Approximate Modified Policy Iteration

Bruno Scherrer, Victor Gabillon, Mohammad Ghavamzadeh, Matthieu Geist

Approximate Modified Policy Iteration

Abstract

Modified policy iteration (MPI) is a dynamic programming (DP) algorithm that contains the two celebrated policy and value iteration methods. Despite its generality, MPI has not been thoroughly studied, especially its approximation form which is used when the state and/or action spaces are large or infinite. In this paper, we propose three approximate MPI (AMPI) algorithms that are extensions of the well-known approximate DP algorithms: fitted-value iteration, fitted-Q iteration, and classification-based policy iteration. We provide an error propagation analysis for AMPI that unifies those for approximate policy and value iteration. We also provide a finite-sample analysis for the classification-based implementation of AMPI (CBMPI), which is more general (and somehow contains) than the analysis of the other presented AMPI algorithms. An interesting observation is that the MPI's parameter allows us to control the balance of errors (in value function approximation and in estimating the greedy policy) in the final performance of the CBMPI algorithm.

1. Introduction

Modified Policy Iteration (MPI) (Puterman & Shin, 1978) is an iterative algorithm to compute the optimal policy and value function of a Markov Decision Process (MDP). Starting from an arbitrary value function v_0, it generates a sequence of value-policy pairs

$$\pi_{k+1} = \mathcal{G}\, v_k \qquad \text{(greedy step)} \qquad (1)$$

$$v_{k+1} = (T_{\pi_{k+1}})^m v_k \qquad \text{(evaluation step)} \qquad (2)$$

where $\mathcal{G}\, v_k$ is the *greedy* policy w.r.t. v_k, T_{π_k} is the Bellman operator associated to the policy π_k, and $m \geq 1$ is a parameter. MPI generalizes the well-known dynamic programming algorithms Value Iteration (VI) and Policy Iteration (PI) for values $m = 1$ and $m = \infty$, respectively. MPI has less computation per iteration than PI (in a way similar to VI), while enjoys the faster convergence (less iteration) of the PI algorithm. Moreover, it has the convergence guarantees of VI and PI, when it is run in its exact form (in problems with small state and action spaces) (Puterman & Shin, 1978).

In problems with large state and/or action spaces, approximate versions of VI and PI have been the focus of a rich literature (see e.g., Bertsekas & Tsitsiklis 1996; Szepesvári 2010). Approximate VI (AVI) generates the next value function as the approximation of the application of the Bellman optimality operator to the current value (Ernst et al., 2005; Antos et al., 2007a; Munos & Szepesvári, 2008). On the other hand, approximate PI (API) first finds an approximation of the value of the current policy and then generates the next policy as greedy w.r.t. this approximation (Lagoudakis & Parr, 2003a). However, apart from the work on the λ-policy iteration algorithm (Bertsekas & Ioffe, 1996; Thiery & Scherrer, 2010), which is somehow a more complicated variation of MPI that involves computing a fixed-point at each iteration, and thus, suffers from some of the drawbacks of PI, to the best of our knowledge MPI has never been studied in approximate form. The goal of this paper is to fill this gap and to show that similar to its exact form, approximate MPI (AMPI) may present an interesting and more general alternative to AVI and API algorithms.

In this paper, we first propose several implementations of AMPI (Sec. 3) that generalize the state-of-the-art AVI (Ernst et al., 2005; Antos et al., 2007a) and classification-based API algorithms (Lagoudakis & Parr, 2003b; Fern et al., 2006; Lazaric et al., 2010; Gabillon et al., 2011). We then provide an error propagation analysis of AMPI (Sec. 4), which shows how the L_p-norm of its performance loss can be controlled by the error at each iteration of the algorithm. We show that the error propagation analysis of AMPI is more involved than that of AVI and API. This is due to the fact that neither the contraction nor monotonicity arguments, that the error propagation analysis of these two algorithms rely on, hold for AMPI. The analysis of this section unifies those for AVI and API and can be used for any of the AMPI implementations pre-

Appearing in *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, Scotland, UK, 2012.

sented in Sec. 3. Finally in Sec. 5, we provide a finite-sample analysis of a classification-based implementation of AMPI presented in Sec. 3, called CBMPI. The reason for selecting CBMPI is the fact that its analysis is more general than the other AMPI algorithms presented in Sec. 3, mainly due to its actor-critic flavour. The results indicate that the parameter m allows us to balance the importance of the errors of approximating the value function (critic) and estimating the greedy policy (actor) in the final performance bound of CBMPI. We also report some preliminary results of applying CBMPI to standard benchmark problems and comparing it with some existing algorithms in Appendix H in the supplementary material.

2. Notations and Background

We consider a discounted MDP $\langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is a finite action space, $P(ds'|s,a)$, for all (s,a), is a probability kernel on $\mathcal{S}$, the reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is bounded by $R_{\max}$, and $\gamma \in (0,1)$ is a discount factor. For any function $f : \mathcal{S} \to \mathbb{R}$ and any distribution μ on $\mathcal{S}$, we define the μ-weighted L_p norm as $\|f\|_{p,\mu} \triangleq \left(\int |f(x)|^p \mu(dx)\right)^{1/p}$. A deterministic policy is defined as a mapping $\pi : \mathcal{S} \to \mathcal{A}$. For a policy π, we may write $r_\pi(s) = r\big(s, \pi(s)\big)$ and $P_\pi(ds'|s) = P\big(ds'|s, \pi(s)\big)$. The value of policy π in a state s is defined as the expected discounted sum of rewards received starting from state s and following the policy π, i.e., $v_\pi(s) = \mathbb{E}\big[\sum_{t=0}^{\infty} \gamma^t r_\pi(s_t) | \, s_0 = s, s_{t+1} \sim P_\pi(\cdot|s_t)\big]$. Similarly, the action-value function of a policy π at a state-action pair (s,a), $Q_\pi(s,a)$, is the expected discounted sum of rewards received starting from state s, taking action a, and then following the policy. Since the rewards are bounded by $R_{\max}$, the values and action-values should be bounded by $V_{\max} = Q_{\max} = R_{\max}/(1-\gamma)$. The Bellman operator T_π of policy π takes a function f on $\mathcal{S}$ as input and returns the function $T_\pi f$ defined as $\forall s,\ [T_\pi f](s) = \mathbb{E}\big[r_\pi(s) + \gamma f(s') \mid s' \sim P_\pi(.|s)\big]$, or in compact form, $T_\pi f = r_\pi + \gamma P_\pi f$. It is known that v_π is the unique fixed-point of T_π. Given a function f on $\mathcal{S}$, we say that a policy π is greedy w.r.t. f, and write it as $\pi = \mathcal{G}\, f$, if $\forall s,\ (T_\pi f)(s) = \max_a (T_a f)(s)$, or equivalently $T_\pi f = \max_{\pi'} (T_{\pi'} f)$. We denote by v_* the optimal value function. It is also known that v_* is the unique fixed-point of the Bellman optimality operator $T : v \to \max_\pi T_\pi v = T_{\mathcal{G}(v)} v$, and that a policy π_* that is greedy w.r.t. v_* is optimal and its value satisfies $v_{\pi_*} = v_*$.

3. Approximate MPI Algorithms

In this section, we describe three approximate MPI (AMPI) algorithms. These algorithms rely on a function space $\mathcal{F}$ to approximate value functions, and in the third algorithm, also on a policy space Π to represent greedy policies. In what follows, we describe the iteration k of these iterative algorithms.

3.1. AMPI-V

For the first and conceptually the simplest AMPI algorithm presented in the paper, we assume that the values v_k are represented in a function space $\mathcal{F} \subseteq \mathbb{R}^{|\mathcal{S}|}$. In any state s, the action $\pi_{k+1}(s)$ that is greedy w.r.t. v_k can be estimated as follows:

$$\pi_{k+1}(s) = \arg\max_{a \in \mathcal{A}} \frac{1}{M} \Big(\sum_{j=1}^{M} r_a^{(j)} + \gamma v_k(s_a^{(j)}) \Big), \qquad (3)$$

where $\forall a \in \mathcal{A}$ and $1 \le j \le M$, $r_a^{(j)}$ and $s_a^{(j)}$ are samples of rewards and next states when action a is taken in state s. Thus, approximating the greedy action in a state s requires $M|\mathcal{A}|$ samples. The algorithm works as follows. It first samples N states from a distribution μ, i.e., $\{s^{(i)}\}_{i=1}^N \sim \mu$. From each sampled state $s^{(i)}$, it generates a rollout of size m, i.e., $\big(s^{(i)}, a_0^{(i)}, r_0^{(i)}, s_1^{(i)}, \ldots, a_{m-1}^{(i)}, r_{m-1}^{(i)}, s_m^{(i)}\big)$, where $a_t^{(i)}$ is the action suggested by π_{k+1} in state $s_t^{(i)}$, computed using Eq. 3, and $r_t^{(i)}$ and $s_{t+1}^{(i)}$ are the reward and next state induced by this choice of action. For each $s^{(i)}$, we then compute a rollout estimate

$$\widehat{v}_{k+1}(s^{(i)}) = \sum_{t=0}^{m-1} \gamma^t r_t^{(i)} + \gamma^m v_k(s_m^{(i)}),$$

which is an unbiased estimate of $\big[\big(T_{\pi_{k+1}}\big)^m v_k\big](s^{(i)})$. Finally, v_{k+1} is computed as the best fit in $\mathcal{F}$ to these estimates, i.e.,

$$v_{k+1} = \mathrm{Fit}_{\mathcal{F}} \left(\Big\{ \big(s^{(i)}, \widehat{v}_{k+1}(s^{(i)})\big) \Big\}_{i=1}^{N} \right).$$

Each iteration of AMPI-V requires N rollouts of size m, and in each rollout any of the m actions needs $M|\mathcal{A}|$ samples to compute Eq. 3. This gives a total of $Nm(M|\mathcal{A}|+1)$ transition samples. Note that the fitted value iteration algorithm (Antos et al., 2007b) is a special case of AMPI-V when $m = 1$.

3.2. AMPI-Q

In AMPI-Q, we replace value function $v : \mathcal{S} \to \mathbb{R}$ with action-value function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The Bellman

operator for a policy π at a state-action pair (s,a) can then be written as

$$[T_\pi Q](s,a) = \mathbb{E}\big[r_\pi(s,a) + \gamma Q(s', \pi(s'))|s' \sim P(\cdot|s,a)\big],$$

and the greedy operator is defined as

$$\pi = \mathcal{G}\, Q \quad \Leftrightarrow \quad \forall s \quad \pi(s) = \arg\max_{a \in \mathcal{A}} Q(s,a).$$

In AMPI-Q, action-value functions Q_k are represented in a function space $\mathcal{F} \subseteq \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$, and the greedy action w.r.t. Q_k at a state s, i.e., $\pi_{k+1}(s)$, is computed as

$$\pi_{k+1}(s) \in \arg\max_{a \in \mathcal{A}} Q_k(s,a). \tag{4}$$

The *evaluation step* is similar to that of AMPI-V, with the difference that now we work with state-action pairs. We sample N state-action pairs from a distribution μ on $\mathcal{S}\times\mathcal{A}$ and build a rollout set $\mathcal{D}_k = \{(s^{(i)}, a^{(i)})\}_{i=1}^N$, $(s^{(i)}, a^{(i)}) \sim \mu$. For each $(s^{(i)}, a^{(i)}) \in \mathcal{D}_k$, we generate a rollout of size m, i.e., $\big(s^{(i)}, a^{(i)}, r_0^{(i)}, s_1^{(i)}, a_1^{(i)}, \cdots, s_m^{(i)}, a_m^{(i)}\big)$, where the first action is $a^{(i)}$, $a_t^{(i)}$ for $t \geq 1$ is the action suggested by π_{k+1} in state $s_t^{(i)}$ computed using Eq. 4, and $r_t^{(i)}$ and $s_{t+1}^{(i)}$ are the reward and next state induced by this choice of action. For each $(s^{(i)}, a^{(i)}) \in \mathcal{D}_k$, we then compute the rollout estimate

$$\widehat{Q}_{k+1}(s^{(i)}, a^{(i)}) = \sum_{t=0}^{m-1} \gamma^t r_t^{(i)} + \gamma^m Q_k(s_m^{(i)}, a_m^{(i)}),$$

which is an unbiased estimate of $\big[(T_{\pi_{k+1}})^m Q_k\big](s^{(i)}, a^{(i)})$. Finally, Q_{k+1} is the best fit to these estimates in $\mathcal{F}$, i.e.,

$$Q_{k+1} = \mathrm{Fit}_{\mathcal{F}}\left(\left\{\big((s^{(i)}, a^{(i)}), \widehat{Q}_{k+1}(s^{(i)}, a^{(i)})\big)\right\}_{i=1}^N\right).$$

Each iteration of AMPI-Q requires Nm samples, which is less than that for AMPI-V. However, it uses a hypothesis space on state-action pairs instead of states. Note that the fitted-Q iteration algorithm (Ernst et al., 2005; Antos et al., 2007a) is a special case of AMPI-Q when $m = 1$.

3.3. Classification-Based MPI

The third AMPI algorithm presented in this paper, called classification-based MPI (CBMPI), uses an explicit representation for the policies π_k, in addition to the one used for value functions v_k. The idea is similar to the classification-based PI algorithms (Lagoudakis & Parr, 2003b; Fern et al., 2006; Lazaric et al., 2010; Gabillon et al., 2011) in which we search for the greedy policy in a policy space Π (defined by a classifier)

Input: Value function space $\mathcal{F}$, policy space Π, state distribution μ
Initialize: Let $\pi_1 \in \Pi$ be an arbitrary policy and $v_0 \in \mathcal{F}$ an arbitrary value function
for $k = 1, 2, \ldots$ **do**
- ***Perform rollouts:***

Construct the rollout set $\mathcal{D}_k = \{s^{(i)}\}_{i=1}^n$, $s^{(i)} \overset{\text{iid}}{\sim} \mu$
for all states $s^{(i)} \in \mathcal{D}_k$ **do**
Perform a rollout and return $\widehat{v}_k(s^{(i)})$
end for
Construct the rollout set $\mathcal{D}'_k = \{s^{(i)}\}_{i=1}^N$, $s^{(i)} \overset{\text{iid}}{\sim} \mu$
for all states $s^{(i)} \in \mathcal{D}'_k$ and actions $a \in \mathcal{A}$ **do**
for $j = 1$ to M **do**
Perform a rollout and return $R_k^j(s^{(i)}, a)$
end for
$\widehat{Q}_k(s^{(i)}, a) = \frac{1}{M}\sum_{j=1}^M R_k^j(s^{(i)}, a)$
end for
- ***Approximate value function:***

$v_k = \underset{v\in\mathcal{F}}{\mathrm{argmin}}\, \widehat{\mathcal{L}}_k^{\mathcal{F}}(\widehat{\mu}; v)$ **(regression)**
- ***Approximate greedy policy:***

$\pi_{k+1} = \underset{\pi\in\Pi}{\mathrm{argmin}}\, \widehat{\mathcal{L}}_k^{\Pi}(\widehat{\mu}; \pi)$ **(classification)**
end for

Figure 1. The pseudo-code of the CBMPI algorithm.

instead of computing it from the estimated value or action-value function (like in AMPI-V and AMPI-Q).

In order to describe CBMPI, we first rewrite the MPI formulation (Eqs. 1 and 2) as

$$v_k = (T_{\pi_k})^m v_{k-1} \qquad \text{(evaluation step)} \tag{5}$$

$$\pi_{k+1} = \mathcal{G}\left[(T_{\pi_k})^m v_{k-1}\right] \qquad \text{(greedy step)} \tag{6}$$

Note that in the new formulation both v_k and π_{k+1} are functions of $(T_{\pi_k})^m v_{k-1}$. CBMPI is an approximate version of this new formulation. As described in Fig. 1, CBMPI begins with arbitrary initial policy $\pi_1 \in \Pi$ and value function $v_0 \in \mathcal{F}$.[1] At each iteration k, a new value function v_k is built as the best approximation of the m-step Bellman operator $(T_{\pi_k})^m v_{k-1}$ in $\mathcal{F}$ (*evaluation step*). This is done by solving a regression problem whose target function is $(T_{\pi_k})^m v_{k-1}$. To set up the regression problem, we build a rollout set $\mathcal{D}_k$ by sampling n states i.i.d. from a distribution μ.[2] For each state $s^{(i)} \in \mathcal{D}_k$, we generate a rollout $\big(s^{(i)}, a_0^{(i)}, r_0^{(i)}, s_1^{(i)}, \ldots, a_{m-1}^{(i)}, r_{m-1}^{(i)}, s_m^{(i)}\big)$ of size m, where $a_t^{(i)} = \pi_k(s_t^{(i)})$, and $r_t^{(i)}$ and $s_{t+1}^{(i)}$ are the reward and next state induced by this choice of action. From

[1] Note that the function space $\mathcal{F}$ and policy space Π are automatically defined by the choice of the regressor and classifier, respectively.

[2] Here we used the same sampling distribution μ for both regressor and classifier, but in general different distributions may be used for these two components.

this rollout, we compute an unbiased estimate $\widehat{v}_k(s^{(i)})$ of $\big[(T_{\pi_k})^m v_{k-1}\big](s^{(i)})$ as

$$\widehat{v}_k(s^{(i)}) = \sum_{t=0}^{m-1} \gamma^t r_t^{(i)} + \gamma^m v_{k-1}(s_m^{(i)}), \tag{7}$$

and use it to build a training set $\big\{\big(s^{(i)}, \widehat{v}_k(s^{(i)})\big)\big\}_{i=1}^n$. This training set is then used by the regressor to compute v_k as an estimate of $(T_{\pi_k})^m v_{k-1}$.

The *greedy step* at iteration k computes the policy π_{k+1} as the best approximation of $\mathcal{G}\big[(T_{\pi_k})^m v_{k-1}\big]$ by solving a cost-sensitive classification problem. From the definition of a greedy policy, if $\pi = \mathcal{G}\big[(T_{\pi_k})^m v_{k-1}\big]$, for each $s \in \mathcal{S}$, we have

$$\big[T_\pi (T_{\pi_k})^m v_{k-1}\big](s) = \max_{a \in \mathcal{A}} \big[T_a (T_{\pi_k})^m v_{k-1}\big](s). \tag{8}$$

By defining $Q_k(s,a) = \big[T_a (T_{\pi_k})^m v_{k-1}\big](s)$, we may rewrite Eq. 8 as

$$Q_k\big(s, \pi(s)\big) = \max_{a \in \mathcal{A}} Q_k(s,a). \tag{9}$$

The cost-sensitive error function used by CBMPI is of the form

$$\mathcal{L}^{\Pi}_{\pi_k, v_{k-1}}(\mu;\pi) = \int_{\mathcal{X}} \Big[\max_{a \in \mathcal{A}} Q_k(s,a) - Q_k\big(s,\pi(s)\big)\Big] \mu(ds).$$

To simplify the notation we use $\mathcal{L}^{\Pi}_k$ instead of $\mathcal{L}^{\Pi}_{\pi_k, v_{k-1}}$. To set up this cost-sensitive classification problem, we build a rollout set $\mathcal{D}'_k$ by sampling N states i.i.d. from a distribution μ. For each state $s^{(i)} \in \mathcal{D}'_k$ and each action $a \in \mathcal{A}$, we build M independent rollouts of size $m+1$, i.e.,[3]

$$\big(s^{(i)}, a, r_0^{(i,j)}, s_1^{(i,j)}, a_1^{(i,j)}, \ldots, a_m^{(i,j)}, r_m^{(i,j)}, s_{m+1}^{(i,j)}\big)_{j=1}^M,$$

where for $t \geq 1$, $a_t^{(i,j)} = \pi_k(s_t^{(i,j)})$, and $r_t^{(i,j)}$ and $s_{t+1}^{(i,j)}$ are the reward and next state induced by this choice of action. From these rollouts, we compute an unbiased estimate of $Q_k(s^{(i)}, a)$ as

$$\begin{aligned}\widehat{Q}_k(s^{(i)}, a) &= \frac{1}{M} \sum_{j=1}^M R_k^j(s^{(i)}, a) \\ &= \frac{1}{M} \sum_{j=1}^M \left(\sum_{t=0}^m \gamma^t r_t^{(i,j)} + \gamma^{m+1} v_{k-1}(s_{m+1}^{(i,j)}) \right).\end{aligned}$$

Given the outcome of the rollouts, CBMPI uses a cost-sensitive classifier to return a policy π_{k+1} that minimizes the following *empirical error*

$$\widehat{\mathcal{L}}^{\Pi}_k(\widehat{\mu};\pi) = \frac{1}{N} \sum_{i=1}^N \Big[\max_{a \in \mathcal{A}} \widehat{Q}_k(s^{(i)}, a) - \widehat{Q}_k\big(s^{(i)}, \pi(s^{(i)})\big)\Big],$$

with the goal of minimizing the true error $\mathcal{L}^{\Pi}_k(\mu;\pi)$.

Each iteration of CBMPI requires $Nm + M|\mathcal{A}|N(m+1)$ (or $M|\mathcal{A}|N(m+1)$ in case we reuse the rollouts, see Footnote 3) transition samples. Note that when m tends to ∞, we recover the DPI algorithm proposed and analyzed by Lazaric et al. (2010).

4. Error propagation

In this section, we derive a general formulation for propagation of error through the iterations of an AMPI algorithm. The line of analysis for error propagation is different in VI and PI algorithms. VI analysis is based on the fact that this algorithm computes the fixed point of the Bellman optimality operator, and this operator is a γ-contraction in max-norm (Bertsekas & Tsitsiklis, 1996; Munos, 2007). On the other hand, it can be shown that the operator by which PI updates the value from one iteration to the next is not a contraction in max-norm in general. Unfortunately, we can show that the same property holds for MPI when it does not reduce to VI (i.e., $m > 1$).

Proposition 1. *If $m > 1$, there exists no norm for which the operator that MPI uses to update the values from one iteration to the next is a contraction.*

Proof. Consider a deterministic MDP with two states $\{s_1, s_2\}$, two actions $\{change, stay\}$, rewards $r(s_1) = 0, r(s_2) = 1$, and transitions $P_{ch}(s_2|s_1) = P_{ch}(s_1|s_2) = P_{st}(s_1|s_1) = P_{st}(s_2|s_2) = 1$. Consider the following two value functions $v = (\epsilon, 0)$ and $v' = (0, \epsilon)$ with $\epsilon > 0$. Their corresponding greedy policies are $\pi = (st, ch)$ and $\pi' = (ch, st)$, and the next iterates of v and v' can be computed as $(T_\pi)^m v = \begin{pmatrix} \gamma^m \epsilon \\ 1 + \gamma^m \epsilon \end{pmatrix}$ and $(T_{\pi'})^m v' = \begin{pmatrix} \frac{\gamma - \gamma^m}{1-\gamma} + \gamma^m \epsilon \\ \frac{1-\gamma^m}{1-\gamma} + \gamma^m \epsilon \end{pmatrix}$. Thus, $(T_{\pi'})^m v' - (T_\pi)^m v = \begin{pmatrix} \frac{\gamma - \gamma^m}{1-\gamma} \\ \frac{\gamma - \gamma^m}{1-\gamma} \end{pmatrix}$ while $v' - v = \begin{pmatrix} -\epsilon \\ \epsilon \end{pmatrix}$. Since ϵ can be arbitrarily small, the norm of $(T_{\pi'})^m v' - (T_\pi)^m v$ can be arbitrarily larger than the norm of $v - v'$ as soon as $m > 1$. □

We also know that the analysis of PI usually relies on the fact that the sequence of the generated values is non-decreasing (Bertsekas & Tsitsiklis, 1996; Munos, 2003). Unfortunately, it can be easily shown that for m finite, the value functions generated by MPI may decrease (it suffices to take a very high initial value). It can be seen from what we just described and Proposition 1 that for $m \neq 1$ and ∞, MPI is neither contracting nor non-decreasing, and thus, a new line of proof is needed for the propagation of error in this algorithm.

[3] We may implement CBMPI more sample efficient by reusing the rollouts generated for the greedy step in the evaluation step. However, in order to simplify the theoretical analysis, we generate two separate sets of rollouts.

To study error propagation in AMPI, we first introduce the errors that occur at each iteration of this algorithm. AMPI starts with an arbitrary value v_0 and at each iteration $k \geq 1$ computes the greedy policy w.r.t. v_{k-1} with some error ϵ'_k, called the *greedy step error*. Thus, we write the new policy π_k as

$$\pi_k = \widehat{\mathcal{G}}_{\epsilon'_k} v_{k-1}. \tag{10}$$

Eq. 10 means that for any policy π', including the true greedy policy w.r.t. v_{k-1}, i.e., $\mathcal{G}v_{k-1}$, we have

$$T_{\pi'} v_{k-1} \leq T_{\pi_k} v_{k-1} + \epsilon'_k.$$

AMPI then generates the new value function v_k with some error ϵ_k, called the *evaluation step error*

$$v_k = (T_{\pi_k})^m v_{k-1} + \epsilon_k. \tag{11}$$

Before showing how these two errors are propagated through the iterations of AMPI, let us first define them in the context of each of the algorithms presented in Section 3 separately.

AMPI-V: ϵ_k is the error in fitting the value function v_k. This error can be further decomposed into two parts: the one related to the approximation power of $\mathcal{F}$ and the one due to the finite number of samples/rollouts. ϵ'_k is the error due to using a finite number of samples M for estimating the greedy actions.

AMPI-Q: $\epsilon'_k = 0$ and ϵ_k is the error in fitting the state-action value function Q_k.

CBMPI: This algorithm iterates as follows:

$$\begin{aligned} v_k &= (T_{\pi_k})^m v_{k-1} + \epsilon_k \\ \pi_{k+1} &= \mathcal{G}\left[(T_{\pi_k})^m v_{k-1}\right] + \epsilon'_{k+1} \end{aligned}$$

Unfortunately, this does not exactly match with the model described in Eqs. 10 and 11. By introducing the auxiliary variable $w_k \triangleq (T_{\pi_k})^m v_{k-1}$, we have $v_k = w_k + \epsilon_k$, and thus, we may write

$$\pi_{k+1} = \widehat{\mathcal{G}}_{\epsilon'_{k+1}}\left[w_k\right]. \tag{12}$$

Using $v_{k-1} = w_{k-1} + \epsilon_{k-1}$, we have

$$\begin{aligned} w_k &= (T_{\pi_k})^m v_{k-1} = (T_{\pi_k})^m (w_{k-1} + \epsilon_{k-1}) \\ &= (T_{\pi_k})^m w_{k-1} + (\gamma P_{\pi_k})^m \epsilon_{k-1}. \end{aligned} \tag{13}$$

Now, Eqs. 12 and 13 exactly match Eqs. 10 and 11 by replacing v_k with w_k and ϵ_k with $(\gamma P_{\pi_k})^m \epsilon_{k-1}$.

The rest of this section is devoted to show how the errors ϵ_k and ϵ'_k propagate through the iterations of an AMPI algorithm. Due to space constraints, we only outline the main arguments that will lead to the performance bound of Thm. 1 and report the proofs in the supplementary material. The results are obtained using the following three quantities:

1) The distance between the optimal value function and the value before approximation at the k^{th} iteration: $d_k \triangleq v_* - (T_{\pi_k})^m v_{k-1} = v_* - (v_k - \epsilon_k)$.

2) The shift between the value before approximation and the value of the policy at the k^{th} iteration: $s_k \triangleq (T_{\pi_k})^m v_{k-1} - v_{\pi_k} = (v_k - \epsilon_k) - v_{\pi_k}$.

3) The Bellman residual at the k^{th} iteration: $b_k \triangleq v_k - T_{\pi_{k+1}} v_k = v_k - T v_k$.

We are interested in finding an upper bound on the **loss** $l_k \triangleq v_* - v_{\pi_k} = d_k + s_k$. To do so, we need to upper bound d_k and s_k, which requires a bound on the Bellman residual b_k. More precisely, the core of the analysis is to prove the following point-wise inequalities for our three quantities of interest.

Lemma 1. *Let $k \geq 1$, $x_k \triangleq (I - \gamma P_{\pi_k})\epsilon_k + \epsilon'_{k+1}$ and $y_k \triangleq -\gamma P_{\pi_*}\epsilon_k + \epsilon'_{k+1}$. We then have the following point-wise inequalities:*

$$\begin{aligned} b_k &\leq (\gamma P_{\pi_k})^m b_{k-1} + x_k, \\ d_{k+1} &\leq \gamma P_{\pi_*} d_k + y_k + \sum_{j=1}^{m-1} (\gamma P_{\pi_{k+1}})^j b_k, \\ s_k &= (\gamma P_{\pi_k})^m (I - \gamma P_{\pi_k})^{-1} b_{k-1}. \end{aligned}$$

Proof. See Appx A in the supplementary material. □

Since the stochastic kernels are non-negative, the bounds in Lemma 1 indicate that the loss l_k will be bounded if the errors ϵ_k and ϵ'_k are controlled. In fact, if we define ϵ as a uniform upper-bound on the errors $|\epsilon_k|$ and $|\epsilon'_k|$, the first inequality in Lemma 1 implies that $b_k \leq O(\epsilon)$, and as a result, the second and third inequalities gives us $d_k \leq O(\epsilon)$ and $s_k \leq O(\epsilon)$. This means that the loss will also satisfy $l_k \leq O(\epsilon)$.

Our bound for the loss l_k is the result of careful expansion and combination of the three inequalities in Lemma 1. Before we state this result, we introduce some notations that will ease our formulation.

Definition 1. *For a positive integer n, we define $\mathbb{P}_n$ as the set of transition kernels that are defined as follows:*

1) *for any set of n policies $\{\pi_1, \ldots, \pi_n\}$, $(\gamma P_{\pi_1})(\gamma P_{\pi_2}) \ldots (\gamma P_{\pi_n}) \in \mathbb{P}_n$,*

2) *for any $\alpha \in (0,1)$ and $(P_1, P_2) \in \mathbb{P}_n \times \mathbb{P}_n$, $\alpha P_1 + (1-\alpha) P_2 \in \mathbb{P}_n$.*

Furthermore, we use the somewhat abusive notation Γ^n for denoting any element of $\mathbb{P}_n$. For example, if we

write a transition kernel P as $P = \alpha_1\Gamma^i + \alpha_2\Gamma^j\Gamma^k = \alpha_1\Gamma^i + \alpha_2\Gamma^{j+k}$, it should be read as there exist $P_1 \in \mathbb{P}_i$, $P_2 \in \mathbb{P}_j$, $P_3 \in \mathbb{P}_k$, and $P_4 \in \mathbb{P}_{k+j}$ such that $P = \alpha_1 P_1 + \alpha_2 P_2 P_3 = \alpha_1 P_1 + \alpha_2 P_4$.

Using the notation introduced in Definition 1, we now derive a point-wise bound on the loss.

Lemma 2. *The loss of the AMPI algorithm after k iterations satisfies*

$$l_k \le 2\sum_{i=1}^{k-1}\sum_{j=i}^{\infty}\Gamma^j|\epsilon_{k-i}| + \sum_{i=0}^{k-1}\sum_{j=i}^{\infty}\Gamma^j|\epsilon'_{k-i}| + h(k),$$

where $h(k) \triangleq 2\sum_{j=k}^{\infty}\Gamma^j|d_0|$ or $h(k) \triangleq 2\sum_{j=k}^{\infty}\Gamma^j|b_0|$.

Proof. See Appx B in the supplementary material. □

Remark 1. A close look at the existing point-wise error bounds for AVI (Munos, 2007, Lemma 4.1) and API (Munos, 2003, Corollary 10) shows that they do not consider error in the greedy step (i.e., $\epsilon'_k = 0$) and that they have the following form:

$$\limsup_{k\to\infty} l_k \le 2\limsup_{k\to\infty}\sum_{i=1}^{\infty}\sum_{j=i}^{\infty}\Gamma^j|\epsilon_{k-i}|.$$

This indicates that the bound in Lemma 2 not only unifies the analysis of AVI and API, but it generalizes them to the case of error in the greedy step and to a finite horizon k. Moreover, our bound suggests that the way the errors are propagated in the whole family of algorithms VI/PI/MPI does not depend on m at the level of the abstraction suggested by Definition 1.[4]

The next step is to show how the point-wise bound of Lemma 2 can turn to a bound in L_p-norm. Munos (2003; 2007); Munos & Szepesvári (2008), and the recent work of Farahmand et al. (2010), which provides the most refined bounds for API and AVI, show how to do this process through quantities, called *concentrability coefficients*, that measure how a distribution over states may concentrate through the dynamics of the MDP. We now state a lemma that generalizes the analysis of Farahmand et al. (2010) to a larger class of concentrability coefficients. We will discuss the potential advantage of this new class in Remark 3. We will also show through the proofs of Thms. 1 and 3, how the result of Lemma 3 provides us with a flexible tool for turning point-wise bounds into L_p-norm bounds. Thm. 3 that we do not report in the paper (see Appendix D) provides an alternative bound for the loss of AMPI, which in analogy with the results of Farahmand et al. (2010) shows that the last iterations have the highest impact on the loss (the influence exponentially decreases towards the initial iterations).

[4] Note however that the dependence on m will reappear if we make explicit what is hidden in the terms Γ^j.

Lemma 3. *Let $\mathcal{I}$ and $(\mathcal{J}_i)_{i\in\mathcal{I}}$ be sets of positive integers, $\{\mathcal{I}_1, \dots, \mathcal{I}_n\}$ be a partition of $\mathcal{I}$, and f and $(g_i)_{i\in\mathcal{I}}$ be functions satisfying*

$$|f| \le \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}_i}\Gamma^j|g_i| = \sum_{l=1}^{n}\sum_{i\in\mathcal{I}_l}\sum_{j\in\mathcal{J}_i}\Gamma^j|g_i|.$$

Then for all p, q and q' such that $\frac{1}{q} + \frac{1}{q'} = 1$, and for all distributions ρ and μ, we have

$$\|f\|_{p,\rho} \le \sum_{l=1}^{n}\left(\mathcal{C}_q(l)\right)^{1/p}\sup_{i\in\mathcal{I}_l}\|g_i\|_{pq',\mu}\sum_{i\in\mathcal{I}_l}\sum_{j\in\mathcal{J}_i}\gamma^j,$$

with the following concentrability coefficients

$$\mathcal{C}_q(l) \triangleq \frac{\sum_{i\in\mathcal{I}_l}\sum_{j\in\mathcal{J}_i}\gamma^j c_q(j)}{\sum_{i\in\mathcal{I}_l}\sum_{j\in\mathcal{J}_i}\gamma^j},$$

in which $c_q(j)$ is defined as

$$c_q(j) = \max_{\pi_1,\cdots,\pi_j}\left\|\frac{\partial(\rho P_{\pi_1}P_{\pi_2}\cdots P_{\pi_j})}{\partial\mu}\right\|_{q,\mu}. \tag{14}$$

Proof. See Appx C in the supplementary material. □

We now derive a L_p-norm bound for the loss of the AMPI algorithm by applying Lemma 3 to the point-wise bound of Lemma 2.

Theorem 1. *Let ρ and μ be distributions over states. Then for all integers p, q, and q' such that $\frac{1}{q} + \frac{1}{q'} = 1$, and after k iterations, the loss of AMPI satisfies*

$$\|l_k\|_{p,\rho} \le \frac{2(\gamma-\gamma^k)\left(\mathcal{C}_q^{1,k}\right)^{\frac{1}{p}}}{(1-\gamma)^2}\sup_{1\le j\le k-1}\|\epsilon_j\|_{pq',\mu} + \frac{(1-\gamma^k)\left(\mathcal{C}_q^{0,k}\right)^{\frac{1}{p}}}{(1-\gamma)^2}\sup_{1\le j\le k}\|\epsilon'_j\|_{pq',\mu} + g(k), \tag{15}$$

where for all q, l, and k, the concentrability coefficients $\mathcal{C}_q^{l,k}$ are defined as

$$\mathcal{C}_q^{l,k} \triangleq \frac{(1-\gamma)^2}{\gamma^l-\gamma^k}\sum_{i=l}^{k-1}\sum_{j=i}^{\infty}\gamma^j c_q(j),$$

with $c_q(j)$ given by Eq. 14, and $g(k)$ is defined as $g(k) \triangleq \frac{2\gamma^k}{1-\gamma}\left(\mathcal{C}_q^{k,k+1}\right)^{\frac{1}{p}}\min\left(\|d_0\|_{pq',\mu}, \|b_0\|_{pq',\mu}\right)$.

Proof. See Appx D in the supplementary material. □

Remark 2. When p tends to infinity, the bound of Thm. 1 reduces to

$$\|l_k\|_\infty \le \frac{2(\gamma-\gamma^k)}{(1-\gamma)^2}\sup_{1\le j\le k-1}\|\epsilon_j\|_\infty + \frac{1-\gamma^k}{(1-\gamma)^2}\sup_{1\le j\le k}\|\epsilon'_j\|_\infty + \frac{2\gamma^k}{1-\gamma}\min(\|d_0\|_\infty, \|b_0\|_\infty). \quad (16)$$

When k goes to infinity, Eq. 16 gives us a generalization of the API ($m=\infty$) bound of Bertsekas & Tsitsiklis (1996, Prop. 6.2), i.e.,

$$\limsup_{k\to\infty}\|l_k\|_\infty \le \frac{2\gamma\sup_j\|\epsilon_j\|_\infty + \sup_j\|\epsilon'_j\|_\infty}{(1-\gamma)^2}.$$

Moreover, since our point-wise analysis generalizes those of API and AVI (as noted in Remark 1), the L_p-bound of Eq. 15 unifies and generalizes those for API (Munos, 2003) and AVI (Munos, 2007).

Remark 3. We can balance the influence of the concentrability coefficients (the bigger the q, the higher the influence) and the difficulty of controlling the errors (the bigger the q', the greater the difficulty in controlling the $L_{pq'}$-norms) by tuning the parameters q and q', given the condition that $\frac{1}{q}+\frac{1}{q'}=1$. This potential leverage is an improvement over the existing bounds and concentrability results that only consider specific values of these two parameters: $q=\infty$ and $q'=1$ in Munos (2007); Munos & Szepesvári (2008), and $q=q'=2$ in Farahmand et al. (2010).

5. Finite-Sample Analysis of CBMPI

In this section, we provide a finite-sample analysis of CBMPI. The reason for selecting CBMPI among the proposed algorithms is that its analysis is more general than the others as we need to bound both greedy $||\epsilon'_k||$ and evaluation $||\epsilon_k||$ step errors (in some norm) defined in Section 4 (Eqs. 12 and 13). We first provide a bound on the *greedy step error* $||\epsilon'_k||$. From the definition of ϵ'_k for CBMPI (Eq. 12) and the description of the greedy step in CBMPI, we can easily observe that $||\epsilon'_k||_{1,\mu} = \mathcal{L}^\Pi_{k-1}(\mu;\pi_k)$.

Lemma 4 (Greedy step error of CBMPI). *Let Π be a policy space with finite VC-dimension $h = VC(\Pi)$ and μ be a distribution over the state space $\mathcal{S}$. Let N be the number of states in $\mathcal{D}'_{k-1}$ drawn i.i.d. from μ, M be the number of rollouts per state-action pair used in the estimation of $\widehat{Q}_{k-1}$, and $\pi_k = \operatorname{argmin}_{\pi\in\Pi}\widehat{\mathcal{L}}^\Pi_{k-1}(\widehat{\mu},\pi)$ be the policy computed at iteration $k-1$ of CBMPI. Then, for any $\delta>0$, we have*

$$||\epsilon'_k||_{1,\mu} = \mathcal{L}^\Pi_{k-1}(\mu;\pi_k) \le \inf_{\pi\in\Pi}\mathcal{L}^\Pi_{k-1}(\mu;\pi) + 2(\epsilon'_1+\epsilon'_2),$$

with probability at least $1-\delta$, where

$$\epsilon'_1 = 16Q_{\max}\sqrt{\frac{2}{N}\Big(h\log\frac{eN}{h}+\log\frac{32}{\delta}\Big)},$$

$$\epsilon'_2 = 8Q_{\max}\sqrt{\frac{2}{MN}\Big(h\log\frac{eMN}{h}+\log\frac{32}{\delta}\Big)}.$$

Proof. See Appx E in the supplementary material. □

We now bound the *evaluation step error* $||\epsilon_k||$. Recall from Sec. 4 that $\epsilon_k = (\gamma P_{\pi_k})^m\eta_{k-1}$, where $\eta_{k-1} = v_{k-1} - (T_{\pi_{k-1}})^m v_{k-2}$. The evaluation step at iteration $k-1$ of CBMPI is a regression problem with the target $(T_{\pi_{k-1}})^m v_{k-2}$ and a training set $\big\{\big(s^{(i)}, \widehat{v}_{k-1}(s^{(i)})\big)\big\}_{i=1}^n$ in which the states $s^{(i)}$ are i.i.d. samples from μ and $\widehat{v}_{k-1}(s^{(i)})$ are unbiased estimates of the target computed according to Eq. 7. Different function spaces $\mathcal{F}$ (linear or non-linear) may be used to approximate $(T_{\pi_{k-1}})^m v_{k-2}$. Here we use a linear architecture with parameters $\alpha\in\mathbb{R}^d$ and bounded (by L) basis functions $\{\varphi_j\}_{j=1}^d$, $||\varphi_j||_\infty \le L$. We denote by $\phi:\mathcal{X}\to\mathbb{R}^d$, $\phi(\cdot) = \big(\varphi_1(\cdot),\dots,\varphi_d(\cdot)\big)^\top$ the feature vector, and by $\mathcal{F}$ the linear function space spanned by the features φ_j, i.e., $\mathcal{F} = \{f_\alpha(\cdot) = \phi(\cdot)^\top\alpha : \alpha\in\mathbb{R}^d\}$. Now if we define v_{k-1} as the truncation (by $V_{\max}$) of the solution of the above linear regression problem, we may bound $||\eta_{k-1}||$ using the following lemma.

Lemma 5 (Evaluation step error of CBMPI). *Consider the linear regression setting described above, then we have*

$$||\eta_{k-1}||_{2,\mu} \le 4\inf_{f\in\mathcal{F}}||(T_{\pi_{k-1}})^m v_{k-2} - f||_{2,\mu} + \epsilon_1 + \epsilon_2,$$

with probability at least $1-\delta$, where

$$\epsilon_1 = 32V_{\max}\sqrt{\frac{2}{n}\log\Big(\frac{27(12e^2n)^{2(d+1)}}{\delta}\Big)},$$

$$\epsilon_2 = 24\Big(V_{\max} + ||\alpha_*||_2\cdot\sup_x||\phi(x)||_2\Big)\sqrt{\frac{2}{n}\log\frac{9}{\delta}},$$

and α_* is such that f_{α_*} is the best approximation (w.r.t. μ) of the target function $(T_{\pi_{k-1}})^m v_{k-2}$ in $\mathcal{F}$.

Proof. See Appx F in the supplementary material. □

Now using the bounds of Lemmas 4 and 5 on $||\epsilon'_k||$ and $||\eta_k||$, we derive a bound on the performance of CBMPI similar to the one in Thm. 1. Since $||\epsilon'_k||_{1,\mu} \le ||\epsilon'_k||_{2,\mu}$, we bound $||\epsilon'_k||$ and $||\eta_k||$ in L_2-norm. In the analogy of Thm. 1, this means $pq'=2$, which gives us either $p=2$, $q'=1$, and $q=\infty$ or $p=1$ and $q=q'=2$. In Thm. 2, we report the bound for the former case. In order to prove Thm. 2, we first derive a point-wise bound similar to the one in Lemma 2 for CBMPI that takes into account the fact that $\epsilon_k = (\gamma P_{\pi_k})^m\eta_{k-1}$. Using the obtained point-wise result, we derive a L_p-bound (similar to Lemma 3) on the loss of CBMPI, and then plug in our bounds on $||\epsilon'_k||$ and $||\eta_k||$ into it.

Theorem 2. *If π_K is the policy obtained after K iteration of CMPI, we have*

$$||v_* - v_{\pi_K}||_{2,\rho} \leq \frac{(1-\gamma^K)\left(\mathcal{C}_\infty^{1,K,0}\right)^{\frac{1}{2}}}{(1-\gamma)^2} \sup_{1\leq j\leq K} \|\epsilon'_j\|_{2,\mu}$$
$$+ \frac{2\gamma^m(\gamma-\gamma^{K-1})\left(\mathcal{C}_\infty^{2,K,m}\right)^{\frac{1}{2}}}{(1-\gamma)^2} \sup_{1\leq j\leq K-2} \|\eta_j\|_{2,\mu} + g(K),$$

with probability at least $1-\delta$, where for all q, l, k, and d, the concentrability coefficients $\mathcal{C}_q^{l,k,d}$ are defined as

$$\mathcal{C}_q^{l,k,d} \triangleq \frac{(1-\gamma)^2}{\gamma^{l+d}-\gamma^{k+d}} \sum_{i=l}^{k-1} \sum_{j=i+d}^{\infty} \gamma^{j+d} c_q(j+d).$$

Proof. See Appx G in the supplementary material. □

Remark 4. It is interesting to see that the parameter m of MPI plays an important role (through the term γ^m) to control the balance of errors (the one in value function approximation ϵ or η and the one in estimating the greedy policy ϵ') in the final performance of CBMPI. This is an important feature of CBMPI that requires more investigation. Using this feature becomes more interesting when we are given a fixed budget (number of rollouts or sample transitions) and supposed to obtain the best performance by tuning m.

6. Summary and Extensions

In this paper, we studied a DP algorithm, called modified policy iteration (MPI), that despite its generality that contains the celebrated policy and value iteration methods, has not been thoroughly investigated in the literature. We proposed three approximate MPI (AMPI) algorithms that are extensions of the well-known ADP algorithms: fitted-value iteration, fitted-Q iteration, and classification-based policy iteration. We reported an error propagation analysis for AMPI that unifies those for approximate policy and value iteration. We also provided a finite-sample analysis for the classification-based implementation of AMPI (CBMPI), whose analysis is more general than the other presented AMPI methods. Our results indicate that the parameter of MPI allows us to control the balance of errors (in value function approximation and estimation of the greedy policy) in the final performance of CBMPI. Although AMPI generalizes the existing AVI and classification-based API algorithms, additional experimental work and careful theoretical analysis are required to obtain a better understanding of the behaviour of its different implementations and their relation to the competitive methods. Extension of CBMPI to problems with continuous action space is another interesting direction to pursue.

References

Antos, A., Munos, R., and Szepesvári, Cs. Fitted Q-iteration in continuous action-space MDPs. In *Proceedings of NIPS*, pp. 9–16, 2007a.

Antos, A., Szepesvári, Cs., and Munos, R. Value-iteration based fitted policy iteration: Learning with a single trajectory. In *ADPRL 2007*, pp. 330–337. IEEE, 2007b.

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

Bertsekas, D. and Tsitsiklis, J. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

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

Farahmand, A., Munos, R., and Szepesvári, Cs. Error propagation for approximate policy and value iteration. In *Proceedings of NIPS*, pp. 568–576, 2010.

Fern, A., Yoon, S., and Givan, R. Approximate Policy Iteration with a Policy Language Bias: Solving Relational Markov Decision Processes. *Journal of Artificial Intelligence Research*, 25:75–118, 2006.

Gabillon, V., Lazaric, A., Ghavamzadeh, M., and Scherrer, B. Classification-based policy iteration with a critic. In *Proceedings of ICML*, pp. 1049–1056, 2011.

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

Lagoudakis, M. and Parr, R. Reinforcement Learning as Classification: Leveraging Modern Classifiers. In *Proceedings of ICML*, pp. 424–431, 2003b.

Lazaric, A., Ghavamzadeh, M., and Munos, R. Analysis of a Classification-based Policy Iteration Algorithm. In *Proceedings of ICML*, pp. 607–614, 2010.

Munos, R. Error Bounds for Approximate Policy Iteration. In *Proceedings of ICML*, pp. 560–567, 2003.

Munos, R. Performance Bounds in Lp-norm for Approximate Value Iteration. *SIAM J. Control and Optimization*, 46(2):541–561, 2007.

Munos, R. and Szepesvári, Cs. Finite-Time Bounds for Fitted Value Iteration. *Journal of Machine Learning Research*, 9:815–857, 2008.

Puterman, M. and Shin, M. Modified policy iteration algorithms for discounted Markov decision problems. *Management Science*, 24(11), 1978.

Szepesvári, Cs. Reinforcement Learning Algorithms for MDPs. In *Wiley Encyclopedia of Operations Research*. Wiley, 2010.

Thiery, C. and Scherrer, B. Least-Squares λ-Policy Iteration: Bias-Variance Trade-off in Control Problems. In *Proceedings of ICML*, Haifa, Israël, 2010.

Supplementary Material for Approximate Modified Policy Iteration

A. Proof of Lemma 1

Before we start, we recall the following definitions:

$$b_k = v_k - T_{\pi_{k+1}} v_k, \qquad d_k = v_* - (T_{\pi_k})^m v_{k-1} = v_* - (v_k - \epsilon_k), \qquad s_k = (T_{\pi_k})^m v_{k-1} - v_{\pi_k} = (v_k - \epsilon_k) - v_{\pi_k}.$$

Bounding b_k

$$\begin{aligned} b_k &= v_k - T_{\pi_{k+1}} v_k = v_k - T_{\pi_k} v_k + T_{\pi_k} v_k - T_{\pi_{k+1}} v_k \overset{(a)}{\leq} v_k - T_{\pi_k} v_k + \epsilon'_{k+1} \\ &= v_k - \epsilon_k - T_{\pi_k} v_k + \gamma P_{\pi_k} \epsilon_k + \epsilon_k - \gamma P_{\pi_k} \epsilon_k + \epsilon'_{k+1} \overset{(b)}{=} v_k - \epsilon_k - T_{\pi_k}(v_k - \epsilon_k) + (I - \gamma P_{\pi_k})\epsilon_k + \epsilon'_{k+1}. \end{aligned} \tag{17}$$

Using the definition of x_k, i.e.,

$$x_k \triangleq (I - \gamma P_{\pi_k})\epsilon_k + \epsilon'_{k+1}, \tag{18}$$

we may write Eq. (17) as

$$\begin{aligned} b_k &\leq v_k - \epsilon_k - T_{\pi_k}(v_k - \epsilon_k) + x_k \overset{(c)}{=} (T_{\pi_k})^m v_{k-1} - T_{\pi_k}(T_{\pi_k})^m v_{k-1} + x_k = (T_{\pi_k})^m v_{k-1} - (T_{\pi_k})^m (T_{\pi_k} v_{k-1}) + x_k \\ &= (\gamma P_{\pi_k})^m (v_{k-1} - T_{\pi_k} v_{k-1}) + x_k = (\gamma P_{\pi_k})^m b_{k-1} + x_k. \end{aligned} \tag{19}$$

(a) From the definition of ϵ'_{k+1}, we have $\forall \pi' \;\; T_{\pi'} v_k \leq T_{\pi_{k+1}} v_k + \epsilon'_{k+1}$, thus this inequality holds also for $\pi' = \pi_k$.

(b) This step is due to the fact that for every v and v', we have $T_{\pi_k}(v + v') = T_{\pi_k} v + \gamma P_{\pi_k} v'$.

(c) This is from the definition of ϵ_k, i.e., $v_k = (T_{\pi_k})^m v_{k-1} + \epsilon_k$.

Bounding d_k

$$\begin{aligned} d_{k+1} &= v_* - (T_{\pi_{k+1}})^m v_k = T_{\pi_*} v_* - T_{\pi_*} v_k + T_{\pi_*} v_k - T_{\pi_{k+1}} v_k + T_{\pi_{k+1}} v_k - (T_{\pi_{k+1}})^m v_k \\ &\overset{(a)}{\leq} \gamma P_{\pi_*}(v_* - v_k) + \epsilon'_{k+1} + g_{k+1} = \gamma P_{\pi_*}(v_* - v_k) + \gamma P_{\pi_*} \epsilon_k - \gamma P_{\pi_*} \epsilon_k + \epsilon'_{k+1} + g_{k+1} \\ &\overset{(b)}{=} \gamma P_{\pi_*}\big(v_* - (v_k - \epsilon_k)\big) + y_k + g_{k+1} = \gamma P_{\pi_*} d_k + y_k + g_{k+1} \overset{(c)}{=} \gamma P_{\pi_*} d_k + y_k + \sum_{j=1}^{m-1} (\gamma P_{\pi_{k+1}})^j b_k. \end{aligned} \tag{20}$$

(a) This step is from the definition of ϵ'_{k+1} (see step **(a)** in bounding b_k) and by defining g_{k+1} as follows:

$$g_{k+1} \triangleq T_{\pi_{k+1}} v_k - (T_{\pi_{k+1}})^m v_k. \tag{21}$$

(b) This is from the definition of y_k, i.e.,

$$y_k \triangleq -\gamma P_{\pi_*} \epsilon_k + \epsilon'_{k+1}. \tag{22}$$

(c) This step comes from rewriting g_{k+1} as

$$\begin{aligned} g_{k+1} &= T_{\pi_{k+1}} v_k - (T_{\pi_{k+1}})^m v_k = \sum_{j=1}^{m-1} \left[(T_{\pi_{k+1}})^j v_k - (T_{\pi_{k+1}})^{j+1} v_k\right] = \sum_{j=1}^{m-1} \left[(T_{\pi_{k+1}})^j v_k - (T_{\pi_{k+1}})^j (T_{\pi_{k+1}} v_k)\right] \\ &= \sum_{j=1}^{m-1} (\gamma P_{\pi_{k+1}})^j (v_k - T_{\pi_{k+1}} v_k) = \sum_{j=1}^{m-1} (\gamma P_{\pi_{k+1}})^j b_k. \end{aligned} \tag{23}$$

Bounding s_k With some slight abuse of notation, we have

$$v_{\pi_k} = (T_{\pi_k})^\infty v_k$$

and thus:

$$\begin{aligned}
s_k &= (T_{\pi_k})^m v_{k-1} - v_{\pi_k} \overset{(a)}{=} (T_{\pi_k})^m v_{k-1} - (T_{\pi_k})^\infty v_{k-1} = (T_{\pi_k})^m v_{k-1} - (T_{\pi_k})^m (T_{\pi_k})^\infty v_{k-1} \\
&= (\gamma P_{\pi_k})^m \big(v_{k-1} - (T_{\pi_k})^\infty v_{k-1}\big) = (\gamma P_{\pi_k})^m \sum_{j=0}^{\infty} \big[(T_{\pi_k})^j v_{k-1} - (T_{\pi_k})^{j+1} v_{k-1}\big] \\
&= (\gamma P_{\pi_k})^m \big(\sum_{j=0}^{\infty} \big[(T_{\pi_k})^j v_{k-1} - (T_{\pi_k})^j T_{\pi_k} v_{k-1}\big] = (\gamma P_{\pi_k})^m \Big(\sum_{j=0}^{\infty} (\gamma P_{\pi_k})^j\Big)(v_{k-1} - T_{\pi_k} v_{k-1}) \\
&= (\gamma P_{\pi_k})^m (I - \gamma P_{\pi_k})^{-1} (v_{k-1} - T_{\pi_k} v_{k-1}) = (\gamma P_{\pi_k})^m (I - \gamma P_{\pi_k})^{-1} b_{k-1}. \qquad (24)
\end{aligned}$$

(a) For any v, we have $v_{\pi_k} = (T_{\pi_k})^\infty v$. This step follows by setting $v = v_{k-1}$, i.e., $v_{\pi_k} = (T_{\pi_k})^\infty v_{k-1}$.

B. Proof of Lemma 2

Here we are interested in bounding the loss $l_k = v_* - v_{\pi_k} = d_k + s_k$.

By induction, from Eqs. (19) and (20), we obtain

$$b_k \leq \sum_{i=1}^{k} \Gamma^{m(k-i)} x_i + \Gamma^{mk} b_0, \tag{25}$$

$$d_k \leq \sum_{j=0}^{k-1} \Gamma^{k-1-j} \Big(y_j + \sum_{l=1}^{m-1} \Gamma^l b_j \Big) + \Gamma^k d_0. \tag{26}$$

in which we have used the notation introduced in Definition 1. In Eq. (26), we also used the fact that from Eq. (23), we may write $g_{k+1} = \sum_{j=1}^{m-1} \Gamma^j b_k$. Moreover, we may rewrite Eq. (24) as

$$s_k = \Gamma^m \sum_{j=0}^{\infty} \Gamma^j b_{k-1} = \sum_{j=0}^{\infty} \Gamma^{m+j} b_{k-1}. \tag{27}$$

Bounding l_k From Eqs. (25) and (26), we may write

$$\begin{aligned} d_k &\leq \sum_{j=0}^{k-1} \Gamma^{k-1-j} \left(y_j + \sum_{l=1}^{m-1} \Gamma^l \Big(\sum_{i=1}^{j} \Gamma^{m(j-i)} x_i + \Gamma^{mj} b_0 \Big) \right) + \Gamma^k d_0 \\ &= \sum_{i=1}^{k} \Gamma^{i-1} y_{k-i} + \sum_{j=0}^{k-1} \sum_{l=1}^{m-1} \sum_{i=1}^{j} \Gamma^{k-1-j+l+m(j-i)} x_i + z_k, \end{aligned} \tag{28}$$

where we used the following definition

$$z_k \triangleq \sum_{j=0}^{k-1} \sum_{l=1}^{m-1} \Gamma^{k-1+l+j(m-1)} b_0 + \Gamma^k d_0 = \sum_{i=k}^{mk-1} \Gamma^i b_0 + \Gamma^k d_0.$$

The triple sum involved in Eq. (28) may be written as

$$\begin{aligned} \sum_{j=0}^{k-1} \sum_{l=1}^{m-1} \sum_{i=1}^{j} \Gamma^{k-1-j+l+m(j-i)} x_i &= \sum_{i=1}^{k-1} \sum_{j=i}^{k-1} \sum_{l=1}^{m-1} \Gamma^{k-1+l+j(m-1)-mi} x_i = \sum_{i=1}^{k-1} \sum_{j=mi+k-i}^{mk-1} \Gamma^{j-mi} x_i \\ &= \sum_{i=1}^{k-1} \sum_{j=k-i}^{m(k-i)-1} \Gamma^j x_i = \sum_{i=1}^{k-1} \sum_{j=i}^{mi-1} \Gamma^j x_{k-i}. \end{aligned} \tag{29}$$

Using Eq. (29), we may write Eq. (28) as

$$d_k \leq \sum_{i=1}^{k} \Gamma^{i-1} y_{k-i} + \sum_{i=1}^{k-1} \sum_{j=i}^{mi-1} \Gamma^j x_{k-i} + z_k. \tag{30}$$

Similarly, from Eqs. (27) and (25), we have

$$\begin{aligned} s_k &\leq \sum_{j=0}^{\infty} \Gamma^{m+j} \Big(\sum_{i=1}^{k-1} \Gamma^{m(k-1-i)} x_i + \Gamma^{m(k-1)} b_0 \Big) = \sum_{j=0}^{\infty} \Big(\sum_{i=1}^{k-1} \Gamma^{m+j+m(k-1-i)} x_i + \Gamma^{m+j+m(k-1)} b_0 \Big) \\ &= \sum_{i=1}^{k-1} \sum_{j=0}^{\infty} \Gamma^{j+m(k-i)} x_i + \sum_{j=0}^{\infty} \Gamma^{j+mk} b_0 = \sum_{i=1}^{k-1} \sum_{j=0}^{\infty} \Gamma^{j+mi} x_{k-i} + \sum_{j=mk}^{\infty} \Gamma^j b_0 = \sum_{i=1}^{k-1} \sum_{j=mi}^{\infty} \Gamma^j x_{k-i} + z'_k, \end{aligned} \tag{31}$$

where we used the following definition

$$z'_k \triangleq \sum_{j=mk}^{\infty} \Gamma^j b_0.$$

Finally, using the bounds in Eqs. (30) and (31), we obtain the following bound on the loss

$$\begin{aligned} l_k \leq d_k + s_k &\leq \sum_{i=1}^{k} \Gamma^{i-1} y_{k-i} + \sum_{i=1}^{k-1} \Big(\sum_{j=i}^{mi-1} \Gamma^j + \sum_{j=mi}^{\infty} \Gamma^j \Big) x_{k-i} + z_k + z'_k \\ &= \sum_{i=1}^{k} \Gamma^{i-1} y_{k-i} + \sum_{i=1}^{k-1} \sum_{j=i}^{\infty} \Gamma^j x_{k-i} + \eta_k, \end{aligned} \tag{32}$$

where we used the following definition

$$\eta_k \triangleq z_k + z'_k = \sum_{j=k}^{\infty} \Gamma^j b_0 + \Gamma^k d_0. \tag{33}$$

Note that we have the following relation between b_0 and d_0

$$b_0 = v_0 - T_{\pi_1} v_0 = v_0 - v_* + T_{\pi_*} v_* - T_{\pi_*} v_0 + T_{\pi_*} v_0 - T_{\pi_1} v_0 \leq (I - \gamma P_{\pi_*})(-d_0) + \epsilon'_1, \tag{34}$$

In Eq. (34), we used the fact that $v_* = T_{\pi_*} v_*$, $\epsilon_0 = 0$, and $T_{\pi_*} v_0 - T_{\pi_1} v_0 \leq \epsilon'_1$ (this is because the policy π_1 is ϵ'_1-greedy w.r.t. v_0). As a result, we may write $|\eta_k|$ either as

$$|\eta_k| \leq \sum_{j=k}^{\infty} \Gamma^j \big[(I - \gamma P_{\pi_*}) |d_0| + |\epsilon'_1| \big] + \Gamma^k |d_0| \leq \sum_{j=k}^{\infty} \Gamma^j \big[(I + \Gamma^1) |d_0| + |\epsilon'_1| \big] + \Gamma^k |d_0| = 2 \sum_{j=k}^{\infty} \Gamma^j |d_0| + \sum_{j=k}^{\infty} \Gamma^j |\epsilon'_1|, \tag{35}$$

or using the fact that from Eq. (34), we have $d_0 \leq (I - \gamma P_{\pi_*})^{-1} (-b_0 + \epsilon'_1)$, as

$$|\eta_k| \leq \sum_{j=k}^{\infty} \Gamma^j |b_0| + \Gamma^k \sum_{j=0}^{\infty} (\gamma P_{\pi_*})^j \big(|b_0| + |\epsilon'_1| \big) = \sum_{j=k}^{\infty} \Gamma^j |b_0| + \Gamma^k \sum_{j=0}^{\infty} \Gamma^j \big(|b_0| + |\epsilon'_1| \big) = 2 \sum_{j=k}^{\infty} \Gamma^j |b_0| + \sum_{j=k}^{\infty} \Gamma^j |\epsilon'_1|. \tag{36}$$

Now, using the definitions of x_k and y_k in Eqs. (18) and (22), the bound on $|\eta_k|$ in Eq. (35) or (36), and the fact that $\epsilon_0 = 0$, we obtain

$$\begin{aligned} |l_k| &\leq \sum_{i=1}^{k} \Gamma^{i-1} \big[\Gamma^1 |\epsilon_{k-i}| + |\epsilon'_{k-i+1}| \big] + \sum_{i=1}^{k-1} \sum_{j=i}^{\infty} \Gamma^j \big[(I + \Gamma^1) |\epsilon_{k-i}| + |\epsilon'_{k-i+1}| \big] + |\eta_k| \\ &= \sum_{i=1}^{k-1} \Big(\Gamma^i + \sum_{j=i}^{\infty} (\Gamma^j + \Gamma^{j+1}) \Big) |\epsilon_{k-i}| + \Gamma^k |\epsilon_0| + \sum_{i=1}^{k-1} \Big(\Gamma^{i-1} + \sum_{j=i}^{\infty} \Gamma^j \Big) |\epsilon'_{k-i+1}| + \Gamma^{k-1} |\epsilon'_1| + \sum_{j=k}^{\infty} \Gamma^j |\epsilon'_1| + h(k) \\ &= 2 \sum_{i=1}^{k-1} \sum_{j=i}^{\infty} \Gamma^j |\epsilon_{k-i}| + \sum_{i=1}^{k-1} \sum_{j=i-1}^{\infty} \Gamma^j |\epsilon'_{k-i+1}| + \sum_{j=k-1}^{\infty} \Gamma^j |\epsilon'_1| + h(k) = 2 \sum_{i=1}^{k-1} \sum_{j=i}^{\infty} \Gamma^j |\epsilon_{k-i}| + \sum_{i=0}^{k-1} \sum_{j=i}^{\infty} \Gamma^j |\epsilon'_{k-i}| + h(k), \end{aligned} \tag{37}$$

where we used the following definition

$$h(k) \triangleq 2 \sum_{j=k}^{\infty} \Gamma^j |d_0|, \qquad \text{or} \qquad h(k) \triangleq 2 \sum_{j=k}^{\infty} \Gamma^j |b_0|.$$

C. Proof of Lemma 3

For any integer t and vector z, the definition of Γ^t and the Hölder's inequality imply that

$$\rho\Gamma^t|z| = \left\|\Gamma^t|z|\right\|_{1,\rho} \leq \gamma^t c_q(t)\|z\|_{q',\mu} = \gamma^t c_q(t)\left(\mu|z|^{q'}\right)^{\frac{1}{q'}}. \tag{38}$$

We define

$$K \triangleq \sum_{l=1}^{n} \xi_l \left(\sum_{i\in\mathcal{I}_l}\sum_{j\in\mathcal{J}_i}\gamma^j\right),$$

where $\{\xi_l\}_{l=1}^n$ is a set of non-negative numbers that we will specify later. We now have

$$\begin{aligned}
\|f\|_{p,\rho}^p &= \rho|f|^p \\
&\leq K^p\rho\left(\frac{\sum_{l=1}^n\sum_{i\in\mathcal{I}_l}\sum_{j\in\mathcal{J}_i}\Gamma^j|g_i|}{K}\right)^p = K^p\rho\left(\frac{\sum_{l=1}^n\xi_l\sum_{i\in\mathcal{I}_l}\sum_{j\in\mathcal{J}_i}\Gamma^j\left(\frac{|g_i|}{\xi_l}\right)}{K}\right)^p \\
&\overset{(a)}{\leq} K^p\rho\frac{\sum_{l=1}^n\xi_l\sum_{i\in\mathcal{I}_l}\sum_{j\in\mathcal{J}_i}\Gamma^j\left(\frac{|g_i|}{\xi_l}\right)^p}{K} = K^p\frac{\sum_{l=1}^n\xi_l\sum_{i\in\mathcal{I}_l}\sum_{j\in\mathcal{J}_i}\rho\Gamma^j\left(\frac{|g_i|}{\xi_l}\right)^p}{K} \\
&\overset{(b)}{\leq} K^p\frac{\sum_{l=1}^n\xi_l\sum_{i\in\mathcal{I}_l}\sum_{j\in\mathcal{J}_i}\gamma^j c_q(j)\left(\mu\left(\frac{|g_i|}{\xi_l}\right)^{pq'}\right)^{\frac{1}{q'}}}{K} \\
&= K^p\frac{\sum_{l=1}^n\xi_l\sum_{i\in\mathcal{I}_l}\sum_{j\in\mathcal{J}_i}\gamma^j c_q(j)\left(\frac{\|g_i\|_{pq',\mu}}{\xi_l}\right)^p}{K} \\
&\leq K^p\frac{\sum_{l=1}^n\xi_l\left(\sum_{i\in\mathcal{I}_l}\sum_{j\in\mathcal{J}_i}\gamma^j c_q(j)\right)\left(\frac{\sup_{i\in\mathcal{I}_l}\|g_i\|_{pq',\mu}}{\xi_l}\right)^p}{K} \\
&\overset{(c)}{=} K^p\frac{\sum_{l=1}^n\xi_l\left(\sum_{i\in\mathcal{I}_l}\sum_{j\in\mathcal{J}_i}\gamma^j\right)\mathcal{C}_q(l)\left(\frac{\sup_{i\in\mathcal{I}_l}\|g_i\|_{pq',\mu}}{\xi_l}\right)^p}{K},
\end{aligned}$$

where **(a)** results from Jensen's inequality, **(b)** from Eq. 38, and **(c)** from the definition of $\mathcal{C}_q(l)$. Now, by setting $\xi_l = \left(\mathcal{C}_q(l)\right)^{1/p}\sup_{i\in\mathcal{I}_l}\|g_i\|_{pq',\mu}$, we obtain

$$\|f\|_{p,\rho}^p \leq K^p\frac{\sum_{l=1}^n\xi_l\left(\sum_{i\in\mathcal{I}_l}\sum_{j\in\mathcal{J}_i}\gamma^j\right)}{K} = K^p,$$

where the last step follows from the definition of K.

D. Proof of Theorem 1 & Another Bound on the Loss of AMPI

Proof. We define $\mathcal{I} = \{1, 2, \cdots, 2k\}$, the partition $\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3\}$ as $\mathcal{I}_1 = \{1, \dots, k-1\}$, $\mathcal{I}_2 = \{k, \dots, 2k-1\}$, and $\mathcal{I}_3 = \{2k\}$, and for each $i \in \mathcal{I}$

$$g_i = \begin{cases} 2\epsilon_{k-i} & \text{if} \quad 1 \le i \le k-1, \\ \epsilon'_{k-(i-k)} & \text{if} \quad k \le i \le 2k-1, \\ 2d_0 \text{ (or } 2b_0) & \text{if} \quad i = 2k, \end{cases} \qquad \text{and} \qquad \mathcal{J}_i = \begin{cases} \{i, i+1, \cdots\} & \text{if} \quad 1 \le i \le k-1, \\ \{i-k, i-k+1, \cdots\} & \text{if} \quad k \le i \le 2k-1, \\ \{k, k+1, \cdots\} & \text{if} \quad i = 2k. \end{cases}$$

Note that here we have divided the terms in the point-wise bound of Lemma 2 into three groups: the *evaluation error* terms $\{\epsilon_j\}_{j=1}^{k-1}$, the *greedy step error* terms $\{\epsilon'_j\}_{j=1}^{k}$, and finally the residual term $h(k)$. With the above definitions and the fact that the loss l_k is non-negative, Lemma 2 may be rewritten as

$$|l_k| \le \sum_{l=1}^{3} \sum_{i \in \mathcal{I}_l} \sum_{j \in \mathcal{J}_i} \Gamma^j |g_i|.$$

The result follows by applying Lemma 3. □

Here in oder to show the flexibility of Lemma 3, we group the terms differently and derive an alternative L_p-bound for the loss of the AMPI algorithms. In analogy with the results of Farahmand et al. (2010), this new bound shows that the last iterations have the highest influence on the loss (the influence exponentially decreases towards the initial iterations).

Theorem 3. *With the notations of Theorem 1, the loss after k iterations satisfies*

$$\|l_k\|_{p,\rho} \le 2 \sum_{i=1}^{k-1} \frac{\gamma^i}{1-\gamma} \left(\mathcal{C}_q^{i,i+1}\right)^{\frac{1}{p}} \|\epsilon_{k-i}\|_{pq',\mu} + \sum_{i=0}^{k-1} \frac{\gamma^i}{1-\gamma} \left(\mathcal{C}_q^{i,i+1}\right)^{\frac{1}{p}} \|\epsilon'_{k-i}\|_{pq',\mu} + g(k).$$

Proof. We define $\mathcal{I}$, (g_i) and $(\mathcal{J}_i)$ as in the proof of Theorem 1. We then make as many groups as terms, i.e., for each $n \in \{1, 2, \dots, 2k-1\}$, we define $\mathcal{I}_n = \{n\}$. The result follows by application of Lemma 3. □

E. Proof of Lemma 4

The proof of this lemma is similar to the proof of Theorem 1 in Lazaric et al. (2010). Before stating the proof, we report the following two lemmas that are used in the proof.

Lemma 6. *Let Π be a policy space with finite VC-dimension $h = VC(\Pi) < \infty$ and N be the number of states in the rollout set $\mathcal{D}_{k-1}$ drawn i.i.d. from the state distribution μ. Then we have*

$$\mathbb{P}_{\mathcal{D}_{k-1}}\left[\sup_{\pi\in\Pi}\left|\mathcal{L}^{\Pi}_{k-1}(\widehat{\mu};\pi) - \mathcal{L}^{\Pi}_{k-1}(\mu;\pi)\right| > \epsilon\right] \leq \delta\ ,$$

with $\epsilon = 16 Q_{\max}\sqrt{\frac{2}{N}\left(h\log\frac{eN}{h} + \log\frac{8}{\delta}\right)}$.

Proof. This is a restatement of Lemma 1 in Lazaric et al. (2010). □

Lemma 7. *Let Π be a policy space with finite VC-dimension $h = VC(\Pi) < \infty$ and $s^{(1)}, \dots, s^{(N)}$ be an arbitrary sequence of states. At each state we simulate M independent rollouts of the form , then we have*

$$\mathbb{P}\left[\sup_{\pi\in\Pi}\left|\frac{1}{N}\sum_{i=1}^{N}\frac{1}{M}\sum_{j=1}^{M}R^{j}_{k-1}\big(s^{(i,j)},\pi(s^{(i,j)})\big) - \frac{1}{N}\sum_{i=1}^{N}Q_{k-1}\big(s^{(i,j)},\pi(s^{(i,j)})\big)\right| > \epsilon\right] \leq \delta\ ,$$

with $\epsilon = 8 Q_{\max}\sqrt{\frac{2}{MN}\left(h\log\frac{eMN}{h} + \log\frac{8}{\delta}\right)}$.

Proof. The proof is similar to the one for Lemma 6. □

Proof. **(Lemma 4)** Let $a^*(\cdot) = \operatorname{argmax}_{a\in\mathcal{A}} Q_{k-1}(\cdot, a)$ be the greedy action. To simplify the notation, we remove the dependency of a^* on states and use a^* instead of $a^*(x_i)$ in the following. We prove the following series of inequalities:

$$\begin{aligned}
\mathcal{L}^{\Pi}_{k-1}(\mu;\pi_k) &\overset{(a)}{\leq} \mathcal{L}^{\Pi}_{k-1}(\widehat{\mu};\pi_k) + \epsilon'_1 \qquad \text{w.p. } 1-\delta' \\
&= \frac{1}{N}\sum_{i=1}^{N}\Big[Q_{k-1}(x_i,a^*) - Q_{k-1}\big(x_i,\pi_k(x_i)\big)\Big] + \epsilon'_1 \\
&\overset{(b)}{\leq} \frac{1}{N}\sum_{i=1}^{N}\Big[Q_{k-1}(x_i,a^*) - \widehat{Q}_{k-1}\big(x_i,\pi_k(x_i)\big)\Big] + \epsilon'_1 + \epsilon'_2 \qquad \text{w.p. } 1-2\delta' \\
&\overset{(c)}{\leq} \frac{1}{N}\sum_{i=1}^{N}\Big[Q_{k-1}(x_i,a^*) - \widehat{Q}_{k-1}\big(x_i,\pi^*(x_i)\big)\Big] + \epsilon'_1 + \epsilon'_2 \\
&\leq \frac{1}{N}\sum_{i=1}^{N}\Big[Q_{k-1}(x_i,a^*) - Q_{k-1}\big(x_i,\pi^*(x_i)\big)\Big] + \epsilon'_1 + 2\epsilon'_2 \qquad \text{w.p. } 1-3\delta' \\
&= \mathcal{L}^{\Pi}_{k-1}(\widehat{\mu};\pi^*) + \epsilon'_1 + 2\epsilon'_2 \leq \mathcal{L}^{\Pi}_{k-1}(\mu;\pi^*) + 2(\epsilon'_1+\epsilon'_2) \qquad \text{w.p. } 1-4\delta' \\
&= \inf_{\pi\in\Pi}\mathcal{L}^{\Pi}_{k-1}(\mu;\pi) + 2(\epsilon'_1+\epsilon'_2).
\end{aligned}$$

The statement of the theorem is obtained by $\delta' = \delta/4$.

(a) This follows from Lemma 6.
(b) Here we introduce the estimated action-value function $\widehat{Q}_{k-1}$ by bounding

$$\sup_{\pi\in\Pi}\left[\frac{1}{N}\sum_{i=1}^{N}\widehat{Q}_{k-1}\big(s^{(i)},\pi(s^{(i)})\big) - \frac{1}{N}\sum_{i=1}^{N}Q_{k-1}\big(s^{(i)},\pi(s^{(i)})\big)\right]$$

using Lemma 7.

(c) From the definition of π_k in CBMPI, we have

$$\pi_k = \underset{\pi\in\Pi}{\operatorname{argmin}}\, \widehat{\mathcal{L}}^{\Pi}_{k-1}(\widehat{\mu};\pi) = \underset{\pi\in\Pi}{\operatorname{argmax}}\, \frac{1}{N}\sum_{i=1}^{N} \widehat{Q}_{k-1}\big(s^{(i)}, \pi(s^{(i)})\big),$$

thus, $-1/N\sum_{i=1}^{N} \widehat{Q}_{k-1}\big(s^{(i)}, \pi_k(s^{(i)})\big)$ can be maximized by replacing π_k with any other policy, particularly with

$$\pi^* = \underset{\pi\in\Pi}{\operatorname{argmin}} \int_{\mathcal{S}} \Big(\max_{a\in\mathcal{A}} Q_{k-1}(s,a) - Q_{k-1}\big(s,\pi(s)\big)\Big)\,\mu(ds).$$

□

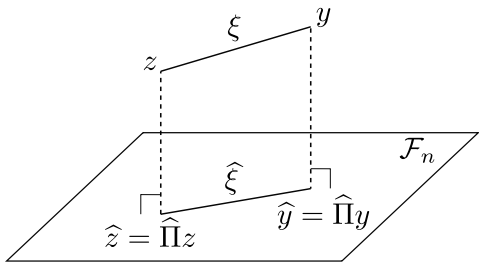

Figure 2. The vectors used in the proof.

F. Proof of Lemma 5

Let us define two n-dimensional vectors $z = \Big(\big[(T_{\pi_{k-1}})^m v_{k-2}\big](s^{(1)}), \ldots, \big[(T_{\pi_{k-1}})^m v_{k-2}\big](s^{(n)})\Big)^\top$ and $y = \big(\widehat{v}_{k-1}(s^{(1)}), \ldots, \widehat{v}_{k-1}(s^{(n)})\big)^\top$ and their orthogonal projections onto the vector space $\mathcal{F}_n$ as $\widehat{z} = \widehat{\Pi} z$ and $\widehat{y} = \widehat{\Pi} y = \big(\widetilde{v}_{k-1}(s^{(1)}), \ldots, \widetilde{v}_{k-1}(s^{(n)})\big)^\top$, where $\widetilde{v}_{k-1}$ is the result of linear regression and its truncation (by $V_{\max}$) is v_{k-1}, i.e., $v_{k-1} = \mathbb{T}(\widetilde{v}_{k-1})$ (see Figure 2). What we are interested is to find a bound on the regression error $||z - \widehat{y}||$ (the difference between the target function z and the result of the regression $\widehat{y}$). We may decompose this error as

$$||z - \widehat{y}||_n \leq ||\widehat{z} - \widehat{y}||_n + ||z - \widehat{z}||_n = ||\widehat{\xi}||_n + ||z - \widehat{z}||_n, \tag{39}$$

where $\widehat{\xi} = \widehat{z} - \widehat{y}$ is the projected noise (estimation error) $\widehat{\xi} = \widehat{\Pi}\xi$, with the noise vector $\xi = z - y$ defined as $\xi_i = \big[(T_{\pi_{k-1}})^m v_{k-2}\big](s^{(i)}) - \widehat{v}_{k-1}(s^{(i)})$. It is easy to see that noise is zero mean, i.e., $\mathbb{E}[\xi_i] = 0$ and is bounded by $2V_{\max}$, i.e., $|\xi_i| \leq 2V_{\max}$. We may write the estimation error as

$$||\widehat{z} - \widehat{y}||_n^2 = ||\widehat{\xi}||_n^2 = \langle \widehat{\xi}, \widehat{\xi} \rangle = \langle \xi, \widehat{\xi} \rangle,$$

where the last equality follows from the fact that $\widehat{\xi}$ is the orthogonal projection of ξ. Since $\widehat{\xi} \in \mathcal{F}_n$, let $f_\alpha \in \mathcal{F}$ be any function whose values at $\{s^{(i)}\}_{i=1}^n$ equals to $\{\xi_i\}_{i=1}^n$. By application of a variation of Pollard's inequality (Györfi et al., 2002), we obtain

$$\langle \xi, \widehat{\xi} \rangle = \frac{1}{n} \sum_{i=1}^n \xi_i f_\alpha(s^{(i)}) \leq 4V_{\max} ||\widehat{\xi}||_n \sqrt{\frac{2}{n} \log\left(\frac{3(9e^2 n)^{d+1}}{\delta'}\right)},$$

with probability at least $1 - \delta'$. Thus, we have

$$||\widehat{z} - \widehat{y}||_n = ||\widehat{\xi}||_n \leq 4V_{\max} \sqrt{\frac{2}{n} \log\left(\frac{3(9e^2 n)^{d+1}}{\delta'}\right)}. \tag{40}$$

From Eqs. 39 and 40, we have

$$||(T_{\pi_{k-1}})^m v_{k-2} - \widetilde{v}_{k-1}||_{\widehat{\mu}} \leq ||(T_{\pi_{k-1}})^m v_{k-2} - \widehat{\Pi}(T_{\pi_{k-1}})^m v_{k-2}||_{\widehat{\mu}} + 4V_{\max} \sqrt{\frac{2}{n} \log\left(\frac{3(9e^2 n)^{d+1}}{\delta'}\right)}, \tag{41}$$

where $\widehat{\mu}$ is the empirical norm induced from the n i.i.d. samples from μ.

Now in order to obtain a random design bound, we first define $f_{\widehat{\alpha}_*} \in \mathcal{F}$ as $f_{\widehat{\alpha}_*}(s^{(i)}) = \big[\widehat{\Pi}(T_{\pi_{k-1}})^m v_{k-2}\big](s^{(i)})$, and then define $f_{\alpha_*} = \Pi(T_{\pi_{k-1}})^m v_{k-2}$ that is the best approximation (w.r.t. μ) of the target function $(T_{\pi_{k-1}})^m v_{k-2}$ in $\mathcal{F}$. Since $f_{\widehat{\alpha}_*}$ is the minimizer of the empirical loss, any function in $\mathcal{F}$ different than $f_{\widehat{\alpha}_*}$ has a bigger empirical loss, thus we have

$$\begin{aligned} ||f_{\widehat{\alpha}_*} - (T_{\pi_{k-1}})^m v_{k-2}||_{\widehat{\mu}} \leq ||f_{\alpha_*} - (T_{\pi_{k-1}})^m v_{k-2}||_{\widehat{\mu}} \leq 2||f_{\alpha_*} - (T_{\pi_{k-1}})^m v_{k-2}||_{\mu} \\ + 12\Big(V_{\max} + ||\alpha_*||_2 \sup_x ||\phi(x)||_2\Big) \sqrt{\frac{2}{n} \log \frac{3}{\delta'}}, \end{aligned} \tag{42}$$

with probability at least $1-\delta'$, where the second inequality is the application of a variation of Theorem 11.2 in the book by Györfi et al., (2002) with $||f_{\alpha_*} - (T_{\pi_{k-1}})^m v_{k-2}||_\infty \leq V_{\max} + ||\alpha^*||_2 \ \sup_x ||\phi(x)||_2$. Similarly, we can write the left-hand-side of Equation 41 as

$$2||(T_{\pi_{k-1}})^m v_{k-2} - \widetilde{v}_{k-1}||_{\widehat{\mu}} \geq 2||(T_{\pi_{k-1}})^m v_{k-2} - \mathbb{T}(\widetilde{v}_{k-1})||_{\widehat{\mu}} \geq ||(T_{\pi_{k-1}})^m v_{k-2} - \mathbb{T}(\widetilde{v}_{k-1})||_{\mu} - 24V_{\max}\sqrt{\frac{2}{n}\Lambda(n,d,\delta')}, \tag{43}$$

with probability at least $1-\delta'$, where $\Lambda(n,d,\delta') = 2(d+1)\log n + \log\frac{e}{\delta'} + \log\left(9(12e)^{2(d+1)}\right)$. Putting together Equations 41, 42, and 43 and using the fact that $\mathbb{T}(\widetilde{v}_{k-1}) = v_{k-1}$, we obtain

$$||\eta_{k-1}||_{2,\mu} = ||(T_{\pi_{k-1}})^m v_{k-2} - v_{k-1}||_\mu \leq 2\Bigg(2||(T_{\pi_{k-1}})^m v_{k-2} - f_{\alpha_*}||_\mu + 12\Big(V_{\max} + ||\alpha_*||_2 \ \sup_x ||\phi(x)||_2\Big)\sqrt{\frac{2}{n}\log\frac{3}{\delta'}}$$
$$+ 4V_{\max}\sqrt{\frac{2}{n}\log\left(\frac{3(9e^2n)^{d+1}}{\delta'}\right)}\Bigg) + 24V_{\max}\sqrt{\frac{2}{n}\Lambda(n,d,\delta')}.$$

The result follows by setting $\delta = 3\delta'$ and some simplification.

G. Proof of Theorem 2

As mentioned in Sec. 5, the evaluation step error of CBMPI, η, and the one used in the propagation analysis of Lemma 2, ϵ, are related as $\epsilon_k = (\gamma P_{\pi_k})^m \eta_{k-1}$. Therefore, using the fact that $\eta_0 = 0$, we may rewrite the bound of Lemma 2 for CBMPI as follows:

$$\begin{aligned} l_k &\le 2\sum_{i=1}^{k-1}\sum_{j=i}^{\infty}\Gamma^{j+m}|\eta_{k-i-1}| + \sum_{i=0}^{k-1}\sum_{j=i}^{\infty}\Gamma^{j}|\epsilon'_{k-i}| + h(k) \\ &= 2\sum_{i=2}^{k-1}\sum_{j=m+i-1}^{\infty}\Gamma^{j}|\eta_{k-i}| + \sum_{i=0}^{k-1}\sum_{j=i}^{\infty}\Gamma^{j}|\epsilon'_{k-i}| + h(k). \end{aligned} \tag{44}$$

Using Lemma 3, we may turn the component-wise bound of Eq. 44 to the following L_p-norm bound, which is a more general form of Theorem 2.

Theorem 4. *Let ρ and μ be distributions over states. Then for all integers p, q, and q' such that $\frac{1}{q} + \frac{1}{q'} = 1$, and after k iterations, the loss of AMPI satisfies*

$$\|l_k\|_{p,\rho} \le \frac{2\gamma^m(\gamma - \gamma^{k-1})\left(\mathcal{C}_q^{2,k,m}\right)^{\frac{1}{p}}}{(1-\gamma)^2} \sup_{1\le j\le k-2}\|\eta_j\|_{pq',\mu} + \frac{(1-\gamma^k)\left(\mathcal{C}_q^{1,k,0}\right)^{\frac{1}{p}}}{(1-\gamma)^2}\sup_{1\le j\le k}\|\epsilon'_j\|_{pq',\mu} + g(k),$$

where for all q, l, k, and d, the concentrability coefficients $\mathcal{C}_q^{l,k,d}$ are defined as

$$\mathcal{C}_q^{l,k,d} \triangleq \frac{(1-\gamma)^2}{\gamma^{l+d} - \gamma^{k+d}}\sum_{i=l}^{k-1}\sum_{j=i+d}^{\infty}\gamma^{j+d}c_q(j+d),$$

with $c_q(j)$ given by Eq. 14, and $g(k)$ is defined as $g(k) \triangleq \frac{2\gamma^k}{1-\gamma}\left(\mathcal{C}_q^{k,k+1}\right)^{\frac{1}{p}}\min\left(\|d_0\|_{pq',\mu}, \|b_0\|_{pq',\mu}\right)$.

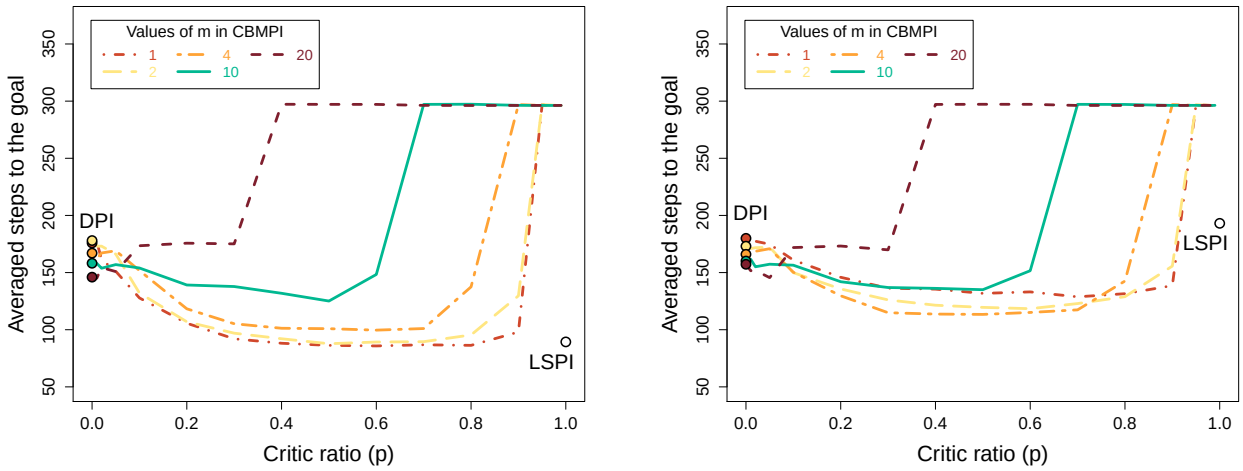

Figure 3. Performance of the learned policies in mountain car with two different 2×2 RBF grids, the one with good approximation of the value function is on the left and the one with poor performance in approximating the value function is on the right. The total budget B is set to 200. The objective is to minimize the number of steps to the goal.

H. Experimental Results

In this section, we report the empirical evaluation of CBMPI and compare it to DPI and LSPI. In the experiments, we show that CBMPI, by combining policy and value function approximation, can improve over DPI and LSPI. In these experiments, we are using the same setting as in Gabillon et al. (2011) to facilitate the comparison.

H.1. Setting

We consider the mountain car (MC) problem with its standard formulation in which the action noise is bounded in $[-1, 1]$ and $\gamma = 0.99$. The value function is approximated using a linear space spanned by a set of radial basis functions (RBFs) evenly distributed over the state space.

Each CBMPI-based algorithm is run with the same fixed budget B per iteration. CBMPI splits the budget into a rollout budget $B_R = B(1-p)$ used to build the training set of the greedy step and a critic budget $B_C = Bp$ used to build the training set of the evaluation step , where $p \in (0, 1)$ is the critic ratio. The rollout budget is divided into M rollouts of length m for each action in $\mathcal{A}$ and each state in the rollout set $\mathcal{D}'$, i.e., $B_R = mMN|\mathcal{A}|$. The critic budget is divided into one rollout of length m for each action in $\mathcal{A}$ and each state in the rollout set $\mathcal{D}$, i.e., $B_C = mn|\mathcal{A}|$.

In Fig. 3, we report the performance of DPI, CBMPI, and LSPI. In MC, the performance is evaluated as the number of steps-to-go with a maximum of 300. The results are averaged over 1000 runs. We report the performance of DPI and LSPI at $p = 0$ and $p = 1$, respectively. DPI can be seen as a special case of CBMPI where $p = 0$. We tested the performance of DPI and CBMPI on a wide range of parameters (m, M, N, n) but we only report their performance for the best choice of M ($M = 1$ was the best choice in all the experiments) and different values of m.

H.2. Experiments

As discussed in Remark 4, the parameter m balances between the error in evaluating the value function and the error in evaluating the policy. The value function approximation error tends to zero for large values of m. Although this would suggest to have large values for m, the size of the rollout sets would correspondingly decrease as $N = O(B/m)$ and $n = O(B/m)$, thus decreasing the accuracy of both the regression and classification problems. This leads to a trade-off between long rollouts and the number of states in the rollout sets. The solution to this trade-off strictly depends on the capacity of the value function space $\mathcal{F}$. A rich value function space would lead to solve the trade-off for small values of m. On the other hand, when the value function space is poor, or as in the DPI case, m should be selected in a way to guarantee a sufficient number of informative rollouts, and

at the same time, a large enough rollout sets.

Figure 3 shows the learning results in MC with budget $B = 200$. On the left panel, the function space is rich enough to approximate v^*. Therefore LSPI has almost optimal results (about 80 steps to reach the goal). On the other hand, DPI achieves a poor performance of about 150 steps, which is obtained by setting $m = 12$ and $N = 5$. We also report the performance of CBMPI for different values of m and p. When p is large enough, the value function approximation becomes accurate enough so that the best solution is to have $m = 1$. This both corresponds to rollouts built almost entirely on the basis of the approximated value function and to a large number of states in the training set N. For $m = 1$ and $p \approx 0.8$, CBMPI achieves a slightly better performance than LSPI.

In the next experiment, we show that CBMPI is able to outperform both DPI and LSPI when $\mathcal{F}$ has a lower accuracy. The results are reported on the right panel of Figure 3. The performance of LSPI now worsens to 190 steps. Simultaneously one can notice $m = 1$ is no longer the best choice for CBMPI. Indeed in the case where $m = 1$, CBMPI becomes an approximated version of the value iteration algorithm relying on a function space not rich enough to approximate $v*$. Notice that relying on this space is still better than setting the value function to zero which is the case in DPI. Therefore, we notice an improvement of CBMPI over DPI for $m = 4$ which trade-off between the estimates of the value function and the rewards collected by the rollouts. Combining those two, CBMPI also improves upon LSPI.