Lambda-Policy Iteration with Randomization for Contractive Models with Infinite Policies: Well-Posedness and Convergence

Yuchao Li YUCHAO@KTH.SE
Karl H. Johansson KALLEJ@KTH.SE
Jonas Mårtensson JONAS1@KTH.SE
Division of Decision and Control Systems, KTH Royal Institute of Technology, Stockholm, Sweden

Editors: A. Bayen, A. Jadbabaie, G. J. Pappas, P. Parrilo, B. Recht, C. Tomlin, M. Zeilinger

Abstract

Abstract dynamic programming models are used to analyze λ-policy iteration with randomization algorithms. Particularly, contractive models with infinite policies are considered and it is shown that well-posedness of the λ-operator plays a central role in the algorithm. The operator is known to be well-posed for problems with finite states, but our analysis shows that it is also well-defined for the contractive models with infinite states studied. Similarly, the algorithm we analyze is known to converge for problems with finite policies, but we identify the conditions required to guarantee convergence with probability one when the policy space is infinite regardless of the number of states. Guided by the analysis, we exemplify a data-driven approximated implementation of the algorithm for estimation of optimal costs of constrained linear and nonlinear control problems. Numerical results indicate potentials of this method in practice.

Keywords: λ-policy iteration, approximate dynamic programming, reinforcement learning

1. Introduction

Temporal-difference (TD) learning is a prominent class of algorithms widely applied in reinforcement learning (RL). Its first formal treatment is given in Sutton (1988) where a family of algorithms, collectively known as TD(λ), is analyzed in the context of absorbing Markov processes. By utilizing the properties of transitional matrices of the process, algorithm convergence guarantees are established. Structural relations between RL and dynamic programming (DP) was noted by Watkins (1989), and foundations for the understanding of RL followed. The monograph by Bertsekas and Tsitsiklis (1996) puts a broad class of RL algorithms in the context of two principle methods of DP, viz., value iteration (VI) and policy iteration (PI), and collects a bundle of research outputs of interests.[1] Among those results, the analysis of TD(λ), originally given in Bertsekas and Ioffe (1996), unveils the underlying DP problem of TD(λ). As is shown, the desired behavior of TD(λ) is inherited from the parameter λ being a discount factor in the classical DP sense and the infinite iterates of TD algorithms can be interpreted as an iteration of a compactly defined operator. In addition, TD(λ) can be embedded into the PI framework, which is then named λ-PI. There has been a tremendous development in algorithms related to λ-PI, such as Thiery and Scherrer (2010); Scherrer et al. (2015). A survey can be found in Bertsekas (2012). Most recently, the connection between TD(λ) and proximal algorithms, which are widely used for solving convex optimization problems, is discussed in Bertsekas (2018b). In light of such relation, λ-PI with randomization (λ-PIR) was

1. A detailed document of the history can be found in (Sutton and Barto, 2018, Chapter 1).

proposed in (Bertsekas, 2018a, Chaper 2). The algorithm resembles the one proposed in Yu and Bertsekas (2015), and offers a scheme to combine the fast computations by proximal algorithms with the convergence behavior by VI. Apart from these algorithmic properties, the abstract approach taken for analyzing λ-PIR is also well worth special attention. Although some operators, in particular the Bellman operator, are often used in algorithmic analysis, they played less of a central role throughout the development, cf. Tsitsiklis and Van Roy (1997); De Farias and Van Roy (2003); Wang et al. (2015); Bellemare et al. (2016); Bian and Jiang (2016); Banjac and Lygeros (2019), in which operator computations are utilized while specific properties of the problem are also taken advantage of. An exception is the analysis of λ-PIR in (Bertsekas, 2018a, Chaper 2), which has solely relied on abstract operator properties. There are many advantages of such an approach, e.g., (a) it can single out the key factor that stands behind the desired behavior of the algorithm; (b) it can shed new lights on the understanding of some algorithms and help to bring together isolated methods; (c) it can help to safeguard the desired behaviors when modifying and generalizing algorithms. One example of this is by Yu et al. (2018), in which the parameter λ is extended to be state-dependent, while fundamental properties are still guaranteed.

In this paper, we use abstract DP models and extend λ-PIR for finite policy problems (Bertsekas, 2018a, Chapter 2) to contractive models with infinite policies. A policy space can be infinite due to infinite states, or infinite control over some finite state space. We make the following main contributions:

(1) We establish the well-posedness of the compact operator that plays a central role in the algorithm (Theorem 3.2). Our result relies solely on the contraction property of the model.

(2) Conditions for convergence of λ-PIR for problems with infinite policies are given (Theorem 4.1). We show that such conditions can be dismissed if the underlying operator exhibits a linear structure (Corollary 4.2).

The rest of the paper is organized as follows: Section 2 gives a brief account of preliminaries of contractive models and existing results on λ-PIR. Section 3 presents results on well-posedness of the λ-PIR algorithm for infinite-state problems. Conditions for convergence of λ-PIR for problems with infinite policies are given in Section 4. Section 5 explains an approximated implementation of λ-PIR and shows its application when embedded in the approximate dynamic programming (ADP) framework. Section 6 concludes the paper.

2. Preliminaries

Here we introduce the concepts and some preliminaries related to contractive models, and the λ-PIR algorithm. The contents here are mostly taken from Bertsekas (2018a).

2.1. Contractive models

Given a state space X, a control space U, and for each $x \in X$ a nonempty control set $U(x) \subset U$, we denote $\mathcal{M} = \{\mu \,|\, \mu(x) \in U(x),\ \forall x \in X\}$ and name it as the set of policies whose elements are denoted by μ. One can see that the set $\mathcal{M}$ can be viewed as the Cartesian product $\prod_{x\in X} U(x)$. We denote by $\mathcal{R}(X)$ the set of functions $J : X \to \mathbb{R}$ and by $\mathcal{E}(X)$ the set of functions $J : X \to \mathbb{R}^*$ where $\mathbb{R}^* = \mathbb{R} \cup \{\infty, -\infty\}$. We study the mappings of the form $H : X \times U \times \mathcal{R}(X) \to \mathbb{R}$. For every $\mu \in \mathcal{M}$, we define $T_\mu : \mathcal{R}(X) \to \mathcal{R}(X)$ as

$$(T_\mu J)(x) = H(x, \mu(x), J),\ \forall x \in X, \tag{1}$$

and the mapping $T : \mathcal{R}(X) \to \mathcal{E}(X)$ as

$$(TJ)(x) = \inf_{\mu \in \mathcal{M}} (T_\mu J)(x), \ \forall x \in X. \tag{2}$$

In view of the definitions of $\mathcal{M}$, T_μ, and T, we have

$$(TJ)(x) = \inf_{u \in U(x)} H(x, u, J) = \inf_{\mu \in \mathcal{M}} H(x, \mu(x), J). \tag{3}$$

Given some positive function $v : X \to \mathbb{R}$, we denote by $\mathcal{B}(X)$ the set of functions J such that $\sup_{x \in X} \frac{|J(x)|}{v(x)} < \infty$. We define a norm $\|\cdot\|$ on $\mathcal{B}(X)$ as

$$\|J\| = \sup_{x \in X} \frac{|J(x)|}{v(x)}.$$

The following lemmas are classical results from functional analysis. The proof of the first can be found in (Bertsekas, 2018a, Appendix B), while the second is explained in (Szepesvári, 2010, Appendix A).

Lemma 2.1 *$\mathcal{B}(X)$ is complete with respect to the metric induced by $\|\cdot\|$.*

Lemma 2.2 *Given a sequence $\{J_k\} \subset \mathcal{B}(X)$ and $J \in \mathcal{B}(X)$, if $J_k \to J$ in the sense that $\lim_{k\to\infty} \|J_k - J\| = 0$, then $\lim_{k\to\infty} J_k(x) = J(x)$, $\forall x \in X$.*

Remark 2.2.1 *The converse of Lemma 2.2 does not necessarily hold, see (Szepesvári, 2010, Appendix A).*

For the mappings H, T_μ and T on $\mathcal{B}(X)$, we introduce the following standard assumptions.

Assumption 2.1 (Well-posedness) *$\forall J \in \mathcal{B}(X)$ and $\forall \mu \in \mathcal{M}$, $T_\mu J \in \mathcal{B}(X)$ and $TJ \in \mathcal{B}(X)$.*

Assumption 2.2 (Uniform contraction) *For some $\alpha \in (0, 1)$, it holds that*

$$\|T_\mu J - T_\mu J'\| \leq \alpha \|J - J'\|, \ \forall J, J' \in \mathcal{B}(X), \ \mu \in \mathcal{M}.$$

One immediate consequence of Assumption 2.2 is that T is also a contraction with the same modulus α, see (Bertsekas, 2018a, Chapter 1). When Assumptions 2.1 and 2.2 hold, the following convergence result holds due to the fixed point theory.

Lemma 2.3 (Bertsekas (2018a), Proposition B.1) *Let Assumptions 2.1, and 2.2 hold. Then:*

(a) There exist unique $J_\mu, J^* \in \mathcal{B}(X)$ such that $TJ^* = J^*$; $T_\mu J_\mu = J_\mu$, $\forall \mu \in \mathcal{M}$.

(b) For arbitrary $J_0 \in \mathcal{B}(X)$, the sequence $\{J_k\}$ where $J_{k+1} = T_\mu J_k$ converges in norm to J_μ.

(c) For arbitrary $J_0 \in \mathcal{B}(X)$, the sequence $\{J_k\}$ where $J_{k+1} = TJ_k$ converges in norm to J^*.

The above results are the backbones of VI. However, they do not guarantee the effectiveness of PI, for which we need some additional assumptions.

Assumption 2.3 (Monotonicity) *$\forall J, J' \in \mathcal{B}(X)$, it holds that $J \leq J'$ implies $H(x,u,J) \leq H(x,u,J')$, $\forall x \in X$, $u \in U(x)$, where $\leq$ indicates point-wise relation.*

Assumption 2.4 (Attainability) *For all $J \in \mathcal{B}(X)$, there exists $\mu \in \mathcal{M}$, such that $T_\mu J = TJ$.*

In fact, only after including Assumption 2.3, in addition to Assumptions 2.1 and 2.2, can J^* be interpreted as optimal in the sense that $J^*(x) = \inf_{\mu \in \mathcal{M}} J_\mu(x)$. Besides, due to the nature of $\mathcal{M}$ being a Cartesian product of feasible control sets $U(x)$, for arbitrary small $\varepsilon > 0$, we can always construct an ε-optimal policy $\mu_\varepsilon \in \mathcal{M}$ in the sense that $J_{\mu_\varepsilon}(x) \leq J^*(x) + \varepsilon$ holds for all x. One such construction in a more general setting can be found in (Bertsekas and Shreve, 1978, Chapter 2) and the details of the above discussion can be found in (Bertsekas, 2018a, Propositions 2.1.1, 2.1.2). Since the infimum in (2) is not always attained, Assumption 2.4 is needed for PI-based methods. In the subsequent sections, we always assume Assumption 2.1 hold, and therefore do not repeat it in all the theoretical statements.

2.2. λ-PIR

The following λ-PIR algorithm is introduced in (Bertsekas, 2018a, Chapter 2) in the abstract setting. Given some $\lambda \in [0,1)$, consider the mappings $T_\mu^{(\lambda)}$ with domain $\mathcal{B}(X)$ and defined point-wise by

$$\big(T_\mu^{(\lambda)} J\big)(x) = (1-\lambda) \sum_{\ell=1}^{\infty} \lambda^{\ell-1} \big(T_\mu^\ell J\big)(x), \tag{4}$$

where T_μ^ℓ denotes the ℓ-fold composition of the operator T_μ, and we refer to the operator $T_\mu^{(\lambda)}$ as λ operator in our discussion. Regarding this operator, we make the following mild assumption, which holds for a broad class of DP problems.

Assumption 2.5 (Commutativeness) *For every $\mu \in \mathcal{M}$, its corresponding λ operator and T_μ commute, viz., for all $J \in \mathcal{B}(X)$, it holds that*

$$T_\mu\big(T_\mu^{(\lambda)} J\big) = T_\mu^{(\lambda)}(T_\mu J).$$

Given $J_k \in \mathcal{B}(X)$ and $p_k \in (0,1)$, then the policy μ^k and cost approximate J_{k+1} is computed as

$$T_{\mu^k} J_k = TJ_k;\ J_{k+1} = \begin{cases} T_{\mu^k} J_k, & \text{with prob. } p_k, \\ T_{\mu^k}^{(\lambda)} J_k, & \text{with prob. } 1-p_k, \end{cases} \tag{5}$$

where the policy improvement step to the left is the same as in classical PI, while the evaluation step on the right is a randomized mix between VI and TD learning.

We list the central statements related to λ-PIR presented in (Bertsekas, 2018a, Chapter 2), which include the assumptions needed and convergence behavior of the algorithm. Except the cases in which $U(x)$ is not singleton for finite number of x, which we refer to as trivial cases, $\mathcal{M}$ being finite implies state space being finite. Therefore, except the trivial cases, with the following finite policy assumption, the λ operator $T_\mu^{(\lambda)}$ is ensured to be well-posed (see (Bertsekas, 2018b, Proposition 2.1)), and the monotonicity of the underlying operator H is not required for the desired behavior.

Assumption 2.6 (Finiteness) *$\mathcal{M}$ is finite.*

Then, the following result holds.

Theorem 2.4 (Bertsekas (2018a), Section 2.5.3) *Let Assumptions 2.2, 2.4, and 2.6 hold. $\forall J_0 \in \mathcal{B}(X)$, the sequence $\{J_k\}$ generated by λ-PIR (5) converges in norm to J^* with probability one.*

3. Well-posedness of $T_\mu^{(\lambda)}$

We first show a general result, and then show that well-posedness of $T_\mu^{(\lambda)}$ is a consequence of it. For the more general operator, we prove first the output of the operator is well-defined within $\mathcal{R}(x)$, viz., point-wise limits do exist in $\mathbb{R}$. Then we show that the output function scaled by the weight function $v(x)$ is bounded, which means that it is an element of $\mathcal{B}(X)$. Then we show the λ operator $T_\mu^{(\lambda)}$ is a special case of the proved results. The proofs of the following results and some additional results can be found in the extended version of this work in Li et al. (2019).

Lemma 3.1 *Let the set of mappings $T_\mu : \mathcal{B}(X) \to \mathcal{B}(X)$, $\mu \in \mathcal{M}$, satisfy Assumption 2.2. Consider the mappings $T_\mu^{(w)}$ with domain $\mathcal{B}(X)$ defined point-wise by*

$$\big(T_\mu^{(w)} J\big)(x) = \sum_{\ell=1}^{\infty} w_\ell(x) \big(T_\mu^\ell J\big)(x), \ x \in X, \ J \in \mathcal{B}(X), \tag{6}$$

where $w_\ell(x)$ are nonnegative scalars such that for all $x \in X$, $\sum_{\ell=1}^\infty w_\ell(x) = 1$. Then the mapping $T_\mu^{(w)}$ is well-defined; namely, for all $x \in X$, $J \in \mathcal{B}(X)$, the sequence $\Big\{ \sum_{\ell=1}^n w_\ell(x) \big(T_\mu^\ell J\big)(x) \Big\}_{n=1}^\infty$ converges with a limit in $\mathbb{R}$, viz., $T_\mu^{(w)} : \mathcal{B}(X) \to \mathcal{R}(X)$.

Theorem 3.2 *Let the set of mappings $T_\mu : \mathcal{B}(X) \to \mathcal{B}(X)$, $\mu \in \mathcal{M}$, satisfy Assumption 2.2. Consider the mappings $T_\mu^{(w)} : \mathcal{B}(X) \to \mathcal{R}(X)$ defined in Eq. (6). Then the range of $T_\mu^{(w)}$ is a subset of $\mathcal{B}(X)$, viz., $T_\mu^{(w)} : \mathcal{B}(X) \to \mathcal{B}(X)$; and $T_\mu^{(w)}$ is a contraction.*

Corollary 3.3 *Let the set of mappings $T_\mu : \mathcal{B}(X) \to \mathcal{B}(X)$, $\mu \in \mathcal{M}$, satisfy Assumption 2.2. The operator $T_\mu^{(\lambda)}$ defined point-wise by Eq. (4) is well-posed in the sense that $T_\mu^{(\lambda)} J \in \mathcal{B}(x)$ for all $J \in \mathcal{B}(x)$, and $T_\mu^{(\lambda)}$ is a contraction with modulus $\alpha_\lambda = \alpha(1-\lambda)/(1-\lambda\alpha)$.*

4. Convergence of λ-PIR

We summarize the convergence results of λ-PIR under the classical contractive model assumptions. The proofs are omitted here and can be found in the extended version of this work in Li et al. (2019).

Theorem 4.1 *Let Assumptions 2.2, 2.3, 2.4, and 2.5 hold. Given $J_0 \in \mathcal{B}(X)$ such that $TJ_0 \leq J_0$, the sequence $\{J_k\}_{k=0}^\infty$ generated by algorithm (5) converges in norm to J^* with probability one.*

The following result, as a special case of Theorem 4.1, shows that if $H(\cdot,\cdot,\cdot)$ has certain 'linear' structure, the initialization condition $TJ_0 \leq J_0$ required in Theorem 4.1 can be dropped and the same convergence result still stands. The proof is obtained by applying Theorem 4.1 and the arguments in Bertsekas and Ioffe (1996) and (Bertsekas and Tsitsiklis, 1996, Chapter 2).

Corollary 4.2 *Let $H(\cdot,\cdot,\cdot)$ have the form*

$$H(x,u,J) = \int_X \big(g(x,u,y) + \alpha J(y)\big) d\mathbb{P}(y|x,u) \tag{7}$$

where $g : X \times U \times X \to \mathbb{R}$, $\alpha \in (0,1)$ and $\mathbb{P}(\cdot|x,u)$ is the probability measure conditioned on (x,u) for certain MDP. Let $v(x) = 1 \ \forall x \in X$, and Assumptions 2.2, 2.3, 2.4, and 2.5 hold. Given arbitrary $J_0 \in \mathcal{B}(X)$, the sequence $\{J_k\}_{k=0}^\infty$ generated by algorithm (5) converges in norm to J^* with probability one.

Remark 4.2.1 *One key insight given in Bertsekas and Ioffe (1996) is that when H has 'linear' form similar to (7), a constant shift of the cost function J does not alter the choice of the optimal policy, which justifies the importance of resembling the 'shape', rather than the 'value', of the optimal costs in the approximation schemes. This is evidently explained in (Bertsekas, 2019, Chapter 3).*

5. Application to ADP

In this section, we exemplify the proposed algorithm for applications of ADP used to solve on-line constrained optimal control problems.

5.1. Constrained optimal control and ADP

Consider optimal control problems with $x_{k+1} = f(x_k, u_k)$, and the abstract operator defined as $H(x, u, J) = g(x, u) + \alpha J(f(x, u))$, where $X \subset \mathbb{R}^n$ and $U \subset \mathbb{R}^m$ are compact sets, and $v(x) = 1\ \forall x \in X$. In addition, we assume the distribution of x_0, denoted as $\mathcal{X}_0$, is given. We denote collectively the problem data as $\mathbf{D}$. Assume $\mathbf{D}$ fulfills contractive model assumption, then there exists $J^* \in \mathcal{R}(X)$ such that $J^* = TJ^*$. However, it is often intractable to compute J^*. Instead, we aim to obtain $\tilde{J}$, a good estimate of J^*. Once $\tilde{J}$ is available, at every instance k, the ADP approach to control the system is to solve online a constrained optimization problem $u_k \in \arg\min_{u \in U(x)} H(x_k, u, \tilde{J})$.

The approximation of λ-PIR implementation comes from two sources. First, the estimate of J^* often uses some form of parametric approximation. In this case, we consider $\tilde{J}(x, \theta)$, where $\theta \in \Theta$ is the parameter to be trained. Second, the $T_\mu^{(\lambda)}$ operation on $\tilde{J}$ can only be performed approximately.

Algorithm 1: Data-driven λ-PIR

Input: problem data $\mathbf{D}$, algorithm data $\mathbf{A}$, initial parameter θ_0, sample size S
Output: θ, the trained parameter
$\theta \leftarrow \theta_0$
for $k \leftarrow 1$ **to** K **do**
  Initialize $\mathbf{x} \in \mathbb{R}^{n \times S}$, $\mathbf{v} \in \mathbb{R}^S$
  for $s \leftarrow 1$ **to** S **do**
    $x_0 \sim \mathcal{X}_0$, $b \sim \text{Ber}(p_k)$
    if $b == 1$ **then**
      $v \leftarrow \inf_{u \in U(x_0)} \left(g(x_0, u) + \alpha \tilde{J}(f(x_0, u), \theta)\right)$
    else
      $L \sim \text{Ge}(\lambda)$, $v = 0$, $x \leftarrow x_0$
      for $\ell \leftarrow 0$ **to** $L - 1$ **do**
        $u \in \arg\min_{u' \in U(x)} \left(g(x, u') + \alpha \tilde{J}(f(x, u'), \theta)\right)$,
        $v \leftarrow v + \alpha^\ell g(x, u)$, $x \leftarrow f(x, u)$
      end
      $v \leftarrow v + \alpha^L \tilde{J}(x, \theta)$
    end
    $\mathbf{x}_s = x_0$, $\mathbf{v}_s = v$
  end
  $\theta \in \arg\min_{\theta' \in \Theta} \sum_{s \in S} |\tilde{J}(\mathbf{x}_s, \theta') - \mathbf{v}_s|^2$
end

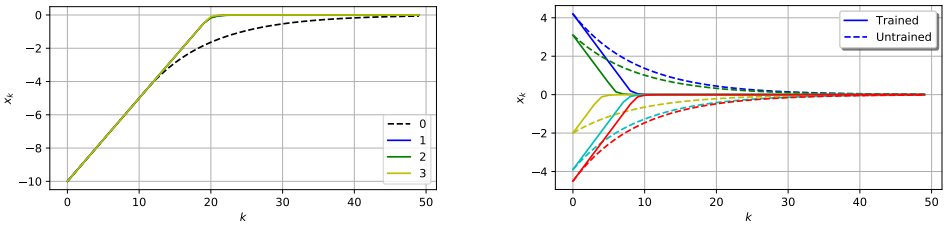
Here we exemplify an data-driven least square evaluation implementation. Our implementation follows closely the projection by Monte Carlo simulation method detailed in (Bertsekas, 2019, Section 5.5). Similar textbook treatment includes (Busoniu et al., 2017, Chapter 5). Denote $\tilde{J}(\cdot, \theta)$, Θ, λ, number of training iterations K, and probability sequence $\{p_k\}_{k=1}^K$, collectively as $\mathbf{A}$. In addition, denote as $\text{Ber}(\cdot)$ the Bernoulli distribution and as $\text{Ge}(\cdot)$ the geometric distribution. The algorithm is summarized in Algorithm 1. At a typical training iteration k, the algorithm starts by sampling from $\text{Ber}(p_k)$ to decide by (5) if the cost estimate of this iteration is obtained via applying T_{μ^k} or $T_{\mu^k}^{(\lambda)}$. For every sample pair (x_0, v), the state x_0 is drawn from $\mathcal{X}_0$, which is part of the problem data. When the T_{μ^k} step is chosen, for all x_0's, their corresponding v's are set to equal to $(T_{\mu^k}\tilde{J})(x_0)$, with μ^k defined by (5). If $T_{\mu^k}^{(\lambda)}$ is selected, an integer ℓ is drawn from $\text{Ge}(\lambda)$ for every

x_0, and its corresponding v is set to $(T^{\ell}_{\mu^k}\tilde{J})(x_0)$. In total, it collects a size of S sample pairs (x_0, v), and updates the parameter θ by solving a lease square regression problem.

5.2. Numerical examples

We apply the proposed algorithm to train the cost function used in ADP for constrained linear and nonlinear systems. Both the training and on-line ADP control problems in the examples are identified as convex and are solved by `cvxpy` (Diamond and Boyd (2016)). Additional implementation details and another example are given in the extended version of this work in Li et al. (2019).

Example 5.1 *Consider a linear scalar control problem with problem data given as* $x_{k+1} = x_k - 0.5u_k$, $H(x, u, J) = x^2 + u^2 + 0.95J(x - 0.5u)$, $\tilde{J}(x, \theta) = ax^2 + b$ *where* $\theta = (a, b)$, $X = [-100, 100]$, $U = [-1, 1]$ *and* $\Theta = \{(a, b) \,|\, a \geq 0\}$. *Similar problems have appeared in Wang et al. (2015); Banjac and Lygeros (2019). The results are shown in Fig. 1, where the performance is greatly improved from initial guess of* θ *after 2 iterations.*

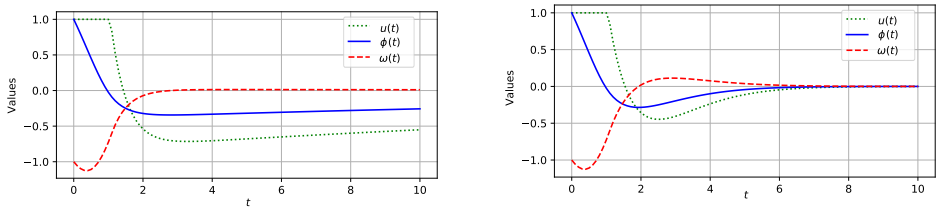


(*a*) System behavior under ADP controls with different cost function estimate.

(*b*) System behavior with trained and untrained cost function from different initial states.

Figure 1: Closed-loop system behavior under ADP control.

Example 5.2 *Consider a torsional pendulum system with states* $\phi \in (-\pi/2, \pi/2)$, $\omega \in [-2, 2]$, *control* $\tau \in [-1, 1]$, *and dynamics* $\dot{\phi} = \omega$, $\dot{\omega} = M^{-1}(-mgl \sin\phi - \gamma\omega + \tau)$, *where* $m = 1/3$ kg, $l = 3/2$ m, $M = 4/3ml^2$, $\gamma = 0.2$ *and* $g = 9.8$ m/s^2. *The discrete dynamics, denoted as* $f(\cdot)$ *and used for ADP control, is obtained by forward Euler method with sampling time* 0.1 s *where the state is* $x = [\phi, \omega]^{\mathrm{T}}$ *with* T *denoting transpose operation, and the control is* $u = \tau$. *Then the problem data is given as* $x_{k+1} = f(x_k, u_k)$, $H(x, u, J) = x^{\mathrm{T}}Qx + u^{\mathrm{T}}Ru + 0.95J(f(x, u))$, $\tilde{J}(x, \theta) = x^{\mathrm{T}}Px + b$, *where* Q *is identity matrix,* $R = 0.1$, $\theta = (P, b)$, $X \subset \mathbb{R}^2$, $U = [-1, 1]$ *and* $\Theta = \{(P, b) \,|\, P \succeq 0\}$. *Similar example has appeared in Si and Wang (2001); Liu and Wei (2013). We set* $S = 100$, $K = 5$, $p_k = 0.5$ *for all* k, *and* $\lambda = 0.1$ *so that the lookahead steps in average is* $1/\lambda = 10$. *The closed loop system behavior with initial* θ *and* θ *after* 5 *iterations are shown in Fig. 2 where the continuous system dynamics is solved by* `ode45`. *The control performance is greatly improved.*

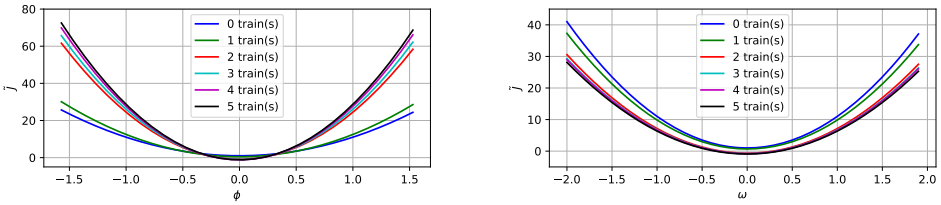


(*a*) With initial guess of θ.

(*b*) With θ after 5 training iterations.

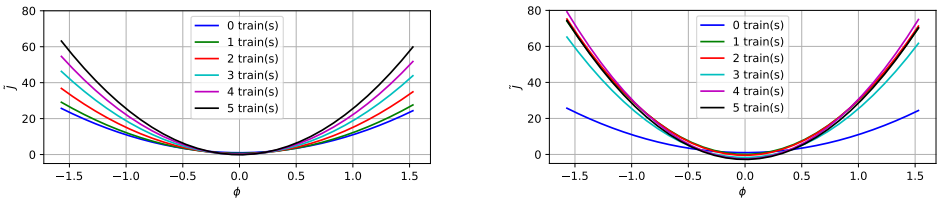
Figure 2: Closed-loop system behavior under ADP control with untrained and trained θ.

Here we show the cost function plots along the axes where $\omega = 0$ and $\phi = 0$, and the cost estimates converged. Besides, we compared the performance of λ-PIR with approximated implementation of VI where in (5) the evaluation step is always applying $T_{\mu^k}\tilde{J}$, and optimistic policy iteration (OPI) where the evaluation is performed as $T^{\ell}_{\mu^k}\tilde{J}$ with ℓ fixed at $1/\lambda = 10$. OPI is analyzed by Scherrer et al. (2015) for finite state case and is known to be closed related to λ-PI. In λ-PIR, the $T^{(\lambda)}_{\mu^k}\tilde{J}$ step occurred in the 2nd iteration, and one can observe a 'boost' towards the optimal in Fig. 3(a), while VI in Fig. 4(a) is yet to converge in the 5th iteration. On the other hand, although OPI in Fig. 4(b) behaves quite similarly to λ-PIR, it does require more sampling efforts compared to λ-PIR. The results here imply that λ-PIR combined the benefits of those two methods.



(*a*) Cost function along the axis $\omega = 0$ after different training iterations. (*b*) Cost function along the axis $\phi = 0$ after different training iterations.

Figure 3: Cost function estimates along the axes $\phi = 0$ and $\omega = 0$ after different training iterations.



(*a*) Cost function along the axis $\omega = 0$ for VI. (*b*) Cost function along the axis $\omega = 0$ for OPI.

Figure 4: Cost function estimates of VI and OPI along the axis $\omega = 0$.

6. Conclusions

We presented results related to λ-PIR aided by abstract DP models. The λ-PIR is originally devised for finite policy problems and our results showed that the algorithm is also well-defined for contractive models with infinite states and the algorithmic convergence can be ensured for problems with infinite policies by adding an additional condition, which can be dismissed if the problem exhibits a linear structure. We exemplified a data-driven approximated implementation of the algorithm to estimate cost functions for constrained optimal control problems and the obtained estimates resulted in good closed-loop behavior when embedded in ADP for online control in numerical examples.

Acknowledgments

This work was supported by the Swedish Foundation for Strategic Research, the Swedish Research Council, and the Knut and Alice Wallenberg Foundation. The authors are grateful to Prof. Dimitri P. Bertsekas for discussions pointing to abstract DP models and to the connection between λ-PI and proximal algorithms, and for his suggestions to improve this work. The helpful comments from the reviewers are also acknowledged.

References

Goran Banjac and John Lygeros. A data-driven policy iteration scheme based on linear programming. In *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019.

Marc G. Bellemare, Georg Ostrovski, Arthur Guez, Philip S. Thomas, and Rémi Munos. Increasing the action gap: New operators for reinforcement learning. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Dimitiri P. Bertsekas and Steven Shreve. *Stochastic optimal control: the discrete-time case*. Academic Press, 1978.

Dimitri P. Bertsekas. Lambda-policy iteration: A review and a new implementation. *Reinforcement learning and approximate dynamic programming for feedback control*, pages 381–406, 2012.

Dimitri P. Bertsekas. *Abstract dynamic programming*. Athena Scientific, 2nd edition, 2018a.

Dimitri P. Bertsekas. Proximal algorithms and temporal difference methods for solving fixed point problems. *Computational Optimization and Applications*, 70(3):709–736, 2018b.

Dimitri P. Bertsekas. *Reinforcement learning and optimal control*. Athena Scientific, 2019.

Dimitri P. Bertsekas and Sergey Ioffe. Temporal differences-based policy iteration and applications in neuro-dynamic programming. lab. for info. and decision systems report lids-p-2349, 1996.

Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-dynamic programming*, volume 5. Athena Scientific Belmont, MA, 1996.

Tao Bian and Zhong-Ping Jiang. Value iteration, adaptive dynamic programming, and optimal control of nonlinear systems. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 3375–3380. IEEE, 2016.

Lucian Busoniu, Robert Babuska, Bart De Schutter, and Damien Ernst. *Reinforcement learning and dynamic programming using function approximators*. CRC press, 2017.

Daniela Pucci De Farias and Benjamin Van Roy. The linear programming approach to approximate dynamic programming. *Operations research*, 51(6):850–865, 2003.

Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

Yuchao Li, Karl H. Johansson, and Jonas Mårtensson. Lambda-policy iteration with randomization for contractive models with infinite policies: Well posedness and convergence (extended version). *arXiv preprint arXiv:1912.08504*, 2019.

Derong Liu and Qinglai Wei. Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems. *IEEE Transactions on Neural Networks and Learning Systems*, 25(3): 621–634, 2013.

Bruno Scherrer, Mohammad Ghavamzadeh, Victor Gabillon, Boris Lesner, and Matthieu Geist. Approximate modified policy iteration and its application to the game of Tetris. *Journal of Machine Learning Research*, 16:1629–1676, 2015.

Jennie Si and Yu-Tsung Wang. Online learning control by association and reinforcement. *IEEE Transactions on Neural networks*, 12(2):264–276, 2001.

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Csaba Szepesvári. *Algorithms for reinforcement learning*. Synthesis lectures on artificial intelligence and machine learning. Morgan & Claypool Publishers, 2010.

Christophe Thiery and Bruno Scherrer. Least-squares policy iteration: Bias-variance trade-off in control problems. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1071–1078, Haifa, Israel, June 2010. Omnipress.

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

Yang Wang, Brendan O'Donoghue, and Stephen Boyd. Approximate dynamic programming via iterated bellman inequalities. *International Journal of Robust and Nonlinear Control*, 25(10): 1472–1496, 2015.

Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

Huizhen Yu and Dimitri P. Bertsekas. A mixed value and policy iteration method for stochastic control with universally measurable policies. *Mathematics of Operations Research*, 40(4):926–968, 2015.

Huizhen Yu, A. Rupam Mahmood, and Richard S. Sutton. On generalized Bellman equations and temporal-difference learning. *Journal of Machine Learning Research*, 19(1):1864–1912, 2018.