Least-Squares λ Policy Iteration: Bias-Variance Trade-off in Control Problems

Christophe Thiery, Bruno Scherrer

Least-Squares λ Policy Iteration: Bias-Variance Trade-off in Control Problems

Christophe Thiery THIERYCH@LORIA.FR
Bruno Scherrer SCHERRER@LORIA.FR
LORIA - INRIA Lorraine - Campus Scientifique - BP 239
54506 Vandœuvre-lès-Nancy CEDEX - FRANCE

Abstract

In the context of large space MDPs with linear value function approximation, we introduce a new approximate version of λ-Policy Iteration (Bertsekas & Ioffe, 1996), a method that generalizes Value Iteration and Policy Iteration with a parameter $\lambda \in (0, 1)$. Our approach, called Least-Squares λ Policy Iteration, generalizes LSPI (Lagoudakis & Parr, 2003) which makes efficient use of training samples compared to classical temporal-differences methods. The motivation of our work is to exploit the λ parameter within the least-squares context, and without having to generate new samples at each iteration or to know a model of the MDP. We provide a performance bound that shows the soundness of the algorithm. We show empirically on a simple chain problem and on the Tetris game that this λ parameter acts as a bias-variance trade-off that may improve the convergence and the performance of the policy obtained.

1. Introduction

We consider the question of solving Markov Decision Processes (MDPs) approximately, in the case where the value function is estimated by a linear architecture and using sampling, typically when the state space is too large for an exact solution.

TD(λ) with linear approximation (Sutton & Barto, 1998) is a fundamental algorithm of the reinforcement learning community. It estimates the value function by using temporal differences based on a sample trajectory and the λ parameter controls the length of the backups made to update the value function. **Least-squares** (LS) methods such as LSTD(0) (Bradtke & Barto, 1996), LSTD(λ) (Boyan, 2002) and LSPE(λ) (Nedić & Bertsekas, 2003; Yu & Bertsekas, 2009) are alternative approaches for estimating the value function. They build explicitly a low-dimensional linear system that characterizes the solution TD(λ) would converge to, but they use simulation data more efficiently, and they usually don't need a decreasing stepsize parameter which is often hard to tune. Moreover, they converge faster in practice, though each iteration has a complexity of p^2 instead of p, where p is the dimension of the linear architecture. It was shown empirically (see Boyan (2002); Yu & Bertsekas (2009); Lagoudakis et al. (2002)) that LS methods are more stable and can succeed much better than TD(λ).

LSTD(λ) and LSPE(λ) focus on the problem of evaluating a policy. We are interested in this article in learning a policy. The λ-Policy Iteration algorithm (λPI), proposed by Bertsekas & Ioffe (1996), iterates over policies and uses a λ paramater that introduces a **bias-variance trade-off** similar to the one of TD(λ) when sampling is used to evaluate the current policy. λPI generalizes the classical Dynamic Programming (DP) algorithms Value Iteration and Policy Iteration to learn a policy: λ represents the size of the step made toward the value function of the policy ($\lambda = 0$ corresponds to Value Iteration and $\lambda = 1$ corresponds to Policy Iteration). One the one hand, when λ is close to 1, the variance of the value function estimation may deteriorate the eventual performance of the obtained policy. On the other hand, reducing λ introduces some kind of **optimism**, since we do not try to compute completely the value of the current policy before we switch to a new policy, and such a bias can degrade the convergence speed. Thus, this parameter λ creates a bias-variance trade-off similar to that of TD(λ), and in practice, an intermediate value between 0 and 1 can

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

Algorithm	bias-variance trade-off	sample efficient (LS)	optimistic evaluation	off-policy
TD(λ) (Sutton & Barto, 1998)	×			
LSTD(0) (Bradtke & Barto, 1996)		×		
LSTD(λ) (Boyan, 2002)	×	×		
LSPE(λ) (Yu & Bertsekas, 2009)	×	×		
λPI (Bertsekas & Ioffe, 1996)	×	×	×	
LSPI (Lagoudakis & Parr, 2003)		×		×
LSλPI	×	×	×	×

Figure 1. Main characteristics of related works: a trade-off parameter λ, the LS context, the optimistic evaluation of the value function and the off-policy evaluation.

give the best results (Bertsekas & Ioffe, 1996).

When they use a λ trade-off parameter, the approaches mentioned above make an on-policy estimation of the value function, that is, they need samples generated by the policy under evaluation. The **off-policy** evaluation can be used in a policy iteration context without having to generate new samples when the policy changes. This idea is developed in LSPI (Lagoudakis & Parr, 2003). The difference with our approach is the fact that LSPI does not make an optimistic evaluation of the policy using a λ parameter.

In this article, we introduce Least-Squares λ Policy Iteration (LSλPI for short), a generalization of LSPI that adds the λ parameter of λPI. LSλPI is to our knowledge the first algorithm that includes all interesting characteristics discussed above: the bias-variance trade-off, the LS framework, the optimistic evaluation of the value function and the off-policy evaluation. Figure 1 lists the related works and summarizes their characteristics to show the main contributions of LSλPI. Overall, LSλPI can be seen as an optimistic generalization of LSPI with a λ parameter, an off-policy, LS implementation of λPI, or as an off-policy, optimistic policy iteration variant of LSPE(λ).

The paper is organized as follows. Section 2 presents the notations and the λPI algorithm in the exact case. In section 3, we consider the approximate case and detail LSλPI. Section 4 finally gives some experimental results showing the interest of this parameter λ.

2. Exact case: λPI

We introduce here the notations we will use and present an overview of λPI in the exact case. The reader can refer to Bertsekas & Ioffe (1996) for a more detailed description of the λPI method.

We define an MDP as a tuple $(\mathcal{S}, \mathcal{A}, P, R)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P is the transition function ($P(s,a,s')$ is the probability of getting to state s' after taking action a from state s) and R is the reward function: $R(s,a,s') \in \mathbb{R}$ is the reward received in state s' after taking action a from state s. We will use the simplified notation $\mathcal{R}(s,a)$ to denote the expected reward of a state-action pair: $\mathcal{R}(s,a) = \sum_{s' \in S} P(s,a,s')R(s,a,s')$.

A policy is a mapping from $\mathcal{S}$ to $\mathcal{A}$. The value function of a policy π is the function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that associates to each state-action pair (s,a) the expected value of the discounted, cumulative reward one can get from state s, taking action a and following policy π then: $Q^\pi(s,a) = E_\pi \left[\sum_{t \geq 0} \gamma^t r_t \middle| s_0 = s, a_0 = a \right]$ where E_π denotes the expected value with respect to the policy π, r_t is the reward received at time t and $\gamma \in (0,1)$ is a discount factor[1]. For any vector Q, let B_π be the Bellman operator defined by $B_\pi Q = \mathcal{R} + \gamma P_\pi Q$ where $\mathcal{R}$ is the reward vector and P_π is the $|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}||\mathcal{A}|$ transition matrix induced by the choice of an action and the policy π then. It is well known (see for instance Puterman, 1994) that this operator is a contraction mapping of modulus γ and that its only fixed point is Q^π. Thus, the value function Q^π is the solution of the Bellman equation $Q = B^\pi Q$. Let us also denote by Q^* the optimal value function, that is, $Q^*(s,a) = \max_\pi Q^\pi(s,a)$. If the optimal value function is known, it is straightforward to deduce an optimal policy by taking a greedy policy with respect to Q^*: $\pi^*(s) \in \arg\max_a Q^*(s,a)$.

We now present λPI, a method introduced by Bertsekas & Ioffe (1996) that generalizes the standard DP algorithms Value Iteration (VI) and Policy Iteration (PI). A parameter $\lambda \in (0,1)$ specifies whether the algorithm is closer to PI ($\lambda = 1$) or VI ($\lambda = 0$). At each iteration, λPI (see Algorithm 1) computes a value function Q_{k+1} from the greedy policy (π_{k+1}) with respect to the previous value function Q_k. The value function update corresponds to a λ-geometric average of the terms $(B_{\pi_{k+1}})^i Q_k$. The bigger λ, the more the

[1] In this paper, we only consider value functions defined over the state-action space because we are interested in learning a policy without a model of the MDP; however, our description of λPI remains valid for value functions defined over the state space.

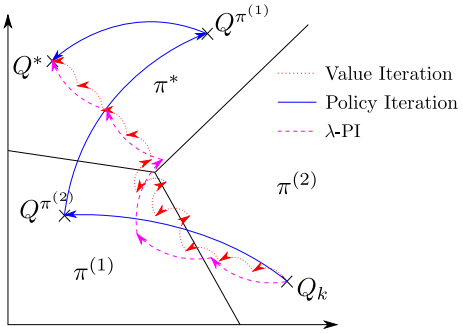

Figure 2. **Visualizing λPI on the greedy partition sketch:** Following Bertsekas & Tsitsiklis (1996, page 226), one can decompose the value space as a collection of polyhedra, such that each polyhedron corresponds to a region where the greedy policy remains the same. PI generates a one-step move toward $Q^{\pi_{k+1}}$ whereas VI makes small steps toward $Q^{\pi_{k+1}}$. λPI is intermediate between PI and VI: it makes a λ-*adjustable step* toward $Q^{\pi_{k+1}}$.

value function gets close to $Q^{\pi_{k+1}}$. An intuitive view of the value function updates made by VI, PI and λPI is given in the greedy partition represented on Figure 2.

Algorithm 1 λ-Policy Iteration

loop

$\pi_{k+1} \leftarrow \text{greedy}(Q_k)$

$Q_{k+1} \leftarrow (1-\lambda)\sum_{i\geq 1} \lambda^{i-1}(B_{\pi_{k+1}})^i Q_k$

end loop

Bertsekas & Ioffe (1996) introduced an operator denoted M_k that gives an alternative formulation of the algorithm. At each iteration k, this operator is defined as follows:

$$M_k Q = (1-\lambda) B_{\pi_{k+1}} Q_k + \lambda B_{\pi_{k+1}} Q. \qquad (1)$$

Intuitively, M_k can be seen as a **damped** application of $B_{\pi_{k+1}}$. They showed that M_k is a contraction mapping of modulus $\gamma\lambda$ and that its only fixed point is the value Q_{k+1} computed by λPI. Then, to calculate Q_{k+1} in practice, one can make successive applications of this M_k operator until the fixed point is obtained.

Bertsekas & Ioffe (1996) also showed that the value function update can be seen as an incremental update $Q_{k+1} \leftarrow Q_k + \Delta_k$, where

$$\Delta_k(s_0, a_0) = E_{\pi_{k+1}}\left[\sum_{n=0}^{\infty} (\lambda\gamma)^n \delta_k(s_n, a_n, s_{n+1})\right] \qquad (2)$$

with $\delta_k(s,a,s') = R(s,a,s') + \gamma Q(s', \pi_{k+1}(s')) - Q(s,a)$. Equation (2) gives some insight on the variance reduction property of small values of λ. Indeed, one can see that the policy evaluation phase may be easier when $\lambda < 1$ because the horizon of the sum estimated is reduced. However, λPI is known to converge with an asymptotic speed that depends on λ (Bertsekas & Ioffe, 1996): small values of λ deteriorate the asymptotic convergence speed because they introduce a bias in the evaluation (the algorithm does not estimate the true value function of the policy anymore). This drawback is less important when using approximation and simulation since the asymptotic convergence speed is not a crucial factor.

3. Approximate case: LSλPI

We have presented λPI in the exact case. However, λPI was designed for a context of approximation and sampling. The algorithm we introduce now, called LSλPI, is an approximate version of λPI which reduces to LSPI (Lagoudakis & Parr, 2003) when $\lambda = 1$.

LSλPI is an iterative, sampling-based algorithm that computes at each iteration an estimation of the value function of the current policy, and then uses this approximation to improve the policy. As it uses value functions defined over the state-action space, knowing a model of the MDP is not required. As LSPI, LSλPI is an off-policy algorithm: the policy under evaluation may be different from the one used to generate samples. Thus, a unique sample set can be reused for all iterations even if the policy changes.

Approximation architecture

LSλPI relies on a usual linear representation of the value function. At each iteration k, we consider a policy π_k and an approximate value function $\widehat{Q}_k$ that estimates the Q_k that the exact version of λPI would compute. The new policy π_{k+1} is the greedy policy with respect to $\widehat{Q}_k$, then we approximate Q_{k+1} by a linear combination of basis functions:

$$\widehat{Q}_{k+1}(s,a) = \sum_{i=1}^{p} \phi_i(s,a) w_i.$$

The ϕ_i terms are p arbitrary basis functions and the w_i are the parameters of this architecture. As in general $p \ll |\mathcal{S}||\mathcal{A}|$ when the state space is large, such a value function requires much less resources than an exact (tabular) representation. If we denote by $\phi(s,a)$ the column vector of size p whose elements are the basis functions applied to (s,a), and Φ the $|\mathcal{S}||\mathcal{A}| \times p$ matrix composed of the transpose of all these vectors, we can write $\widehat{Q}_{k+1} = \Phi w_{k+1}$, where w_{k+1} is the parameter vector representing the approximate value function $\widehat{Q}_{k+1}$.

To compute w_{k+1}, LSλPI can use two standard methods. Those methods are described for instance in Lagoudakis & Parr (2003) and we generalize them to LSλPI. They both try to compute an approximate value function $\widehat{Q}_{k+1}$ which approaches the vector Q_{k+1}. In other words, they look for a $\widehat{Q}_{k+1}$ such that $M_k\widehat{Q}_{k+1} \simeq \widehat{Q}_{k+1}$. When $\lambda = 1$, we have $M_k = B_{\pi_{k+1}}$ and the resulting algorithm, which approximates the Bellman equation's solution, matches LSPI.

3.1. Fixed-point method (FP)

As the fixed-point equation $\widehat{Q}_{k+1} = M_k\widehat{Q}_{k+1}$ has no solution in general since $M_k\widehat{Q}_{k+1}$ is not necessarily in the space induced by the basis functions, the fixed-point method (FP) applies an orthogonal projection to it. This means that we seek an approximate value function $\widehat{Q}_{k+1}$ that verifies

$$\widehat{Q}_{k+1} = \Phi(\Phi^{\mathrm{T}} D_\mu \Phi)^{-1}\Phi^{\mathrm{T}} D_\mu M_k \widehat{Q}_{k+1}. \qquad (3)$$

D_μ represents the diagonal matrix of size $|\mathcal{S}||\mathcal{A}|$ whose terms are the projection weights, denoted by $\mu(s,a)$. μ is a probability distribution over $\mathcal{S}\times\mathcal{A}$. By developping Equation (3), it can be verified that w_{k+1} is the solution of the linear system $Aw = b$ of size $p \times p$ with

$$\begin{aligned} A &= \Phi^{\mathrm{T}} D_\mu (\Phi - \lambda\gamma P_{\pi_{k+1}}\Phi) \quad \text{and} \\ b &= \Phi^{\mathrm{T}} D_\mu (\mathcal{R} + (1-\lambda)\gamma P_{\pi_{k+1}}\Phi w_k). \end{aligned}$$

When the number of states is large, A and b cannot be computed directly even if a model of the MDP is available. We can estimate them by using a set of L samples of the form (s,a,r',s') where $(s,a) \sim \mu$, $s' \sim P(s,a,\cdot)$ and $r' = R(s,a,s')$. It can be seen that

$$\begin{aligned} A &= E\Big[\phi(s,a)(\phi(s,a) - \lambda\gamma\phi(s',\pi_{k+1}(s')))^{\mathrm{T}}\Big] \\ \text{and } b &= E\Big[\phi(s,a)(r' + (1-\lambda)\gamma\phi(s',\pi_{k+1}(s'))^{\mathrm{T}} w_k\Big]. \end{aligned}$$

Let us denote by $\widetilde{A}$ and $\widetilde{b}$ the sampling-based estimations of A and b. For each sample (s,a,r',s') considered, we can update $\widetilde{A}$ and $\widetilde{b}$ as follows

$$\begin{aligned} \widetilde{A} &\leftarrow \widetilde{A} + \frac{1}{L}\phi(s,a)\left(\phi(s,a) - \lambda\gamma\phi(s',\pi_{k+1}(s'))\right)^{\mathrm{T}}, \\ \widetilde{b} &\leftarrow \widetilde{b} + \frac{1}{L}\phi(s,a)\left(r' + (1-\lambda)\gamma\phi(s',\pi_{k+1}(s'))^{\mathrm{T}} w_k\right). \end{aligned}$$

To simplify these update rules, as we intend to solve the linear system $\widetilde{A}w = \widetilde{b}$, we can remove the $\frac{1}{L}$ factor in the $\widetilde{A}$ and $\widetilde{b}$ updates without changing the solution. Once we have estimated A and b from a sample source, we solve the linear $p \times p$ system $\widetilde{A}w = \widetilde{b}$ to obtain the new parameter vector w_{k+1}. As in Lagoudakis & Parr (2003), it is possible to update $\widetilde{A}^{-1}$ incrementally by using the Sherman-Morisson formula (starting with an initial estimate βI for a small β), instead of solving the system after each sample:

$$\widetilde{A}^{-1} \leftarrow \widetilde{A}^{-1} + \frac{\widetilde{A}^{-1}uv^{\mathrm{T}}\widetilde{A}^{-1}}{1 + v^{\mathrm{T}}\widetilde{A}^{-1}u} \qquad (4)$$

where $u = \phi(s,a)$ and $v = \phi(s,a) - \lambda\gamma\phi(s',\pi_{k+1}(s'))$. The weight vector w_{k+1} is then computed as $\widetilde{A}^{-1}\widetilde{b}$, which reduces the complexity to p^2 instead of p^3.

3.2. Bellman residual minimization (BR)

To calculate the approximate value function $\widehat{Q}_{k+1}$, an alternative to FP is the Bellman residual minimization method (BR). Let us consider the (generalized) Bellman equation $Q_{k+1} = M_k Q_{k+1}$ and the corresponding *residual* defined by $Q_{k+1} - M_k Q_{k+1}$. We seek a value function $\widehat{Q}_{k+1}$ that minimizes the quadratic, weighted norm of this residual $\|\widehat{Q}_{k+1} - M_k\widehat{Q}_{k+1}\|_{\mu,2}$ where $\|\cdot\|_{\mu,2}$ denotes the quadratic norm weighted by the distribution μ: $\|Q\|_{\mu,2} = \sqrt{\sum_{s,a}\mu(s,a)Q(s,a)^2}$. It can be verified that the norm to minimize equals

$$\|(\Phi - \lambda\gamma P_{\pi_{k+1}}\Phi)w_{k+1} - \mathcal{R} - (1-\lambda)\gamma P_{\pi_{k+1}}\Phi w_k\|_{\mu,2}.$$

Therefore, by a standard least-squares analysis, the parameter vector w_{k+1} that minimizes the Bellman residual is the solution of the $p \times p$ linear system $Aw = b$ with

$$\begin{aligned} A &= (\Phi - \lambda\gamma P_{\pi_{k+1}}\Phi)^{\mathrm{T}} D_\mu (\Phi - \lambda\gamma P_{\pi_{k+1}}\Phi) \quad \text{and} \\ b &= (\Phi - \lambda\gamma P_{\pi_{k+1}}\Phi)^{\mathrm{T}} D_\mu (\mathcal{R} + (1-\lambda)\gamma P_{\pi_{k+1}}\Phi w_k). \end{aligned}$$

As in the case of FP, A and b can be estimated from a set of samples whose distribution corresponds to μ. However, BR requires for each sample to generate two possible next independant states instead of only one. This requirement is known for $\lambda = 1$ (Sutton & Barto, 1998) and also applies for $\lambda > 0$. Let us denote by (s,a,r',s',s'') a sample, where s' and s'' are the results of two independent realizations of the action a from the state s, and $r' = R(s,a,s')$ (the reward of s'' is not needed). Again, if (s,a) is drawn from μ, s' and s'' from $P(s,a,\cdot)$, and $r' = R(s,a,\cdot)$, we have

$$\begin{aligned} A &= E\Big[(\phi(s,a) - \lambda\gamma\phi(s'',\pi_{k+1}(s'')))\\ &\qquad (\phi(s,a) - \lambda\gamma\phi(s',\pi_{k+1}(s')))^{\mathrm{T}}\Big] \\ \text{and } b &= E\Big[(\phi(s,a) - \lambda\gamma\phi(s'',\pi_{k+1}(s'')))\\ &\qquad (r' + (1-\lambda)\gamma\phi(s',\pi_{k+1}(s'))^{\mathrm{T}} w_k)\Big] \end{aligned}$$

As in FP, it is straightforward from these equations to build estimations $\widetilde{A}$ and $\widetilde{b}$ from the samples, by using similar incremental update rules. We can solve the $p \times p$ linear system $\widetilde{A}w = \widetilde{b}$ to get the new value function after each sample, or incrementally update $\widetilde{A}^{-1}$ using the Sherman-Morisson formula: this update is identical to that of FP (Equation 4) except that in this case $u = \phi(s,a) - \lambda\gamma\phi(s'', \pi_{k+1}(s''))$.

3.3. Discussion

For both methods, if a model of the MDP is available, we can integrate it in the update rule: the samples are then reduced to state-action pairs (s,a) and we replace the successor terms $\phi(s', \pi_{k+1}(s'))$ and r' by their respective expected values $\sum_{s' \in \mathcal{S}} P(s,a,s')\phi(s', \pi_{k+1}(s'))$ and $\mathcal{R}(s,a)$. Note that for BR, when there is a model of the MDP, the double sampling issue discussed above disappears.

FP and BR find different solutions in general, though when $\lambda = 0$, they are equivalent (this can be seen in the formulas above) and they reduce to Fitted Value Iteration (Szepesvári & Munos, 2005). When $\lambda = 1$, according to the literature (Lagoudakis & Parr, 2003), FP seems to give better results. One can find examples where BR does not compute the right solution while FP does (Sutton et al., 2009). However, Schoknecht (2002) showed that FP can suffer from numerical instability (the matrix $\widetilde{A}$ for FP is not always invertible).

LSλPI iterates over policies. State-of-the-art LS policy evaluation approaches such as LSTD(λ) (Boyan, 2002) and LSPE(λ) (Yu & Bertsekas, 2009) could also be used in a policy iteration context to address control problems. The major difference is that LSλPI is an optimistic algorithm: our algorithm switches to the next policy before the current policy is completely evaluated. In LSTD(λ), the value is evaluated completely and λ controls the depth of the backups made from the sampled trajectories. LSPE(λ) is an approximation of λPI, applied to the evaluation of a fixed policy. It makes multiple λ-steps towards the value of the policy using an LS method until the policy is evaluated completely. LSλPI makes only one λ-step and changes the policy then. Another difference between LSPE(λ) and LSλPI is the fact that they estimate different terms to make the approximate λ step: LSPE(λ) uses one or several trajectories generated with the current policy and relies on the equation $Q_{k+1} = Q_k + \Delta_k$ (see Equation (2)) whereas LSλPI considers the equation $Q_{k+1} = M_k Q_{k+1}$ and can use off-policy samples.

3.4. Performance bound

Bertsekas & Tsitsiklis (1996) provided a performance bound for the policy sequence generated by approximate policy/value iteration algorithms (the cases $\lambda = 0$ and $\lambda = 1$). It turns out that a similar bound applies to a large class of approximate optimistic policy iteration algorithm, of which LSλPI is a specific case. For all these algorithms, if the approximation error is bounded at each iteration, then the asymptotic distance to the optimal policy is bounded:

Theorem 1 (Performance bound for opt. PI) *Let $(\lambda_n)_{n \geq 1}$ be a sequence of positive weights such that $\sum_{n \geq 1} \lambda_n = 1$. Let Q_0 be an arbitrary initialization. We consider an iterative algorithm that generates the sequence $(\pi_k, Q_k)_{k \geq 1}$ with*

$$\pi_{k+1} \leftarrow \text{greedy}(Q_k),$$
$$Q_{k+1} \leftarrow \sum_{n \geq 1} \lambda_n (B_{\pi_{k+1}})^n Q_k + \epsilon_{k+1}.$$

ϵ_{k+1} is the approximation error made when estimating the next value function. Let ϵ be a uniform majoration of that error, i.e. for all k, $\|\epsilon_k\|_\infty \leq \epsilon$. Then

$$\limsup_{k \to \infty} \|Q^* - Q^{\pi_k}\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2}\epsilon.$$

The proof, which is omitted for lack of space, can be found in (Thiery & Scherrer, 2010). Approximate versions of λPI such as LSλPI correspond to the case where $\lambda_n = (1-\lambda)\lambda^{n-1}$ for all n, but the result remains valid for any choice of coefficients λ_n that sum to 1. Thus, the bound also applies to Modified Policy Iteration (Puterman, 1994), which consists in applying m times the Bellman operator with m fixed (i.e. we take $\lambda_m = 1$ and $\lambda_n = 0$ for all $n \neq m$).

4. Experiments

We now present an experimental validation of LSλPI on two problems addressed by Lagoudakis et al. (2002; 2003) for LSPI: a simple chain MDP where the exact optimal value function is known, and the more challenging Tetris problem.

4.1. Chain problem

We consider the exact same chain problem as in Lagoudakis & Parr (2003), that is, a chain of 20 states with two possibles actions: left (L) or right (R). Each action goes in the wanted direction with probability 0.9, and in the wrong direction with probability 0.1. When the agent gets to the left or right end of the chain, a reward of 1 is obtained. In all other cases,

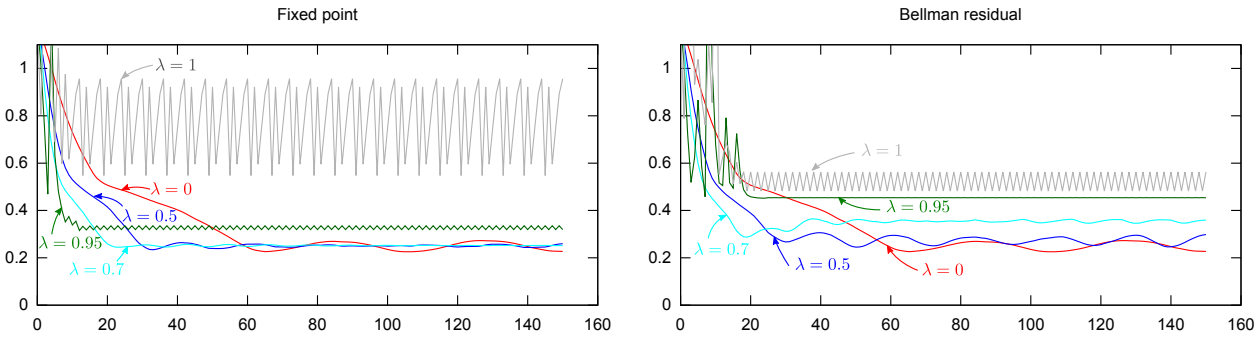

Figure 3. Chain problem. Evolution of $\|\widehat{Q}_k - Q^*\|_\infty$, distance between the current approximate value function and the optimal value function, for several values of λ, $\gamma = 0.95$ and with the Gaussian basis function set. Average curves of 10 runs, each run using a different set of episodes of 200 samples.

the reward is 0. The optimal value can be computed exactly. In our experiments, we compare at each iteration the current value Q_{k+1} to the optimal one, and the (exact) value of the current policy to the optimal value. We do not use the knowledge of the model (transitions and rewards) of the MDP and we represent the state space with the set of polynomial or Gaussian basis functions proposed by Lagoudakis & Parr (2003).

4.1.1. Influence of λ

We naturally observe that the value function convergence is harder when the number of samples is low, or when γ is high (i.e. the horizon is bigger). When this is the case, λ has an influence. Figure 3 represents the distance between the approximate value function and the optimal value function for several values of λ, $\gamma = 0.95$, averaged over 10 executions with different samples. The left graph corresponds to FP and the right graph to BR. As expected, we observe that for $\lambda < 1$, the approximation is better because the sampling variance is reduced, and more iterations are needed to obtain this better approximation. For $\lambda = 1$, after only a few iterations, the value function stops improving and oscillates between values that are relatively far from the optimal ones, compared to lower values of λ. Thus, intermediate values of λ seem to offer the best trade-off between approximation capability and convergence speed. We actually observe, especially for BR, that it might be interesting to use a decreasing value of λ: indeed, in the first iterations, values of λ close to 1 come faster near the optimal value function, and then, smaller values of λ lead to better asymptotic convergence. One can notice that these curves are similar to the ones of Kearns & Singh (2000), which provide a formal analysis of the bias-variance trade-off of TD(λ).

On most of the experiments we ran, FP and BR give similar performance, with a slight advantage for FP. We observe on this typical experiment that for FP, there is a bigger range of λ values that make the value function converge to a good approximation. This confirms the discussion of section 3.3: in practice, FP seems to give better results than BR even though in theory, numeric stability problems can occur.

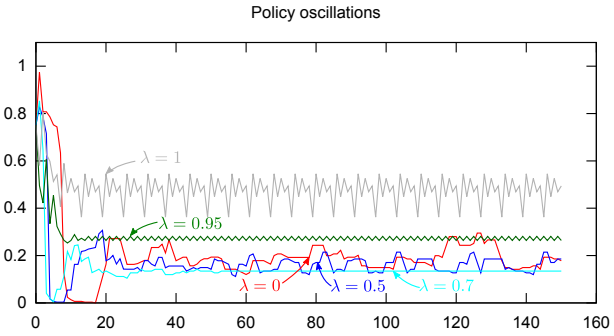

Figure 5. Chain problem. Evolution of $\|Q^{\pi_k} - Q^*\|_\infty$, distance between the value of the current policy and the optimal value, for the FP experiment of Figure 3.

Another observation is that when the value function does not converge, the policy oscillates with a frequency that increases with λ. This happens when there is a cycle in the sequence of policies. It can be seen on Figure 5, which draws $\|Q^{\pi_k} - Q^*\|_\infty$ for the FP experiment of Figure 3. For small values of λ, the policy oscillates slowly because LSλPI makes small steps. When λ increases, the oscillations are faster as the steps are bigger (recall the intuitive view of Figure 2). For big values of λ, the policy does not converge anymore and oscillates with a higher frequency. For $\lambda = 0.7$, we observed that the policy converged: this is all the more interesting that in such a convergent case, the performance guarantee of Theorem 1 can slightly be refined (Thiery & Scherrer, 2010).

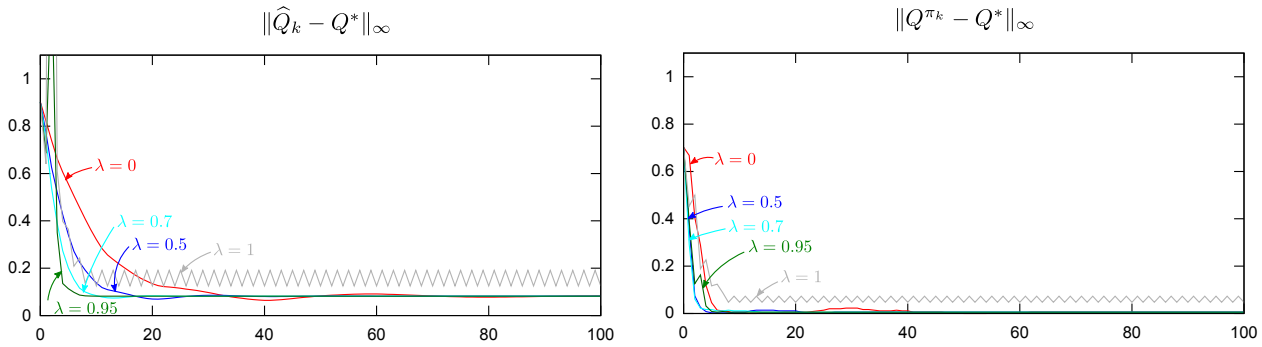

Figure 4. Chain problem. Result of FP applied with $\gamma = 0.9$ for several values of λ and the polynomial basis function set. Left: evolution of $\|\widehat{Q}_k - Q^*\|_\infty$. Right: evolution of $\|Q^{\pi_k} - Q^*\|_\infty$. Average curves of 10 runs, each run using a different set of episodes of 200 samples. We observe that when the policy is the same for different values of λ, the value seems to converge to a limit that does not depend on λ. We verified this property, which does not apply to BR.

4.1.2. Convergence of FP

Figure 4 plots an experiment with FP where convergence is easier than in the previous example as we set $\gamma = 0.9$ (other parameters are unchanged). The right graph shows the quality of the policy: it seems to converge except when $\lambda = 1$. The left graph represents the distance between the current approximate value and the optimal value. After about 40 iterations, the policy becomes the same for all converging values of λ. Then, it seems that the value function converges to the same vector for all these values of λ.

We can verify that for FP, if the policy and the value fonction both converge, then the limit of the value function does not depend on λ (this is in general not the case for BR). If we denote by Π the projection $\Phi(\Phi^{\mathrm{T}} D_\mu \Phi)^{-1} \Phi^{\mathrm{T}} D_\mu$, by using the definition of M_k (Equation (1)) and the facts that $\widehat{Q}_{k+1} = \widehat{Q}_k$ and $\pi_{k+1} = \pi_k$, we have $\widehat{Q}_{k+1} = \Pi M_k \widehat{Q}_{k+1} = \Pi((1-\lambda) B_{\pi_{k+1}} \widehat{Q}_{k+1} + \lambda B_{\pi_{k+1}} \widehat{Q}_{k+1}) = \Pi B_{\pi_{k+1}} \widehat{Q}_{k+1}$.

4.2. Tetris

Tetris is a popular video game that consists in moving and rotating different shapes to fill as many rows as possible in a 10×20 grid. We reproduced the experimental settings of Lagoudakis et al. (2002). Though our initial policy is the same as theirs (personal communication), the scores cannot be compared easily. The initial policy scores about 250 rows per game on our implementation, whereas they report an initial mean score of 600 rows. This may be due to Tetris-specific implementation differences (see the review of Thiery & Scherrer, 2009).

We first ran LSλPI on a set of 10000 samples, as Lagoudakis et al. (2002) did for $\lambda = 1$. We observed that diminishing λ did not improve the performance (it just made the convergence slower). We think that the sample set was too large to make λ useful. We then tried a smaller set of training data (1000 samples instead of 10000) in order to make the convergence more difficult. Figure 6 represents the performance of the learned policies for various values of λ. When $\lambda = 1$, the algorithm diverges: this is because there is a small number of samples. The score switches between 0 and about 600 for FP, and falls to 0 for BR. Better scores are achieved for other values of λ. As in the chain walk experiment, λ has more influence on the performances in BR. After convergence, the best performance is reached by BR with $\lambda = 0.9$ and the corresponding policy scores about 800 rows per game.

Conclusion

We introduced LSλPI, an implementation of λPI (Bertsekas & Ioffe, 1996) within the LS context. Our algorithm generalizes LSPI (Lagoudakis & Parr, 2003) by adding the optimistic evaluation of λPI. LSλPI is to our knowledge the first method that has the following four characteristics: it uses a λ parameter to make a bias-variance trade-off in the value estimation phase, makes an optimistic evaluation of the policy, fits within the LS framework to makes an efficient use of training data, and iterates over policies in an off-policy way while most LS works make on-policy evaluation. It shares the virtues of LSPI: a model of the MDP is not needed (though it can be integrated when available), and as the value function is estimated off-policy, generating new trajectories or samples with each policy is not required.

We provide the analytical claim that optimistic policy iteration algorithms such as LSλPI are sound algorithms and we ran experiments that emphasize the influence of λ in two control problems: a simple chain

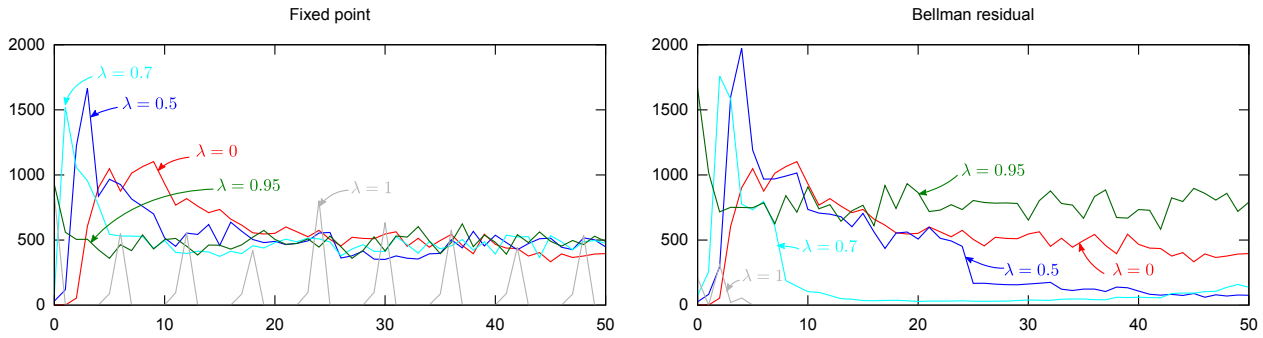

Figure 6. Average score of 100 Tetris games for several values of λ at each iteration of LSλPI. Due to the low number of training samples (1000), the algorithm diverges when $\lambda = 1.0$ for both methods. When convergence is obtained, the best performance is reached by BR with $\lambda = 0.9$.

walk problem and the Tetris game. Our empirical results show that intermediate values of λ can give the best results in practice when the number of samples is low. This may be of particular interest in online, off-policy learning applications.

References

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

Bertsekas, D.P. and Tsitsiklis, J.N. *Neurodynamic Programming*. Athena Scientific, 1996.

Boyan, J. A. Technical update: Least-squares temporal difference learning. *Machine Learning*, 49:233–246, 2002.

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

Kearns, M. and Singh, S. Bias-variance error bounds for temporal difference updates. In *In Proceedings of the 13th Annual Conference on Computational Learning Theory*, pp. 142–147, 2000.

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

Lagoudakis, Michail G., Parr, Ronald, and Littman, Michael L. Least-squares methods in reinforcement learning for control. In *In SETN'02: Proceedings of the Second Hellenic Conference on AI*, pp. 249–260. Springer-Verlag, 2002.

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

Puterman, M. *Markov Decision Processes*. Wiley, New York, 1994.

Schoknecht, Ralf. Optimality of reinforcement learning algorithms with linear function approximation. In *NIPS*, pp. 1555–1562, 2002.

Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C., and Wiewiora, E. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *ICML'09: Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 993–1000, 2009.

Sutton, R.S. and Barto, A.G. *Reinforcement Learning, An introduction*. BradFord Book. The MIT Press, 1998.

Szepesvári, Csaba and Munos, Rémi. Finite time bounds for sampling based fitted value iteration. In *ICML'05: Proceedings of the 22nd international conference on Machine learning*, pp. 880–887. ACM, 2005.

Thiery, Christophe and Scherrer, Bruno. Building Controllers for Tetris. *International Computer Games Association Journal*, 32:3–11, 2009.

Thiery, Christophe and Scherrer, Bruno. Performance bound for Approximate Optimistic Policy Iteration. Technical report, 2010. URL http://hal.inria.fr/inria-00480952.

Yu, H. and Bertsekas, D. P. Convergence Results for Some Temporal Difference Methods Based on Least Squares. *IEEE Trans. Automatic Control*, 54:1515–1531, 2009.