand.

5 Example: Car on the Hill

In order to exemplify the behavior of least-squares methods for policy iteration, we apply two such methods (offline LSPI and online, optimistic LSPI) to the car-on-the-hill problem (Moore and Atkeson, 1995), a classical benchmark for approximate reinforcement learning. Thanks to its low dimensionality, this problem can be solved using simple linear approximators, in which the BFs are distributed on an equidistant grid. This frees us from the difficulty of customizing the BFs or resorting to a nonparametric approximator, and allows us to focus instead on the behavior of the algorithms.

In the car-on-the-hill problem, a point mass (the 'car') must be driven past the top of a frictionless hill by applying a horizontal force, see Figure 2, left. For some initial states, due to the limited available force, the car must first be driven to the left, up the opposite slope, and gain momentum prior to accelerating to the right, towards the goal.

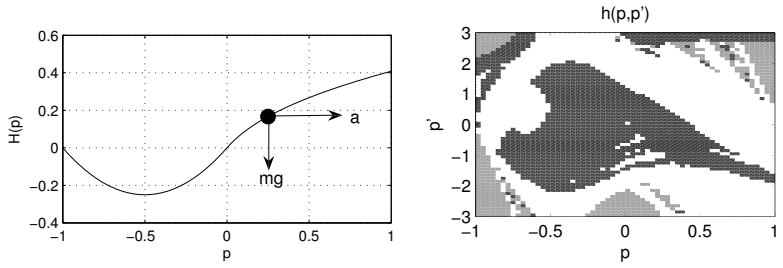

Fig. 2 Left: The car on the hill, with the "car" shown as a black bullet. Right: a near-optimal policy (black means $a = -4$, white means $a = +4$, gray means both actions are equally good).

Denoting the horizontal position of the car by p, its dynamics are (in the variant of Ernst et al, 2005):

$$\ddot{p} = \frac{a - 9.81 H'(p) - \dot{p}^2 H'(p) H''(p)}{1 + [H'(p)]^2} \tag{27}$$

where the hill shape $H(p)$ is given by p^2+p for $p<0$, and by $p^{-1}\sqrt{1+5p^2}$ for $p\geq 0$. The notations $\dot{p}$ and $\ddot{p}$ indicate the first and second time derivatives of p, while $H'(p)$ and $H''(p)$ denote the first and second derivatives of H with respect to p. The state variables are the position and the speed, $s=[p,\dot{p}]^\top$, and the action a is the applied force. Discrete-time dynamics T are obtained by numerically integrating (27) between consecutive time steps, using a discrete time step of 0.1 s. Thus, s_t and p_t are sampled versions of the continuous variables s and p. The state space is $S=[-1,1]\times[-3,3]$ plus a terminal state that is reached whenever s_{t+1} would fall outside these bounds, and the discrete action space is $A=\{-4,4\}$. The goal is to drive past the top of the hill to the right with a speed within the allowed limits, while reaching the terminal state in any other way is considered a failure. To express this goal, the following reward is chosen:

$$R(s_t,a_t,s_{t+1})=\begin{cases}-1 & \text{if } p_{t+1}<-1 \text{ or } |\dot{p}_{t+1}|>3\\ 1 & \text{if } p_{t+1}>1 \text{ and } |\dot{p}_{t+1}|\leq 3\\ 0 & \text{otherwise}\end{cases} \tag{28}$$

with the discount factor $\gamma=0.95$. Figure 2, right shows a near-optimal policy for this reward function.

To apply the algorithms considered, the Q-function is approximated over the state space using bilinear interpolation on an equidistant 13×13 grid. To represent the Q-function over the discrete action space, separate parameters are stored for the two discrete actions, so the approximator can be written as:

$$\widehat{Q}(s,a_j)=\sum_{i=1}^{169}\varphi_i(s)\theta_{i,j}$$

where the state-dependent BFs $\varphi_i(s)$ provide the interpolation coefficients, with at most 4 BFs being non-zero for any s, and $j=1,2$. By replicating the state-dependent BFs for both discrete actions, this approximator can be written in the standard form (10), and can therefore be used in least-squares methods for policy iteration.

First, we apply LSPI with this approximator to the car-on-the-hill problem. A set of 10 000 random, uniformly distributed state-action samples are independently generated; these samples are reused to evaluate the policy at each policy iteration. With these settings, LSPI typically converges in 7 to 9 iterations (from 20 independent runs, where convergence is considered achieved when the difference between consecutive parameter vectors drops below 0.001). Figure 3, top illustrates a subsequence of policies found during a representative run. Subject to the resolution limitations of the chosen representation, a reasonable approximation of the near-optimal policy in Figure 2, right is found.

For comparison purposes, we also run an approximate *value iteration* algorithm using the same approximator and the same convergence threshold. (The actual algorithm, called fuzzy Q-iteration, is described in Buşoniu et al (2010c); here, we are only interested in the fact that it is representative for the approximate value iteration class.) The algorithm converges after 45 iterations. This slower convergence of

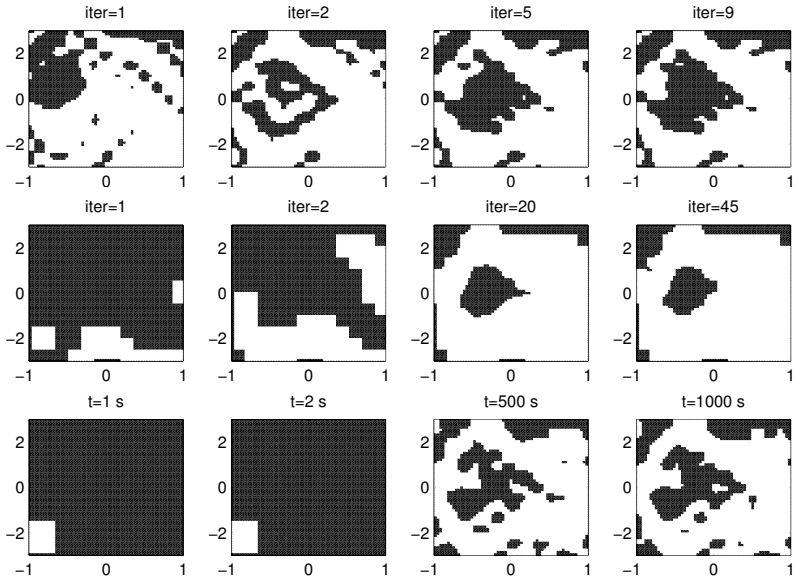

Fig. 3 Representative subsequences of policies found by the algorithms considered. Top: offline LSPI; middle: fuzzy Q-iteration; bottom: online LSPI. For axis and color meanings, see Figure 2, right, additionally noting that here the negative (black) action is preferred when both actions are equally good.

value iteration compared to policy iteration is often observed in practice. Figure 3, middle shows a subsequence of policies found by value iteration. Like for the PI algorithms we considered, the policies are implicitly represented by the approximate Q-function, in particular, actions are chosen by maximizing the Q-function as in (5). The final solution is different from the one found by LSPI, and the algorithms also converge differently: LSPI initially makes large steps in the policy space at each iteration (so that, e.g., the structure of the policy is already visible after the second iteration), whereas value iteration makes smaller, incremental steps.

Finally, online, optimistic LSPI is applied to the car on the hill. The experiment is run for 1000 s of simulated time, so that in the end 10 000 samples have been collected, like for the offline algorithm. This interval is split into separate learning trials, initialized at random initial states and stopping when a terminal state has been reached, or otherwise after 3 s. Policy improvements are performed once every 10 samples (i.e., every 1 s), and an ε-greedy exploration strategy is used (see Chapter 2), with $\varepsilon = 1$ initially and decaying exponentially so that it reaches a value of 0.1 after 350 s. Figure 3, bottom shows a subsequence of policies found during a representative run. Online LSPI makes smaller steps in the policy space than offline LSPI because, in-between consecutive policy improvements, it processes fewer samples, which come from a smaller region of the state space. In fact, at the end of learning LSPI has processed each sample only once, whereas offline LSPI processes all the samples once at every iteration.

Figure 4 shows the performance of policies found by online LSPI along the online learning process, in comparison to the final policies found by offline LSPI. The performance is measured by the average empirical return over an equidistant grid of initial states, evaluated by simulation with a precision of 0.001; we call this average return "score". Despite theoretical uncertainty about its convergence and near-optimality, online LSPI empirically reaches at least as good performance as the offline algorithm (for encouraging results in several other problems, see our papers (Buşoniu et al, 2010d,b) and Chapter 5 of (Buşoniu et al, 2010a)). For completeness, we also report the score of the – deterministically obtained – value iteration solution: 0.219, slightly lower than that obtained by either version of LSPI.

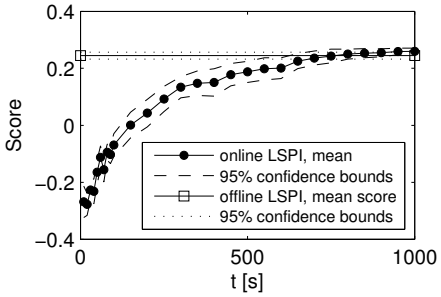

Fig. 4 Evolution of the policy score in online LSPI, compared with offline LSPI. Mean values with 95% confidence intervals are reported, from 20 independent experiments.

The execution time was around 34 s for LSPI, around 28 s for online LSPI, and 0.3 s for value iteration. This illustrates the fact that the convergence rate advantage of policy iteration does not necessarily translate into computational savings – since each policy evaluation can have a complexity comparable with the entire value iteration. In our particular case, LSPI requires building the estimates of A, B and b and solving a linear system of equations, whereas for the interpolating approximator employed, each approximate value iteration reduces to a very simple update of the parameters. For other types of approximators, value iteration algorithms will be more computationally intensive, but still tend to require less computation per iteration than PI algorithms. Note also that the execution time of online LSPI is much smaller than the 1000 s *simulated* experiment duration.

6 Performance Guarantees

In this section we review the main theoretical guarantees about least-squares methods for policy evaluation and policy iteration. We first discuss convergence properties and the quality of the solutions obtained *asymptotically*, as the number of samples processed and iterations executed grows to infinity. Then, probabilistic bounds

are provided on the performance of the policy obtained by using a *finite* number of samples and iterations.

6.1 Asymptotic Convergence and Guarantees

While theoretical results are mainly given for V-functions in the literature, they directly extend to Q-functions, by considering the Markov chain of state-action pairs under the current policy, rather than the Markov chain of states as in the case of V-functions. We will therefore use our Q-function-based derivations above to explain and exemplify the guarantees.

Throughout, we require that the BFs are linearly independent, implying that the BF matrix $\boldsymbol{\Phi}$ has full column rank. Intuitively, this means there are no redundant BFs.

Projected Policy Evaluation

In the context of projected policy evaluation, one important difference between the LSTD and LSPE families of methods is the following. LSTD-Q will produce a meaningful solution whenever the equation (19) has a solution, which can happen for many weight functions ρ. In contrast, to guarantee the convergence of the basic LSPE-Q iteration, one generally must additionally require that the samples follow ρ^π, the *stationary distribution* over state-action pairs induced by the policy considered (stationary distribution of π, for short). Intuitively, this means that the weight of each state-action pair (s,a) is equal to the steady-state probability of this pair along an infinitely-long trajectory generated with the policy π. The projection mapping [6] Π^{ρ^π} is nonexpansive with respect to the norm $\|\cdot\|_{\rho^\pi}$ weighted by the stationary distribution ρ^π, and because additionally the original Bellman mapping B_Q^π is a contraction with respect to this norm, the projected Bellman mapping $\Pi^{\rho^\pi}(B_Q^\pi(\cdot))$ is also a contraction.[7]

Confirming the LSTD is not very dependent on using ρ^π, Yu (2010) proved that with a minor modification, the solution found by LSTD converges as $n \to \infty$, even when one policy is evaluated using samples from a different policy – the so-called off-policy case. Even for LSPE, it may be possible to mitigate the destabilizing effects of violating convergence assumptions, by controlling the step size α. Furthermore, a modified LSPE-like update has recently been proposed that converges without requiring that $\Pi^\rho(B_Q^\pi(\cdot))$ is a contraction, see Bertsekas (2010b, 2011). These types of results are important in practice because they pave the way for *reusing*

[6] A projection mapping Π^{ρ^π} applied to a function f w.r.t. a space $\mathcal{F}$ returns the closest element in $\mathcal{F}$ to the function f, where the distance is defined according to the L2 norm and the measure ρ^π.

[7] A mapping $f(x)$ is a contraction with factor $\gamma < 1$ if for any x,x', $\|f(x) - f(x')\| \leq \gamma\|x - x'\|$. The mapping is a nonexpansion (a weaker property) if the inequality holds for $\gamma = 1$.

samples to evaluate different policies, that is, at different iterations of the overall PI algorithm. In contrast, if the stationary distribution must be followed, new samples have to be generated at each iteration, using the current policy.

Assuming now, for simplicity, that the stationary distribution ρ^π is used, and thus that the projected Bellman equation (19) has a unique solution (the projected Bellman operator is a contraction and it admits one unique fixed point), the following informal, but intuitive line of reasoning is useful to understand the convergence of the sample-based LSTD-Q and LSPE-Q. Asymptotically, as $n \to \infty$, it is true that $\frac{1}{n}\widehat{A}_n \to A$, $\frac{1}{n}\widehat{B}_n \to B$, and $\frac{1}{n}\widehat{b}_n \to b$, because the empirical distribution of the state-action samples converges to ρ, while the empirical distribution of next-state samples s' for each pair (s,a) converges to $T(s,a,s')$. Therefore, the practical, sample-based LSTD-Q converges to its idealized version (19), and LSTD-Q asymptotically finds the solution of the projected Bellman equation. Similarly, the sample-based LSPE-Q asymptotically becomes equivalent to its idealized version (20), which is just an incremental variant of (19) and will therefore produce the same solution in the end. In fact, it can additionally be shown that, as n grows, the solutions of LSTD-Q and LSPE-Q converge to each other faster than they converge to their limit, see Yu and Bertsekas (2009).

Let us investigate now the quality of the solution. Under the stationary distribution ρ^π, we have (Bertsekas, 2010b; Tsitsiklis and Van Roy, 1997):

$$\left\|Q^\pi - \widehat{Q}^\pi\right\|_{\rho^\pi} \leq \frac{1}{\sqrt{1-\gamma^2}} \left\|Q^\pi - \Pi^{\rho^\pi}(Q^\pi)\right\|_{\rho^\pi} \tag{29}$$

where $\widehat{Q}^\pi$ is the Q-function given by the parameter θ that solves the projected Bellman equation (19) for $\rho = \rho^\pi$. Thus, we describe the representation power of the approximator by the distance $\left\|Q^\pi - \Pi^{\rho^\pi}(Q^\pi)\right\|_{\rho^\pi}$ between the true Q-function Q^π and its projection $\Pi^{\rho^\pi}(Q^\pi)$. As the approximator becomes more powerful, this distance decreases. Then, projected policy evaluation leads to an approximate Q-function $\widehat{Q}^\pi$ with an error proportional to this distance. The proportion is given by the discount factor γ, and grows as γ approaches 1. Recently, efforts have been made to refine this result in terms of properties of the dynamics and of the set $\widehat{\mathscr{Q}}$ of representable Q-functions (Scherrer, 2010; Yu and Bertsekas, 2010).

Bellman Residual Minimization

The following relationship holds for any Q-function Q, see e.g. Scherrer (2010):

$$\|Q^\pi - Q\|_{\rho^\pi} \leq \frac{1}{1-\gamma} \left\|Q - B^\pi_Q(Q)\right\|_{\rho^\pi} \tag{30}$$

and with ρ^π the stationary distribution. Consider now the on-policy BRM solution – the Q-function $\widehat{Q}^\pi$ given by the parameter that solves the BRM equation (22) for

$\rho = \rho^\pi$. Because this Q-function minimizes the right-hand side of the inequality (30), the error $\left\| Q^\pi - \widehat{Q}^\pi \right\|_{\rho^\pi}$ of the solution found is also small.

Comparing projected policy evaluation with BRM, no general statements can be made about the relative quality of their solution. For instance, in the context of policy evaluation only, Scherrer (2010) suggests based on an empirical study that projected policy evaluation may outperform BRM more often (for more problem instances) than the other way around; but when it fails, it may do so with much larger errors than those of BRM. For additional insight and comparisons, see, e.g., Munos (2003); Scherrer (2010).

Approximate Policy Iteration

A general result about policy iteration can be given in terms of the infinity norm, as follows. If the policy evaluation error $\left\| \widehat{Q}^{\pi_k} - Q^{\pi_k} \right\|_\infty$ is upper-bounded by ε at every iteration $k \geq 0$ (see again Algorithm 1), and if policy improvements are exact (according to our assumptions in Section 2), then policy iteration eventually produces policies with a performance (i.e., the corresponding value function) that lies within a bounded distance from the optimal performance (Bertsekas and Tsitsiklis, 1996; Lagoudakis and Parr, 2003a) (i.e., the optimal value function):

$$\limsup_{k \to \infty} \| Q^{\pi_k} - Q^* \|_\infty \leq \frac{2\gamma}{(1-\gamma)^2} \cdot \varepsilon \tag{31}$$

Here, Q^* is the optimal Q-function and corresponds to the optimal performance, see Chapter 2. Note that if approximate policy improvements are performed, a similar bound holds, but the policy improvement error must also be included in the right hand side.

An important remark is that the sequence of policies is generally not guaranteed to converge to a fixed policy. For example, the policy may end up oscillating along a limit cycle. Nevertheless, all the policies along the cycle will have a high performance, in the sense of (31).

Note that (31) uses infinity norms, whereas the bounds (29) and (30) for the policy evaluation component use Euclidean norms. The two types of bounds cannot be easily combined to yield an overall bound for approximate policy iteration. Policy iteration bounds for Euclidean norms, which we will not detail here, were developed by Munos (2003).

Consider now optimistic variants of policy iteration, such as online LSPI. The performance guarantees above rely on small policy evaluation errors, whereas in the optimistic case, the policy is improved before an accurate value function is available, which means the policy evaluation error can be very large. For this reason, the behavior of optimistic policy iteration is theoretically poorly understood at the moment, although the algorithms often work well in practice. See Bertsekas (2010b)

and Section 6.4 of Bertsekas and Tsitsiklis (1996) for discussions of the difficulties involved.

6.2 Finite-Sample Guarantees

The results reported in the previous section analyze the asymptotic performance of policy evaluation methods when the number of samples tends to infinity. Nonetheless, they do not provide any guarantee about how the algorithms behave when only a finite number of samples is available. In this section we report recent finite-sample bounds for LSTD and BRM and we discuss how they propagate through iterations in the policy iteration scheme.

While in the previous sections we focused on algorithms for the approximation of Q-functions, for sake of simplicity we report here the analysis for V-function approximation. The notation and the setting is exactly the same as in Section 3.2 and we simply redefine it for V-functions. We use a linear approximation architecture with parameters $\theta \in \mathbb{R}^d$ and basis functions $\varphi_i, i = 1,\ldots,d$ now defined as a mapping from the state space S to $\mathbb{R}$. We denote by $\phi : S \to \mathbb{R}^d$, $\phi(\cdot) = [\varphi_1(\cdot),\ldots,\varphi_d(\cdot)]^\top$ the BF vector (feature vector), and by $\mathscr{F}$ the linear function space spanned by the BFs φ_i, that is $\mathscr{F} = \left\{ f_\theta | \theta \in \mathbb{R}^d \text{ and } f_\theta(\cdot) = \phi^\top(\cdot)\theta \right\}$. We define $\tilde{\mathscr{F}}$ as the space obtained by truncating the functions in $\mathscr{F}$ at $V_{\max}$ (recall that $V_{\max}$ gives the maximum return and upper-bounds any value function). The truncated function $\tilde{f}$ is equal to $f(s)$ in all the states where $|f(s)| \leq V_{\max}$ and it is equal to $\text{sgn}(f(s))V_{\max}$ otherwise. Furthermore, let L be an upper bound for all the BFs, i.e., $\|\varphi_i\|_\infty \leq L$ for $i = 1,\ldots,d$. In the following we report the finite-sample analysis of the policy evaluation performance of LSTD-V and BRM-V, followed by the analysis of policy iteration algorithms that use LSTD-V and BRM-V in the policy evaluation step. The truncation to $V_{\max}$ is used for LSTD-V, but not for BRM-V.

Pathwise LSTD

Let π be the current policy and V^π its V-function. Let (s_t, r_t) with $t = 1,\ldots,n$ be a sample path (trajectory) of size n generated by following the policy π and $\boldsymbol{\Phi} = [\phi^\top(s_1);\ldots;\phi^\top(s_n)]$ be the BF matrix defined at the encountered states, where ";" denotes a vertical stacking of the vectors $\phi^\top(s_t)$ in the matrix. Pathwise LSTD-V is a version of LSTD-V obtained by defining an empirical transition matrix $\hat{\boldsymbol{T}}$ as follows: $\hat{\boldsymbol{T}}_{ij} = 1$ if $j = i+1, j \neq n$, otherwise $\hat{\boldsymbol{T}}_{ij} = 0$. When applied to a vector $s = [s_1,\ldots,s_n]^\top$, this transition matrix returns $(\hat{\boldsymbol{T}}s)_t = s_{t+1}$ for $1 \leq t < n$ and $(\hat{\boldsymbol{T}}s)_n = 0$. The rest of the algorithm exactly matches the standard LSTD and returns a vector θ as the solution of the linear system $A\theta = \gamma B\theta + b$, where $A = \boldsymbol{\Phi}^\top\boldsymbol{\Phi}$, $B = \gamma\boldsymbol{\Phi}^\top\hat{\boldsymbol{T}}\boldsymbol{\Phi}$, and $b = \boldsymbol{\Phi}^\top\boldsymbol{R}$. Similar to the arguments used in Section 6.1, it is easy to verify that the empirical transition matrix $\hat{\boldsymbol{T}}$ results in an empirical Bellman operator which is a contraction and thus that the previous system always admits at least one solution.

Although there exists a unique fixed point, there might be multiple solutions θ. In the following, we use $\hat{\theta}$ to denote the solution with minimal norm, that is $\hat{\theta} = (A - \gamma B)^{+} b$, where $(A - \gamma B)^{+}$ is the Moore-Penrose pseudo-inverse of the matrix $A - \gamma B$.[8]

For the pathwise LSTD-V algorithm the following performance bound has been derived in (Lazaric et al, 2010b).

Theorem 1. ***(Pathwise LSTD-V)*** *Let $\omega > 0$ be the smallest eigenvalue of the Gram matrix $G \in \mathbb{R}^{d \times d}$, $G_{ij} = \int \phi_i^{\top}(x)\phi_j(x)\rho^{\pi}(dx)$. Let assume that the policy π induces a stationary β-mixing process (Meyn and Tweedie, 1993) on the MDP at hand with a stationary distribution ρ^{π} [9]. Let (s_t, r_t) with $t = 1, \ldots, n$ be a path generated by following policy π for $n > n^{\pi}(\omega, \delta)$ steps, where $n^{\pi}(\omega, \delta)$ is a suitable number of steps depending on parameters ω and δ. Let $\hat{\theta}$ be the pathwise LSTD-V solution and $\tilde{f}_{\hat{\theta}}$ be the truncation of its corresponding function, then:* [10]

$$||\tilde{f}_{\hat{\theta}} - V^{\pi}||_{\rho^{\pi}} \leq \frac{2}{\sqrt{1-\gamma^2}} \left(2\sqrt{2}||V^{\pi} - \Pi^{\rho^{\pi}} V^{\pi}||_{\rho^{\pi}} + \tilde{O}\left(L||\theta^*|| \sqrt{\frac{d \log 1/\delta}{n}} \right) \right) + \frac{1}{1-\gamma} \tilde{O}\left(V_{\max} L \sqrt{\frac{d \log 1/\delta}{\omega n}} \right) \tag{32}$$

with probability $1 - \delta$, where θ^* is such that $f_{\theta^*} = \Pi^{\rho^{\pi}} V^{\pi}$.

In the following we analyze the main terms of the previous theorem.

- $||V^{\pi} - \Pi^{\rho^{\pi}} V^{\pi}||_{\rho^{\pi}}$: this term is the *approximation error* and it only depends on the target function V^{π} and how well the functions in $\mathcal{F}$ can approximate it. This can be made small whenever some prior knowledge about V^{π} is available and the BFs are carefully designed. Furthermore, we expect it to decrease with the number d of BFs but at the cost of increasing the other terms in the bound. As it can be noticed, when n goes to infinity, the remaining term in the bound is $\frac{4\sqrt{2}}{\sqrt{1-\gamma^2}}||V^{\pi} - \Pi^{\rho^{\pi}} V^{\pi}||_{\rho^{\pi}}$ which matches the asymptotic performance bound (29) of LSTD up to constants.
- $\tilde{O}\left(V_{\max} L \sqrt{\frac{d \log 1/\delta}{\omega n}} \right)$: this is the first *estimation error* and it is due to bias introduced by the fact that the pathwise LSTD-V solution is computed on a finite number of random samples. As it can be noticed, this term decreases with the number of samples as $n^{-1/2}$ but it also depends on the number of BFs d in $\mathcal{F}$, the range of the BFs L, and the range of the V-functions on the MDP at hand. The

[8] Note that whenever a generic square matrix D is invertible $D^{+} = D^{-1}$.

[9] Roughly speaking, a fast mixing process is such that starting from an arbitrary initial distribution and following the current policy, the state probability distribution rapidly tends to the stationary distribution of the Markov chain.

[10] For simplicity, in the $\tilde{O}$ terms we omit constants, logarithmic factors in d and n, and the dependency on the characteristic parameters of the β-mixing process.

term also shows a critical dependency on the smallest eigenvalue ω of the model-based Gram matrix G. Whenever the BFs are linearly independent and the matrix G is well-conditioned under the stationary distribution ρ^π, ω is guaranteed to be far from 0.

- $\tilde{O}\left(L||\theta^*||\sqrt{\frac{d\log 1/\delta}{n}}\right)$: this is the second *estimation error* and similarly to the previous one, it decreases with the number of samples. This estimation error does not depend on the smallest eigenvalue ω but it has an additional dependency on $||\theta^*||$ which in some cases might add a further dependency on the dimensionality of $\mathscr{F}$ and could be large whenever the BFs are not linear independent under the stationary distribution ρ^π.

The previous bound gives a better understanding of what are the most relevant terms determining the quality of the approximation of LSTD (most notably some properties of the function space). Furthermore, the bound can also be used to have a rough estimate of the number of samples needed to achieve a certain desired approximation quality. For instance, let us assume the approximation error is zero and an error level ε is desired (i.e., $||\tilde{f}_{\hat{\theta}} - V^\pi||_{\rho^\pi}$). In this case, the bound suggests that n should be bigger than $(1-\gamma)^{-2}V_{\max}^2 L^2 d/\varepsilon^2$ (where here we ignore the terms that are not available at design time such as $||\theta^*||$ and ω).

Pathwise LSTD-V strictly requires the samples to be generated by following the current policy π (i.e., it is *on-policy*) and the finite-sample analysis bounds its performance according to the stationary distribution ρ^π. Although an asymptotic analysis for off-policy LSTD exists (see Section 6.1), to the best of our knowledge no finite-sample result is available for off-policy LSTD. The closest result has been obtained for the modified Bellman residual algorithm introduced by Antos et al (2008), which is an off-policy algorithm which reduces to LSTD when linear spaces are used. Nonetheless, the finite-sample analysis reported by Antos et al (2008) does not extend from bounded to linear spaces and how to bound the performance of off-policy LSTD is still an open question.

Bellman Residual Minimization

We now consider the BRM-V algorithm, which finds V-functions. Similar to BRM-Q (see Section 3.2) n samples $(s_i, s'_{i,1}, r_{i,1}, s'_{i,2}, r_{i,2})$ are available, where s_i are drawn i.i.d. from an arbitrary distribution μ, $(s'_{i,1}, r_{i,1})$ and $(s'_{i,2}, r_{i,2})$ are two independent samples drawn from $T(s_i, \pi(s_i), \cdot)$, and $r_{i,1}, r_{i,2}$ are the corresponding rewards. The algorithm works similarly to BRM-Q but now the BFs are functions of the state only. Before reporting the bound on the performance of BRM-V, we introduce:

$$C^\pi(\mu) = (1-\gamma)||(I-\gamma P^\pi)^{-1}||_\mu, \tag{33}$$

which is related to the concentrability coefficient (see e.g. Antos et al (2008)) of the discounted future state distribution starting from μ and following policy π, i.e., $(1-\gamma)\mu(I-\gamma^\pi)^{-1}$ w.r.t. μ. Note that if the discounted future state distribution is

not absolutely continuous w.r.t. μ, then $C^\pi(\mu) = \infty$. We can now report the bound derived by Maillard et al (2010).

Theorem 2. ***(BRM-V)*** *Let $n \geq n^\pi(\omega, \delta)$ be the number of samples drawn i.i.d. according to an arbitrary distribution μ and ω be the smallest eigenvalue of the model-based Gram matrix $G \in \mathbb{R}^{d \times d}$, $G_{ij} = \int \phi_i^\top(x)\phi_j(x)\mu(dx)$. If $\hat{\theta}$ is the BRM-V solution and $f_{\hat{\theta}}$ the corresponding function, then the Bellman residual is bounded as:*

$$||f_{\hat{\theta}} - B_V^\pi f_{\hat{\theta}}||_\mu^2 \leq \inf_{f \in \mathscr{F}} ||f - B_V^\pi f||_\mu^2 + \tilde{O}\left(\frac{1}{\xi_\pi^2} L^4 R_{\max}^2 \sqrt{\frac{d \log 1/\delta}{n}}\right),$$

with probability $1-\delta$, where $\xi_\pi = \frac{\omega(1-\gamma)^2}{C^\pi(\mu)^2}$. Furthermore, the approximation error of V^π is:

$$||V^\pi - f_{\hat{\theta}}||_\mu^2 \leq \left(\frac{C^\pi(\mu)}{1-\gamma}\right)^2 \left((1+\gamma||P^\pi||_\mu)^2 \inf_{f \in \mathscr{F}} ||V^\pi - f||_\mu^2 + \tilde{O}\left(\frac{1}{\xi_\pi^2} L^4 R_{\max}^2 \sqrt{\frac{d \log 1/\delta}{n}}\right)\right).$$

We analyze the first statement of the previous theorem.

- $||f_{\hat{\theta}} - B_V^\pi f_{\hat{\theta}}||_\mu^2$: the left-hand side of the bound is the exact Bellman residual of the BRM-V solution $f_{\hat{\theta}}$. This result is important to show that minimizing the empirical Bellman residual (i.e., the one computed using only the two independent samples $(s'_{i,1}, r_{i,1})$ and $(s'_{i,2}, r_{i,2})$) on a finite number of states s_i leads to a small exact Bellman residual on the whole state space S.
- $\inf_{f \in \mathscr{F}} ||f - B_V^\pi f||_\mu^2$: this term is the smallest Bellman residual that can be achieved with functions in the space $\mathscr{F}$ and it plays the role of the *approximation error*. Although difficult to analyze, it is possible to show that this term can be small for the specific class of MDPs where both the reward function and transition kernel are Lipschitz (further discussion can be found in (Munos and Szepesvári, 2008)).
- $\tilde{O}\left(\frac{1}{\xi_\pi^2} L^4 R_{\max}^2 \sqrt{\frac{d \log 1/\delta}{n}}\right)$: this is the *estimation error*. As it can be noticed, unlike in LSTD bounds, here the *squared* loss is bounded and the estimation error decreases with a slower rate of $O(n^{-1/2})$. Beyond the dependency on some terms characteristic of the MDP and the BFs, this term also depends on the inverse of the smallest eigenvalue of the Gram matrix computed w.r.t. the sampling distribution μ. Although this dependency could be critical, ω can be made big with a careful design of the distribution μ and the BFs.

The second statement does not introduce additional relevant terms in the bound. We postpone the discussion about the term $C^\pi(\mu)$ and a more detailed comparison to LSTD to the next section.

Policy Iteration

Up until now, we reported the performance bounds for pathwise-LSTD-V and BRM-V for policy evaluation. Next, we provide finite-sample analysis for *policy iteration* algorithms, LSPI and BRM-PI, in which at each iteration k, pathwise-LSTD-V and, respectively, BRM-V are used to compute an approximation of V^{π_k}. In particular, we report performance bounds by comparing the value of the policy returned by these algorithms after K iterations, V^{π_K}, and the optimal V-function, V^*. In order to derive performance bounds for LSPI and BRM-PI, one needs to first relate the error after K iterations, $||V^* - V^{\pi_K}||$, to the one at each iteration k of these algorithms, and then replace the error at each iteration with the policy evaluation bounds of Theorems 1 and 2. Antos et al (2008) (Lemma 12) derived the following bound that relates $||V^* - V^{\pi_K}||_\sigma$ to the error at each iteration k

$$||V^* - V^{\pi_K}||_\sigma \leq \frac{4\gamma}{(1-\gamma)^2}\left(\sqrt{C_{\sigma,\nu}} \max_{0\leq k<K} ||V_k - B_V^{\pi_k} V_k||_\nu + \gamma^{\frac{K-1}{2}} R_{\max}\right), \qquad (34)$$

where V_k is the approximation returned by the algorithm of the value function V^{π_k} at each iteration, σ and ν are any measures over the state space, and $C_{\sigma,\nu}$ is the concentrability term from Definition 2 of Antos et al (2008). It is important to note that there is no assumption in this bound on how the sequence V_k is generated. Before reporting the bounds for pathwise-LSPI and BRM-PI, it is necessary to make the following assumptions.

Assumption 1 (Uniform stochasticity) *Let μ be a distribution over the state space. There exists a constant C, such that for all $s \in S$ and for all $a \in A$, $T(s,a,\cdot)$ is uniformly dominated by $\mu(\cdot)$, i.e., $T(s,a,\cdot) \leq C\mu(\cdot)$.*

This assumption guarantees that there is not state-action pair whose transition to the next state is too much concentrated to a limited set of states. It can be easily shown that under this assumption, $C^\pi(\mu)$ defined in Eq. 33 and the concentrability term $C_{\sigma,\mu}$, for any σ, are both upper-bounded by C. We may now report a bound for BRM-PI when Assumption 1 holds for the distribution μ used at each iteration of the algorithm.

Theorem 3. *For any $\delta > 0$, whenever $n \geq n(\delta/K)$ with $n(\delta/K)$ is a suitable number of samples, with probability $1-\delta$, the empirical Bellman residual minimizer f_{θ_k} exists for all iterations $1 \leq k < K$, thus the BRM-PI algorithm is well defined, and the performance V^{π_K} of the policy π_K returned by the algorithm is such that:*

$$||V^* - V^{\pi_K}||_\infty \leq \tilde{O}\left(\frac{1}{(1-\gamma)^2}\left(C^{3/2} E_{BRM}(\mathscr{F}) + \frac{\sqrt{C}}{\xi}\left(\frac{d\log(K/\delta)}{n}\right)^{1/4} + \gamma^{K/2}\right)\right),$$

where $E_{BRM}(\mathscr{F}) = \sup_{\pi\in\mathscr{G}(\mathscr{F})} \inf_{f\in\mathscr{F}} ||f - V^\pi||_\mu$, $\mathscr{G}(\mathscr{F})$ is the set of all greedy policies w.r.t. the functions in $\mathscr{F}$, and $\xi = \frac{\omega(1-\gamma)^2}{C^2} \leq \xi_\pi$, $\pi \in \mathscr{G}(\mathscr{F})$.

In this theorem, $E_{\mathrm{BRM}}(\mathscr{F})$ looks at the smallest approximation error for the V-function of a policy π, and takes the worst case of this error across the set of policies

greedy in some approximate V-function from $\mathcal{F}$. The second term is the estimation error and it contains the same terms as in Theorem 2 and it decreases as $\tilde{O}(d/n)$. Finally, the last term $\gamma^{K/2}$ simply accounts for the error due to the finite number of iterations and it rapidly goes to zero as K increases.

Now we turn to pathwise-LSPI and report a performance bound for this algorithm. At each iteration k of pathwise-LSPI, samples are collected by following a single trajectory generated by the policy under evaluation, π_k, and pathwise-LSTD is used to compute an approximation of V^{π_k}. In order to plug in the pathwise-LSTD bound (Theorem 1) in Eq. 34, one should first note that $||V_k - B_V^{\pi_k} V_k||_{\rho^{\pi_k}} \leq (1+\gamma)||V_k - V^{\pi_k}||_{\rho^{\pi_k}}$. This way instead of the Bellman residual, we bound the performance of the algorithm by using the approximation error $V_k - V^{\pi_k}$ at each iteration. It is also important to note that the pathwise-LSTD bound requires the samples to be collected by following the policy under evaluation. This might introduce severe problems in the sequence of iterations. In fact, if a policy concentrates too much on a small portion of the state space, even if the policy evaluation is accurate on those states, it might be arbitrary bad on the states which are not covered by the current policy. As a result, the policy improvement is likely to generate a bad policy which could, in turn, lead to an arbitrary bad policy iteration process. Thus, the following assumption needs to be made here.

Assumption 2 (Lower-bounding distribution) *Let μ be a distribution over the state space. For any policy π that is greedy w.r.t. a function in the truncated space $\tilde{\mathcal{F}}$, $\mu(\cdot) \leq \kappa \rho^{\pi}(\cdot)$, where $\kappa < \infty$ is a constant and ρ^{π} is the stationary distribution of policy π.*

It is also necessary to guarantee that with high probability a unique pathwise-LSTD solution exists at each iteration of the pathwise-LSPI algorithm, thus, the following assumption is needed.

Assumption 3 (Linear independent BFs) *Let μ be the lower-bounding distribution from Assumption 2. We assume that the BFs $\phi(\cdot)$ of the function space $\mathcal{F}$ are linearly independent w.r.t. μ. In this case, the smallest eigenvalue ω_μ of the Gram matrix $G_\mu \in \mathbb{R}^{d \times d}$ w.r.t. μ is strictly positive.*

Assumptions 2 and 3 (plus some minor assumptions on the characteristic parameters of the β-mixing processes observed during the execution of the algorithm) are sufficient to have a performance bound for pathwise-LSPI. However, in order to make it easier to compare the bound with the one for BRM-PI, we also assume that Assumption 1 holds for pathwise-LSPI.

Theorem 4. *Let us assume that at each iteration k of the pathwise-LSPI algorithm, a path of size $n > n(\omega_\mu, \delta)$ is generated from the stationary β-mixing process with stationary distribution $\rho_{k-1} = \rho^{\pi_{k-1}}$. Let $V_{-1} \in \tilde{\mathcal{F}}$ be an arbitrary initial V-function, $V_0, \ldots, V_{K-1}$ ($\tilde{V}_0, \ldots, \tilde{V}_{K-1}$) be the sequence of V-functions (truncated V-functions) generated by pathwise-LSPI after K iterations, and π_K be the greedy policy w.r.t. the truncated V-function $\tilde{V}_{K-1}$. Then under Assumptions 1–3 and some assumptions on*

the characteristic parameters of the β-mixing processes observed during the execution of the algorithm, with probability $1-\delta$, we have:

$$||V^* - V^{\pi_K}||_\infty \leq \tilde{O}\left(\frac{1}{(1-\gamma)^2}\left(\frac{\sqrt{C\kappa}}{1-\gamma}E_{LSTD}(\mathscr{F}) + \frac{\kappa\sqrt{C}}{\sqrt{\omega_\mu}}\left(\frac{d\log(K/\delta)}{n}\right)^{1/2} + \gamma^{K/2}\right)\right),$$

where $E_{LSTD}(\mathscr{F}) = \sup_{\pi\in\mathscr{G}(\tilde{\mathscr{F}})}\inf_{f\in\mathscr{F}}||f-V^\pi||_{\rho^\pi}$ and $\mathscr{G}(\tilde{\mathscr{F}})$ is the set of all greedy policies w.r.t. the functions in $\tilde{\mathscr{F}}$.

Note that the initial policy π_0 is greedy in the V-function V_{-1}, rather than being arbitrary; that is why we use the index -1 for the initial V-function.

Comparing the performance bounds of BRM-PI and pathwise-LSPI we first notice that BRM-PI has a poorer estimation rate of $O(n^{-1/4})$ instead of $O(n^{-1/2})$. We may also see that the approximation error term in BRM-PI, $E_{\text{BRM}}(\mathscr{F})$, is less complex than that for pathwise-LSPI, $E_{\text{LSTD}}(\mathscr{F})$, as the norm in $E_{\text{BRM}}(\mathscr{F})$ is only w.r.t. the distribution μ while the one in $E_{\text{LSTD}}(\mathscr{F})$ is w.r.t. the stationary distribution of any policy in $\mathscr{G}(\tilde{\mathscr{F}})$. The assumptions used by the algorithms are also different. In BRM-PI, it is assumed that a generative model is available, and thus, the performance bounds may be obtained under any sampling distribution μ, specifically the one for which Assumption 1 holds. On the other hand, at each iteration of pathwise-LSPI, it is required to use a single trajectory generated by following the policy under evaluation. This can provide performance bounds only under the stationary distribution of that policy, and accurately approximating the current policy under the stationary distribution may not be enough in a policy iteration scheme, because the greedy policy w.r.t. that approximation may be arbitrarily poor. Therefore, we may conclude that the performance of BRM-PI is better controlled than that of pathwise-LSPI. This is reflected in the fact that the concentrability terms may be controlled in the BRM-PI by only choosing a uniformly dominating distribution μ (Assumption 1), such as a uniform distribution, while in pathwise-LSPI, we are required to make stronger assumptions on the stationary distributions of the policies encountered at the iterations of the algorithm, such as being lower-bounded by a uniform distribution (Assumption 2).

7 Further Reading

Many extensions and variations of the methods introduced in Section 3 have been proposed, and we unfortunately do not have the space to describe them all. Instead, in the present section, we will (non-exhaustively) touch upon some of the highlights in this active field of research, providing pointers to the literature for the reader interested in more details.

As previously mentioned in Footnote 2, variants of approximate policy evaluation that employ a *multistep* Bellman mapping can be used. This mapping is parameter-

ized by $\lambda \in [0,1)$, and is given, e.g., in the case of Q-functions by:

$$B^{\pi}_{Q,\lambda}(Q) = (1-\lambda)\sum_{t=0}^{\infty}\lambda^t (B^{\pi}_Q)^{t+1}(Q)$$

where $(B^{\pi}_Q)^t$ denotes the t-times composition of B^{π}_Q with itself. In this chapter, we only considered the single-step case, in which $\lambda = 0$, but in fact approximate policy evaluation is often discussed in the general $\lambda \in [0,1)$ case, see e.g. Nedić and Bertsekas (2003); Yu (2010); Bertsekas and Ioffe (1996); Yu and Bertsekas (2009) and also the discussion of the so-called TD(λ) algorithm in Chapter 2. Note that in the original LSPI, a nonzero λ would prevent sample reuse; instead, at every iteration, new samples would have to be generated with the current policy. However, Thiery and Scherrer (2010) recently introduced an extension of LSPI to general $\lambda \in [0,1)$, which does not impose this requirement.

Nonparametric approximators are not predefined, but are automatically constructed from the data, so to a large extent they free the user from the difficult task of designing the BFs. A prominent class of nonparametric techniques is kernel-based approximation, which was combined, e.g., with LSTD by Xu et al (2005, 2007); Jung and Polani (2007b); Farahmand et al (2009), with LSPE by Jung and Polani (2007a,b), and with BRM by Farahmand et al (2009). The related framework of Gaussian processes has also been used in policy evaluation (Rasmussen and Kuss, 2004; Engel et al, 2005; Taylor and Parr, 2009). In their basic form, the computational demands of kernel-based methods and Gaussian processes grow with the number of samples considered. Since this number can be large in practice, many of the approaches mentioned above employ kernel sparsification techniques to limit the number of samples that contribute to the solution (Xu et al, 2007; Engel et al, 2003, 2005; Jung and Polani, 2007a,b).

Closely related to sparsification is the technique of *regularization*, which controls the complexity of the solution (Farahmand et al, 2009; Kolter and Ng, 2009). For instance, to obtain a regularized variant of LSTD, a penalty term can be added to the projected policy evaluation problem (8) to obtain:

$$\min_{\widehat{Q}\in\widehat{\mathcal{Q}}} \left[\left\| \widehat{Q} - \Pi(B^{\pi}_Q(\widehat{Q})) \right\| + \beta \nu(\widehat{Q}) \right]$$

where β is a positive regularization coefficient and ν is a penalty function that grows with the complexity of $\widehat{Q}$ (e.g., a norm of the approximator's parameter vector). Note that Farahmand et al (2009) also add a similar regularization term to the projection Π, and additionally discusses a regularized version of BRM.

In an effort to reduce the computational costs of LSTD, so-called *incremental* variants have been proposed, in which only a few of the parameters are updated at a given iteration (Geramifard et al, 2006, 2007).

A generalized, *scaled* variant of LSPE, discussed e.g. by Bertsekas (2010b), can be obtained by first rewriting the LSPE update (20) into the equivalent form (assuming A is invertible):

$$\theta_{\tau+1} = \theta_\tau - \alpha A^{-1}[(A - \gamma B)\theta_\tau - b]$$

and then replacing A^{-1} by a more general scaling matrix Γ:

$$\theta_{\tau+1} = \theta_\tau - \alpha \Gamma[(A - \gamma B)\theta_\tau - b]$$

For appropriate choices of Γ and α, this algorithm converges under more general conditions than the original LSPE.

In the context of BRM, Antos et al (2008) eliminated the requirement of double sampling by modifying the minimization problem solved by BRM. They showed that single-sample estimates of the coefficients in this modified problem are unbiased, while the solution stays meaningful.

We also note active research into alternatives to least-squares methods for policy evaluation and iteration, in particular, techniques based on gradient updates (Sutton et al, 2009b,a; Maei et al, 2010) and on Monte Carlo simulations (Lagoudakis and Parr, 2003b; Dimitrakakis and Lagoudakis, 2008; Lazaric et al, 2010a).

While an extensive tutorial such as this one, focusing on a unified view and theoretical study of policy iteration with LSTD, LSPE, and BRM policy evaluation, has not been available in the literature until now, these methods have been treated in various recent books and surveys on reinforcement learning and dynamic programming. For example, the interested reader should know that Chapter 3 of (Buşoniu et al, 2010a) describes LSTD-Q and LSPE-Q as part of an introduction to approximate reinforcement learning and dynamic programming, that (Munos, 2010) touches upon LSTD and BRM, that Chapter 3 of (Szepesvári, 2010) outlines LSTD and LSPE methods, and that (Bertsekas, 2010b,a) concern to a large extent these two types of methods.

Chapter 8 of this book presents a more general view over the field of approximate reinforcement learning, without focusing on least-squares methods.

References

Antos A, Szepesvári Cs, Munos R (2008) Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. Machine Learning 71(1):89–129

Baird L (1995) Residual algorithms: Reinforcement learning with function approximation. In: Proceedings 12th International Conference on Machine Learning (ICML-95), Tahoe City, US, pp 30–37

Bertsekas DP (1995) A counterexample to temporal differences learning. Neural Computation 7:270–279

Bertsekas DP (2010a) Approximate dynamic programming, update of Chapter 6 in volume 2 of the book Dynamic Programming and Optimal Control. Available at http://web.mit.edu/dimitrib/www/dpchapter.html

Bertsekas DP (2010b) Approximate policy iteration: A survey and some new methods. Tech. Rep. LIDS 2833, Massachusetts Institute of Technology, Cambridge, US

Bertsekas DP (2011) Temporal difference methods for general projected equations. IEEE Transactions on Automatic Control 56, to appear.

Bertsekas DP, Ioffe S (1996) Temporal differences-based policy iteration and applications in neuro-dynamic programming. Tech. Rep. LIDS-P-2349, Massachusetts Institute of Technology, Cambridge, US, available at http://web.mit.edu/dimitrib/www/Tempdif.pdf

Bertsekas DP, Tsitsiklis JN (1996) Neuro-Dynamic Programming. Athena Scientific

Bertsekas DP, Borkar V, Nedić A (2004) Improved temporal difference methods with linear function approximation. In: Si J, Barto A, Powell W (eds) Learning and Approximate Dynamic Programming, IEEE Press

Boyan J (2002) Technical update: Least-squares temporal difference learning. Machine Learning 49:233–246

Bradtke SJ, Barto AG (1996) Linear least-squares algorithms for temporal difference learning. Machine Learning 22(1–3):33–57

Buşoniu L, Babuška R, De Schutter B, Ernst D (2010a) Reinforcement Learning and Dynamic Programming Using Function Approximators. Automation and Control Engineering, Taylor & Francis CRC Press

Buşoniu L, De Schutter B, Babuška R, Ernst D (2010b) Using prior knowledge to accelerate online least-squares policy iteration. In: 2010 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR-10), Cluj-Napoca, Romania

Buşoniu L, Ernst D, De Schutter B, Babuška R (2010c) Approximate dynamic programming with a fuzzy parameterization. Automatica 46(5):804–814

Buşoniu L, Ernst D, De Schutter B, Babuška R (2010d) Online least-squares policy iteration for reinforcement learning control. In: Proceedings 2010 American Control Conference (ACC-10), Baltimore, US, pp 486–491

Dimitrakakis C, Lagoudakis M (2008) Rollout sampling approximate policy iteration. Machine Learning 72(3):157–171

Engel Y, Mannor S, Meir R (2003) Bayes meets Bellman: The Gaussian process approach to temporal difference learning. In: Proceedings 20th International Conference on Machine Learning (ICML-03), Washington, US, pp 154–161

Engel Y, Mannor S, Meir R (2005) Reinforcement learning with Gaussian processes. In: Proceedings 22nd International Conference on Machine Learning (ICML-05), Bonn, Germany, pp 201–208

Ernst D, Geurts P, Wehenkel L (2005) Tree-based batch mode reinforcement learning. Journal of Machine Learning Research 6:503–556

Farahmand AM, Ghavamzadeh M, Szepesvári Cs, Mannor S (2009) Regularized policy iteration. In: Koller D, Schuurmans D, Bengio Y, Bottou L (eds) Advances in Neural Information Processing Systems 21, MIT Press, pp 441–448

Geramifard A, Bowling MH, Sutton RS (2006) Incremental least-squares temporal difference learning. In: Proceedings 21st National Conference on Artificial Intelligence and 18th Innovative Applications of Artificial Intelligence Conference (AAAI-06), Boston, US, pp 356–361

Geramifard A, Bowling M, Zinkevich M, Sutton RS (2007) iLSTD: Eligibility traces & convergence analysis. In: Schölkopf B, Platt J, Hofmann T (eds) Advances in Neural Information Processing Systems 19, MIT Press, pp 440–448

Golub GH, Van Loan CF (1996) Matrix Computations, 3rd edn. Johns Hopkins

Jung T, Polani D (2007a) Kernelizing LSPE(λ). In: Proceedings 2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL-07), Honolulu, US, pp 338–345

Jung T, Polani D (2007b) Learning RoboCup-keepaway with kernels. In: Gaussian Processes in Practice, JMLR Workshop and Conference Proceedings, vol 1, pp 33–57

Kolter JZ, Ng A (2009) Regularization and feature selection in least-squares temporal difference learning. In: Proceedings 26th International Conference on Machine Learning (ICML-09), Montreal, Canada, pp 521–528

Konda V (2002) Actor-critic algorithms. PhD thesis, Massachusetts Institute of Technology, Cambridge, US

Lagoudakis M, Parr R, Littman M (2002) Least-squares methods in reinforcement learning for control. In: Methods and Applications of Artificial Intelligence, Lecture Notes in Artificial Intelligence, vol 2308, Springer, pp 249–260

Lagoudakis MG, Parr R (2003a) Least-squares policy iteration. Journal of Machine Learning Research 4:1107–1149

Lagoudakis MG, Parr R (2003b) Reinforcement learning as classification: Leveraging modern classifiers. In: Proceedings 20th International Conference on Machine Learning (ICML-03), Washington, US, pp 424–431

Lazaric A, Ghavamzadeh M, Munos R (2010a) Analysis of a classification-based policy iteration algorithm. In: Proceedings 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, pp 607–614

Lazaric A, Ghavamzadeh M, Munos R (2010b) Finite-sample analysis of LSTD. In: Proceedings 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, pp 615–622

Li L, Littman ML, Mansley CR (2009) Online exploration in least-squares policy iteration. In: Proceedings 8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-09), Budapest, Hungary, vol 2, pp 733–739

Maei HR, Szepesvári C, Bhatnagar S, Sutton RS (2010) Toward off-policy learning control with function approximation. In: Proceedings 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, pp 719–726

Maillard OA, Munos R, Lazaric A, Ghavamzadeh M (2010) Finite-sample analysis of Bellman residual minimization. vol 13, pp 299–314

Meyn S, Tweedie L (1993) Markov chains and stochastic stability. Springer–Verlag

Moore AW, Atkeson CR (1995) The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. Machine Learning 21(3):199–233

Munos R (2003) Error bounds for approximate policy iteration. In: Proceedings 20th International Conference (ICML-03), Washington, US, pp 560–567

Munos R (2010) Approximate dynamic programming. In: Markov Decision Processes in Artificial Intelligence, Wiley

Munos R, Szepesvári Cs (2008) Finite time bounds for fitted value iteration. Journal of Machine Learning Research 9:815–857

Nedić A, Bertsekas DP (2003) Least-squares policy evaluation algorithms with linear function approximation. Discrete Event Dynamic Systems: Theory and Applications 13(1–2):79–110

Rasmussen CE, Kuss M (2004) Gaussian processes in reinforcement learning. In: Thrun S, Saul LK, Schölkopf B (eds) Advances in Neural Information Processing Systems 16, MIT Press

Scherrer B (2010) Should one compute the Temporal Difference fix point or minimize the Bellman Residual? the unified oblique projection view. In: Proceedings 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, pp 959–966

Schweitzer PJ, Seidmann A (1985) Generalized polynomial approximations in Markovian decision processes. Journal of Mathematical Analysis and Applications 110(2):568–582

Sutton R, Maei H, Precup D, Bhatnagar S, Silver D, Szepesvari Cs, Wiewiora E (2009a) Fast gradient-descent methods for temporal-difference learning with linear function approximation. In: Proceedings 26th International Conference on Machine Learning (ICML-09), Montreal, Canada, pp 993–1000

Sutton RS (1988) Learning to predict by the method of temporal differences. Machine Learning 3:9–44

Sutton RS, Szepesvári Cs, Maei HR (2009b) A convergent O(n) temporal-difference algorithm for off-policy learning with linear function approximation. In: Koller D, Schuurmans D, Bengio Y, Bottou L (eds) Advances in Neural Information Processing Systems 21, MIT Press, pp 1609–1616

Szepesvári Cs (2010) Algorithms for Reinforcement Learning. Morgan & Claypool Publishers

Taylor G, Parr R (2009) Kernelized value function approximation for reinforcement learning. In: Proceedings 26th International Conference on Machine Learning (ICML-09), Montreal, Canada, pp 1017–1024

Thiery C, Scherrer B (2010) Least-squares λ policy iteration: Bias-variance trade-off in control problems. In: Proceedings 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, pp 1071–1078

Tsitsiklis JN (2002) On the convergence of optimistic policy iteration. Journal of Machine Learning Research 3:59–72

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

Xu X, Xie T, Hu D, Lu X (2005) Kernel least-squares temporal difference learning. International Journal of Information Technology 11(9):54–63

Xu X, Hu D, Lu X (2007) Kernel-based least-squares policy iteration for reinforcement learning. IEEE Transactions on Neural Networks 18(4):973–992

Yu H (2010) Convergence of least squares temporal difference methods under general conditions. In: Proceedings 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, pp 1207–1214

Yu H, Bertsekas DP (2009) Convergence results for some temporal difference methods based on least squares. IEEE Transactions on Automatic Control 54(7):1515–1531

Yu H, Bertsekas DP (2010) Error bounds for approximations from projected linear equations. Mathematics of Operations Research 35(2):306–329

Index