# Habilitation à diriger des recherches
# Université Lille 1 - Sciences et Technologies

préparée au sein du centre de recherche INRIA Lille - Nord Europe

**Specialité : Mathématiques**

## Mohammad GHAVAMZADEH

# COMPLEXITÉ D'ÉCHANTILLONNAGE POUR LA PRISE DE DÉCISION SÉQUENTIELLE

---

# SAMPLE COMPLEXITY IN SEQUENTIAL DECISION-MAKING

---

Habilitation soutenue à Villneuve d'Ascq le 11 juin 2014 devant le jury composé de

| | | |
|---|---|---|
| Pr. Peter AUER | University of Leoben | Rapporteur |
| Pr. Mohamed DAOUDI | Université Lille 1 | Président |
| Pr. Michael LITTMAN | Brown University | Rapporteur |
| Pr. Shie MANNOR | Technion - Israel Institute of Technology | Examinateur |
| Pr. Philippe PREUX | Université Lille 3 | Examinateur |
| Pr. Csaba SZEPESVÁRI | University of Alberta | Examinateur |
| Pr. Benjamin VAN ROY | Stanford University | Rapporteur |

# Résumé

De nombreux problèmes intéressants de prise de décision séquentielle peuvent être formulés comme des problèmes d'apprentissage par renforcement. En apprentissage par renforcement, un agent interagit avec un environnement dynamique, stochastique et qu'il ne connaît que partiellement, dans le but de trouver une stratégie de prise d'actions, ou *politique*, qui maximise une certaine mesure de performance à long terme. Les algorithmes de programmation dynamique sont les outils les plus puissants pour résoudre les problèmes d'apprentissage par renforcement, c'est à dire pour trouver la politique optimale. Cependant, pour ces algorithmes, la découverte du comportement décisionnel optimal n'est garantie que si l'environnement (à savoir, la dynamique de l'état du système et les récompenses) est connu de manière complète et que les espaces d'état et d'action ne sont pas de trop grandes tailles. Lorsque l'une de ces conditions se trouve violée (par exemple si l'unique information disponible sur l'environnement prend la forme d'échantillons de transitions et de récompenses), des algorithmes d'approximation sont requis, et dès lors, les méthodes de programmation dynamique se convertissent en méthodes de programmation dynamique approchée et en algorithmes d'apprentissage par renforcement.[1]

La théorie de l'apprentissage statistique est fondamentale pour l'étude

[1] Le terme apprentissage par renforcement est plus fréquemment utilisé dans la communauté d'IA et d'apprentissage automatique, alors que programmation dynamique approchée est plus commun en recherche opérationnelle. Ici nous utiliserons ces termes de manière interchangeable.

des propriétés statistiques des algorithmes développés en apprentissage automatique. En particulier, apprentissage statistique décrit l'interaction entre le processus générant les échantillons et l'espace d'hypothèse utilisé par l'algorithme d'apprentissage, et établit à quelles conditions et dans quelles mesures les problèmes de régression et de classification peuvent être résolus. Ces résultats ont aussi montré leur utilité pour dimensionner les problèmes d'apprentissage automatique (nombre d'échantillons, complexité de l'espace d'hypothèse) et pour ajuster les paramètres des algorithmes (par exemple le paramètre de régularisation des méthodes de régularisation).

L'objet principal de ce travail est d'employer les outils de l'apprentissage statistique afin d'étudier les performances des algorithmes d'apprentissage par renforcement hors ligne et de programmation dynamique approchée[2] pour aboutir à des bornes en échantillons finis sur la perte en performance (par rapport à la politique optimale) de la politique apprise par ces algorithmes. Un tel objectif demande de combiner efficacement des outils de l'apprentissage statistique avec les algorithmes de programmation dynamique approchée, et de montrer comment l'erreur se propage d'itération en itération chez ces algorithmes itératifs. Nous considérons différents types d'algorithmes de programmation dynamique approchée : basés soit sur une régression, une classification ou une méthode de point fixe, et, pour chacun, nous soulignons les principaux défis que posent leurs analyses en échantillons finis.

[2] En apprentissage par renforcement hors ligne (à l'inverse des versions en ligne ou incrémentales de ces algorithmes), une politique d'échantillonnage est utilisée pour construire un ensemble d'apprentissage pour l'algorithme d'apprentissage.

# Abstract

Many interesting sequential decision-making tasks can be formulated as reinforcement learning (RL) problems. In a RL problem, an agent interacts with a dynamic, stochastic, and incompletely known environment, with the goal of finding an action-selection strategy, or *policy*, to maximize some measure of its long-term performance. Dynamic programming (DP) algorithms are the most powerful tools to solve a RL problem, i.e., to find an optimal policy. However, these algorithms guarantee to find an optimal policy only if the environment (i.e., the dynamics and the rewards) is completely known and the size of the state and action spaces are not too large. When one of these conditions is violated (e.g., the only information about the environment is of the form of samples of transitions and rewards), approximate algorithms are needed, and thus, DP methods turn to approximate dynamic programming (ADP) and RL algorithms.[3] In this case, the convergence and performance guarantees of the standard DP algorithms are no longer valid, and the main theoretical challenge is to study the performance of ADP and RL algorithms.

Statistical learning theory (SLT) has been fundamental in understanding the statistical properties of the algorithms developed in machine learning. In particular, SLT has explained the interaction between the process generating the samples and the hypothesis space used by the learning algorithm, and shown when and how-well classification and regression problems can be

[3] While the term RL is more often used in the AI and machine learning community, ADP is more common in the field of operations research. Here we use them interchangeably.

solved. These results also proved to be particularly useful in dimensioning the machine learning problems (i.e., number of samples, complexity of the hypothesis space) and tuning the parameters of the algorithms (e.g., the regularizer in regularized methods).

The main objective of this work is to use the tools from SLT to study the performance of batch RL and ADP algorithms[4] with the objective of deriving finite-sample bounds on the performance loss (w.r.t. the optimal policy) of the policy learned by these methods. Such an objective requires to effectively combine SLT tools with the ADP algorithms, and to show how the error is propagated through the iterations of these iterative algorithms. We consider several different types of ADP algorithms: regression-based, classification-based and fixed-point, and in each case highlight the main challenges in their finite-sample performance analysis.

[4] In batch RL (v.s. incremental or online version of these algorithms), a sampling policy is used to build a training set for the learning algorithm.

# Contents

# Publications this manuscript is based on

## Journal Papers

[MGH1] A. Lazaric, M. Ghavamzadeh, and R. Munos. Finite-Sample Analysis of Least-Squares Policy Iteration. *Journal of Machine Learning Research* (JMLR), 13:3041-3074, 2012..

[MGH2] M. Ghavamzadeh, A. Lazaric, and R. Munos. Analysis of Classification-based Policy Iteration Algorithms. *submitted to the Journal of Machine Learning Research* (JMLR), 2013.

[MGH3] B. Scherrer, M. Ghavamzadeh, V. Gabillon, B. Lesner, and M. Geist. Approximate Modified Policy Iteration. *submitted to the Journal of Machine Learning Research* (JMLR), 2013.

[MGH4] A. Farahmand, M. Ghavamzadeh, Cs. Szepesvári, and S. Mannor. Regularized Policy Iteration. *submitted to the Journal of Machine Learning Research* (JMLR), 2013.

[MGH5] A. Farahmand, D. Precup, M. Ghavamzadeh, and A. Barreto. On Classification-based Approximate Policy Iteration. *submitted to the IEEE Transactions on Automatic Control* (TAC), 2013.

## Conference Papers

[MGH6] V. Gabillon, M. Ghavamzadeh, and B. Scherrer. Approximate Dynamic Programming Finally Performs Well in the Game of Tetris. *Proceedings of the Advances in Neural Information Processing Systems 26*, pp. 1754-1762, 2013.

[MGH7] B. Ávila Pires, M. Ghavamzadeh, and Cs. Szepesvári. Cost-sensitive Multiclass Classification Risk Bounds. *Proceedings of the Thirtieth International Conference on Machine Learning*, pp. 1391-1399, 2013.

[MGH8] B. Scherrer, M. Ghavamzadeh, V. Gabillon, and M. Geist. Approximate Modified Policy Iteration. *Proceedings of the Twenty-Ninth International Conference on Machine Learning*, pp. 1207-1214, 2012.

[MGH9] M. Geist, B. Scherrer, A. Lazaric, and M. Ghavamzadeh. A Dantzig Selector Approach to Temporal Difference Learning. *Proceedings of the Twenty-Ninth International Conference on Machine Learning*, pp. 1399-1406, 2012.

[MGH10] M. Ghavamzadeh and A. Lazaric. Conservative and Greedy Approaches to Classification-based Policy Iteration. *Proceedings of the Twenty-Sixth Conference on Artificial Intelligence*, pp. 914-920, 2012.

[MGH11] V. Gabillon, M. Ghavamzadeh, and A. Lazaric. Best Arm Identification: A Unified Approach to Fixed Budget and Fixed Confidence. *Proceedings of the Advances in Neural Information Processing Systems 25*, pp. 3221-3229, 2012.

[MGH12] M. Ghavamzadeh, A. Lazaric, R. Munos, and M. Hoffman. Finite-Sample Analysis of Lasso-TD. *Proceedings of the Twenty-Eighth International Conference on Machine Learning*, pp. 1177-1184, 2011.

[MGH13] V. Gabillon, A. Lazaric, M. Ghavamzadeh, and B. Scherrer. Classification-based Policy Iteration with a Critic. *Proceedings of the Twenty-Eighth International Conference on Machine Learning*, pp. 1049-1056, 2011.

[MGH14] V. Gabillon, M. Ghavamzadeh, A. Lazaric, and S. Bubeck. Multi-Bandit Best Arm Identification. *Proceedings of the Advances in Neural Information Processing Systems 24*, pp. 2222-2230, 2011.

[MGH15] M. Ghavamzadeh, A. Lazaric, O. Maillard, and R. Munos. LSTD with Random Projections. *Proceedings of the Advances in Neural Information Processing Systems 23*, pp.721-729, 2010.

[MGH16] O. Maillard, R. Munos, A. Lazaric, and M. Ghavamzadeh. Finite-Sample Analysis of Bellman Residual Minimization. *Proceedings of the Second Asian Conference on Machine Learning*, pp.299-314, 2010.

[MGH17] A. Lazaric, M. Ghavamzadeh, and R. Munos. Analysis of a Classification-based Policy Iteration Algorithm. *Proceedings of the Twenty-Seventh International Conference on Machine Learning*, pp. 607-614, 2010.

[MGH18] A. Lazaric, M. Ghavamzadeh, and R. Munos. Finite-Sample Analysis of LSTD. *Proceedings of the Twenty-Seventh International Conference on Machine Learning*, pp. 615-622, 2010.

[MGH19] A. Farahmand, M. Ghavamzadeh, Cs. Szepesvári, and S. Mannor. Regularized Fitted Q-iteration for Planning in Continuous-Space Markovian Decision Problems. *Proceedings of the American Control Conference*, pp. 725-730, 2009.

[MGH20] A. Farahmand, M. Ghavamzadeh, Cs. Szepesvári, and S. Mannor. Regularized Policy Iteration. *Proceedings of the Advances in Neural Information Processing Systems 21*, pp. 441-448, 2008.

# Acknowledgements

Thanks for reading.

# Chapter 1

# Introduction

## 1.1 The Reinforcement Learning Problem

Many interesting sequential decision-making tasks such as moving around in the physical world (e.g., driving or navigation), playing a game, retrieving information over the web, medical diagnosis and treatment, maximizing the throughput of a factory, optimizing the performance of a rescue team, and many more can be formulated as reinforcement learning (RL) problems. In a RL problem, an agent interacts with a dynamic, stochastic, and incompletely known environment, with the goal of finding an action-selection strategy, or *policy*, to maximize some measure of its long-term performance. The agent's interaction with the environment is often modeled as a Markov decision process (MDP) or in case the state of the agent is not always fully observable, as a partially observable Markov decision process (POMDP). In this work, we focus on the fully observable case, and thus, use a MDP to model this interaction.

A (discounted) MDP is a tuple $\mathcal{M} = \langle \mathcal{X}, \mathcal{A}, r, P, \gamma \rangle$ where the state space $\mathcal{X}$ is a bounded closed subset of $\mathbb{R}^d$, $\mathcal{A}$ is the action space,[1] the reward function $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is uniformly bounded by $R_{\max}$, the transition kernel

[1] We assume that the action space is finite, i.e., $|\mathcal{A}| < \infty$ in this work.

$P$ is such that for all $x \in \mathcal{X}$ and $a \in \mathcal{A}$, $P(\cdot|x, a)$ is a distribution over $\mathcal{X}$, and $\gamma \in (0, 1]$ is a discount factor. The main objective of the agent is to find a good or an optimal policy $\pi$, which is a mapping from states to actions $\pi : \mathcal{X} \to \mathcal{A}$ (*deterministic policy*) or a mapping from states to a distribution over actions $\pi : \mathcal{X} \times \mathcal{A} \to [0, 1]$ (*stochastic policy*). By optimal here we mean a policy that maximizes the agent's long-term performance. Finding a good or an optimal policy requires searching in the policy space and to be able to compare them. However, a policy is a mapping from states to actions, which brings up this important question that how can we compare two policies (mappings) and search in the policy space? The answer to this question is: this comparison is done using a quantity called *value function.* For a policy $\pi$, its value function, $V^\pi$, is a function from the states to real numbers, $V^\pi : \mathcal{X} \to \mathbb{R}$, that at each state $x$ is defined as the expected sum of (discounted) rewards of starting at that state and then following the policy $\pi$, i.e.,

$$V^\pi(x) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r\big(X_t, \pi(X_t)\big)|X_0 = x\right]. \tag{1.1}$$

Now using this quantity we can compare two policies and say policy $\pi_1$ is better than or equal to policy $\pi_2$ if and only if its value function at every state is larger than or equal to the value function of policy $\pi_2$, i.e., $\forall x \in \mathcal{X},\ V^{\pi_1}(x) \geq V^{\pi_2}(x)$. Similarly for any policy $\pi$, we can define another quantity, called *action-value function*, $Q^\pi$, which is a mapping from the state-action pairs to real numbers, $Q^\pi : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$, and at each state-action pair $(x, a)$ is defined as the expected sum of (discounted) rewards of starting at state $x$, taking action $a$, and then following the policy $\pi$, i.e.,

$$Q^\pi(x, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r\big(X_t, A_t\big)|X_0 = x,\ A_0 = a,\ \forall t \geq 1, A_t = \pi(X_t)\right]. \tag{1.2}$$

It has been shown (see e.g., Bertsekas and Tsitsiklis 1996) that the value function of a policy $\pi$ is the unique fixed-point of the *Bellman operator* $\mathcal{T}^\pi$, i.e., $\forall x \in \mathcal{X},\ V^\pi(x) = (\mathcal{T}^\pi V^\pi)(x)$, defined as

$$\forall x \in \mathcal{X},\ \forall f : \mathcal{X} \to \mathbb{R}, \quad (\mathcal{T}^\pi f)(x) = r\big(x, \pi(x)\big) + \gamma \sum_{x' \in \mathcal{X}} P\big(x'|x, \pi(x)\big) f(x'). \tag{1.3}$$

From Equations 1.1–1.3, it is easy to see that

$$\forall x \in \mathcal{X},\ a \in \mathcal{A}, \qquad Q^\pi(x,a) = r(x,a) + \gamma \sum_{x' \in \mathcal{X}} P(x'|x,a) V^\pi(x').$$

The *optimal value function*, $V^*$, is the value function of a policy that has the maximum value at every state, i.e., $\forall x \in \mathcal{X},\ V^*(x) = \sup_\pi V^\pi(x)$.[2] Now a policy $\pi^*$ is called *optimal* if its value is equal to the optimal value function at every state, i.e., $\forall x \in \mathcal{X},\ V^{\pi^*}(x) = V^*(x)$. The main message here is that while we may have several optimal policies, the optimal value function is always unique.

Similar to the value function of a policy, it has been shown (see e.g., Bertsekas and Tsitsiklis 1996) that the optimal value function $V^*$ is the unique fixed-point of the *Bellman optimality operator* $\mathcal{T}$, i.e., $\forall x \in \mathcal{X},\ V^*(x) = (\mathcal{T} V^*)(x)$, defined as

$$\forall x \in \mathcal{X},\ \forall f : \mathcal{X} \to \mathbb{R}, \quad (\mathcal{T} f)(x) = \max_{a \in \mathcal{A}} \left[ r(x,a) + \gamma \sum_{x' \in \mathcal{X}} P(x'|x,a) f(x') \right]. \tag{1.4}$$

Both Bellman and Bellman optimality operators have two the following important properties that are fundamental to the convergence proofs of most RL algorithms:

- **Monotonicity:** If $V_1 \le V_2$ component-wise then

$$\mathcal{T}^\pi V_1 \le \mathcal{T}^\pi V_2 \qquad \text{and} \qquad \mathcal{T} V_1 \le \mathcal{T} V_2 \,.$$

[2] Similarly we can define the optimal action-value function $Q^*$ as $\forall x \in \mathcal{X},\ a \in \mathcal{A},\ Q^*(x,a) = \sup_\pi Q^\pi(x,a)$.

- **Max-Norm Contraction:** For every pair of value functions $V_1$ and $V_2$, we have

$$\begin{aligned} ||\mathcal{T}^{\pi} V_1 - \mathcal{T}^{\pi} V_2||_\infty &\leq \gamma ||V_1 - V_2||_\infty \ , \\ ||\mathcal{T} V_1 - \mathcal{T} V_2||_\infty &\leq \gamma ||V_1 - V_2||_\infty \ . \end{aligned}$$

## 1.2 Dynamic Programming

Most of the solutions to the RL problem are based on one of the two celebrated dynamic programming (DP) algorithms: *value iteration* and *policy iteration* [Bertsekas and Tsitsiklis, 1996, Sutton and Barto, 1998, Szepesvári, 2010].

**Value Iteration (VI)** is an iterative algorithm that starts with an arbitrary initial value function $V_0$ and at each iteration $k$ of the algorithm generates a new value function $V_k$ from the current value function $V_{k-1}$ as $V_k = \mathcal{T} V_{k-1}$. Note that the computational cost at each iteration of VI is $O(|\mathcal{X}|^2|\mathcal{A}|)$ and its number of iterations depends of our desired level of accuracy. So, we can consider VI as a polynomial algorithm in the number of states and actions.

It is easy to show that as the number of iterations approaches infinity, $k \to \infty$, the value function generated by the value iteration algorithm converges to the optimal value function.

**Lemma 1** *Let $V_k$ be the value function generated at the iteration $k$ of the value iteration algorithm, then* $\lim_{k\to\infty} V_k = V^*$.

**Proof**

$$\begin{aligned} ||V^* - V_k||_\infty &\overset{(a)}{=} ||\mathcal{T} V^* - \mathcal{T} V_{k-1}||_\infty \overset{(b)}{\leq} \gamma ||V^* - V_{k-1}||_\infty \\ &\overset{(c)}{\leq} \gamma^k ||V^* - V_0||_\infty \xrightarrow{k\to\infty} 0. \end{aligned}$$

**(a)** This is from the fact that $V^*$ is the unique fixed-point of the Bellman optimality operator $\mathcal{T}$ and the definition of $V_k$ in the value iteration algorithm.
**(b)** This is from the max-norm contraction property of the Bellman optimality operator $\mathcal{T}$.
**(c)** This is by replacing $V_{k-1}$ with $\mathcal{T}V_{k-2}$ and using the contraction property of $\mathcal{T}$, and then repeating this process for $k-1$ steps. ■

**Policy Iteration (PI)** [Howard, 1960] is an iterative algorithm that starts with an arbitrary initial policy $\pi_0$ and discovers a deterministic optimal policy by generating a sequence of monotonically improving policies. Each iteration $k$ of PI consists of two phases: *policy evaluation* in which the value function of the current policy $V^{\pi_{k-1}}$ is computed, and *policy improvement* in which the new (improved) policy $\pi_k$ is generated as a *greedy* policy w.r.t. $V^{\pi_{k-1}}$, i.e.,

$$\begin{aligned}\forall x \in \mathcal{X}, \qquad \pi_k(x) &= (\mathcal{G}\pi_{k-1})(x) \\ &= \arg\max_{a\in\mathcal{A}} \left[ r(x,a) + \gamma \sum_{x'\in\mathcal{X}} P(x'|x,a) V^{\pi_{k-1}}(x') \right] \\ &= \arg\max_{a\in\mathcal{A}} Q^{\pi_{k-1}}(x,a),\end{aligned}$$

where $\mathcal{G}$ is the greedy operator that maps a policy to the greedy policy w.r.t. its value function as defined above. It is easy to show that the *greedification* process in the policy improvement phase guarantees that the new policy $\pi_k$ is not worse than the current policy $\pi_{k-1}$.

**Lemma 2** *Let $\pi_{k-1}$ and $\pi_k$ be the current and next policies at iteration $k$ of the policy iteration algorithm. Then, the new policy $\pi_k$ is not worse than the current policy $\pi_{k-1}$, i.e., $\forall x \in \mathcal{X}$, $V^{\pi_{k-1}}(x) \leq V^{\pi_k}(x)$.*

**Proof**

$$V^{\pi_{k-1}} \overset{\text{(a)}}{=} \mathcal{T}^{\pi_{k-1}} V^{\pi_{k-1}} \overset{\text{(b)}}{\leq} \mathcal{T} V^{\pi_{k-1}} \overset{\text{(c)}}{=} \mathcal{T}^{\pi_k} V^{\pi_{k-1}} \overset{\text{(d)}}{\leq} \lim_{n\to\infty} (\mathcal{T}^{\pi_k})^n V^{\pi_{k-1}} = V^{\pi_k}.$$

**(a)** This is from the fact that $V^{\pi_{k-1}}$ is the unique fixed-point of the Bellman operator $\mathcal{T}^{\pi_{k-1}}$.

**(b)** This is from the fact that the Bellman optimality operator $\mathcal{T}$ uses the max, and thus, is bigger when it is applied to a function $V^{\pi_{k-1}}$ than the Bellman operator $\mathcal{T}^{\pi_{k-1}}$.

**(c)** This is from the definition of $\pi_k$ as the greedy policy w.r.t. $\pi_{k-1}$.

**(d)** This is from the monotonicity property of the Bellman operator $\mathcal{T}^{\pi_k}$. ■

Lemma 2 indicates that the sequence of policies generated by PI is monotonically improving. Therefore, when the numbers of states and actions are finite (i.e., the total number of policies is finite), PI stops after a finite number of iterations with an optimal policy $\pi^*$ **(convergence proof of PI)**.

The computational cost at each iteration of PI is $O\big(\max\{|\mathcal{X}|^3, |\mathcal{X}|^2|\mathcal{A}|\}\big)$, which is the combination of the costs of the policy evaluation[3] $O(|\mathcal{X}|^2|\mathcal{A}|)$ and policy improvement $O(|\mathcal{X}|^3)$ steps. Although PI is more computationally expensive than VI at each iteration, it usually takes less iterations to converge than VI. It has been recently shown that the maximum number of iterations that PI needs to converge to an optimal policy is polynomial in the number of states and actions [Ye, 2011, Hansen et al., 2013, Scherrer, 2013a]. This means that the PI algorithm is in fact *strongly polynomial* in the number of states and actions.

[3] The cost of the policy evaluation phase is the cost of solving a linear system of equations of size $|\mathcal{X}|$, which is the cost of inverting a square matrix of size $|\mathcal{X}|$. Using the new improvements in matrix inversion, this phase can be done in $O(|\mathcal{X}|^{2.807})$.

## 1.3 Approximate Dynamic Programming

As shown in Section 1.2, DP algorithms are the most powerful tools to solve a RL problem, i.e., to find an optimal policy. However, these algorithms guarantee to find an optimal policy only if the environment (i.e., the dynamics and the rewards) is completely known and the size of the state and action spaces are not too large. When one of these conditions is violated, e.g.,

- the state space $\mathcal{X}$ and/or action space $\mathcal{A}$ are large or infinite,
- the model of the system (the transition probability $P$ and reward $r$ functions) is unknown and the only information about the environment is of the form of samples of transitions and rewards,
- we do not have enough time and/or sample to compute the quantity of interest at each iteration $k$ of the DP algorithm (i.e., $\mathcal{T}V_{k-1}$ for VI and $V^{\pi_{k-1}}$ for PI algorithms),

approximate algorithms are needed, and thus, DP methods turn to approximate dynamic programming (ADP) and RL algorithms. Unfortunately, in this case, the convergence and performance guarantees of the standard DP algorithms (Lemmas 1 and 2) are no longer valid, and the main theoretical challenge is to study the performance of ADP and RL algorithms, which is the main focus of this work.

Assume that at the $k$'th iteration of VI, we are not able to compute $\mathcal{T}V_{k-1}$, and have its approximation instead. As a result, the next value function $V_k$ won't be $\mathcal{T}V_{k-1}$ as in the standard VI, but will be an approximation of this quantity, i.e., $V_k \approx \mathcal{T}V_{k-1}$. Thus, we can no longer use the max-norm contraction property of the Bellman optimality operator and write

$$||V^* - V_k||_\infty \leq \gamma ||V^* - V_{k-1}||_\infty.$$

As it was shown, this property is at the heart of the proof of the standard VI, and thus without it, it is not possible to prove the convergence of the approximate value iteration (AVI) algorithm to the optimal value function.

Similarly, assume that at the $k$'th iteration of PI, we are not able to compute the value of the current policy $V^{\pi_{k-1}}$, and have its approximation $\widehat{V}^{\pi_{k-1}} \approx V^{\pi_{k-1}}$ instead. As a result, the next policy generated by PI, i.e., the greedy policy w.r.t. $\widehat{V}^{\pi_{k-1}}$, is no longer the greedy policy w.r.t. $\pi_{k-1}$, i.e.,

$$\pi_k(x) = \arg\max_{a \in \mathcal{A}} \left[ r(x,a) + \gamma \sum_{x' \in \mathcal{X}} P(x'|x,a) \widehat{V}^{\pi_{k-1}}(x') \right] \neq (\mathcal{G}\pi_{k-1})(x),$$

and thus, we cannot guarantee that $V^{\pi_k} \geq V^{\pi_{k-1}}$, and maintain the monotonically improving behavior of the standard PI. As it was shown, this property is at the heart of the proof of the standard PI, and thus without it, it is not possible to prove the convergence of the approximate policy iteration (API) algorithm to an optimal policy.

Now that the ADP algorithms fail to converge to the optimal value function or an optimal policy, it is quite important to find out how close they get to an optimal solution given a fixed budget of computation (computation complexity) or samples (sample complexity).[4] In this work, we are mainly interested in the *sample complexity* of the ADP algorithms and aim to answer the following question:

*If we are given a fixed number of samples (or samples per iteration) and we run an ADP algorithm for a number of iterations $K > 0$, how close would the quantity returned by the algorithm be to its optimal value?*

More precisely, if an ADP algorithm returns the policy $\pi_K$ after $K$ iterations, we would like to have a bound on $||V^* - V^{\pi_K}||$ as a function of the sample budget and the approximation scheme used by the algorithm.[5] Note

[4] Note that samples are of the form of transitions and rewards, and thus, the number of samples shows our number of interactions with the system or its simulator.

[5] In the case of AVI, $\pi_K$ is the greedy policy w.r.t. the value function $V_K$ returned by the algorithm, i.e., $\forall x \in \mathcal{X},\ \pi_K(x) = \arg\max_{a \in \mathcal{A}} \big[ r(x,a) + \gamma \sum_{x' \in \mathcal{X}} P(x'|x,a) V_K(x') \big]$.

that the norm $||\cdot||$ can be any $\ell_p$-norm, $p = 1, \ldots, \infty$, but $\ell_1$ and $\ell_2$ norms are more desirable, and we try to avoid $\ell_\infty$-norm as it could be very loose.

# Chapter 2

# Finite-Sample Analysis of Least-Squares Policy Iteration

**[MGH1, MGH9, MGH12, MGH15, MGH18]**

In this paper, we report a performance bound for the widely used least-squares policy iteration (LSPI) algorithm. We first consider the problem of policy evaluation in reinforcement learning, i.e., learning the value function of a fixed policy, using the least-squares temporal-difference (LSTD) learning method, and report finite-sample analysis for this algorithm. To do so, we first derive a bound on the performance of the LSTD solution evaluated at the states generated by the Markov chain and used by the algorithm to learn an estimate of the value function. This result is general in the sense that no assumption is made on the existence of a stationary distribution for the Markov chain. We then derive generalization bounds in the case when the Markov chain possesses a stationary distribution and is $\beta$-mixing. Finally, we analyze how the error at each policy evaluation step is propagated through the iterations of a policy iteration method, and derive a performance bound for the LSPI algorithm.

## 2.1 Introduction

Least-squares temporal-difference (LSTD) learning [Bradtke and Barto, 1996, Boyan, 1999] is a widely used algorithm for prediction in general, and in the context of reinforcement learning (RL), for learning the value function $V^\pi$ of a given policy $\pi$. LSTD has been successfully applied to a number of problems especially after the development of the least-squares policy iteration (LSPI) algorithm [Lagoudakis and Parr, 2003a], which extends LSTD to control by using it in the policy evaluation step of policy iteration. More precisely, LSTD computes the fixed point of the operator $\Pi\mathcal{T}$, where $\mathcal{T}$ is the Bellman operator and $\Pi$ is the projection operator in a linear function space $\mathcal{F}$. Although LSTD and LSPI have been widely used in the RL community, a finite-sample analysis of LSTD, i.e., performance bounds in terms of the number of samples, the space $\mathcal{F}$, and the characteristic parameters of the MDP at hand, is still missing.

Most of the theoretical work analyzing LSTD have been focused on the model-based case, where explicit models of the reward function and the dynamics are available. In particular, Tsitsiklis and Van Roy [1997] showed that the distance between the LSTD solution and the value function $V^\pi$ is bounded by the distance between $V^\pi$ and its closest approximation in the linear space, multiplied by a constant which increases as the discount factor approaches 1. In this bound, it is assumed that the Markov chain possesses a stationary distribution $\rho^\pi$ and the distances are measured according to $\rho^\pi$. Yu [2010] has extended this analysis and derived an asymptotic convergence analysis for off-policy LSTD($\lambda$), that is when the samples are collected following a behavior policy different from the policy $\pi$ under evaluation. Finally, on-policy empirical LSTD has been analyzed by Bertsekas [2007]. His analysis reveals a critical dependency on the inverse of the smallest eigenvalue of the LSTD's $A$ matrix (note that the LSTD solution is obtained by solving a system of linear equations $Ax = b$). Nonetheless, Bertsekas [2007] does not provide a finite-sample analysis of the algorithm. Although these analyses already provide some insights on the behavior of LSTD, asymptotic results do not give a full characterization of the performance of the algorithm when only a finite number of samples is available (which is the most common situation in practice). On the other hand, a finite-sample analysis has a number of important advantages: **1)** unlike in Tsitsiklis and Van Roy [1997], where they assume that model-based LSTD always returns a solution, in a finite-sample analysis we study the characteristics of the actual empirical LSTD fixed point, including its existence, **2)** a finite-sample bound explicitly reveals how the prediction error of LSTD is related to the characteristic parameters of the MDP at hand, such as the discount factor, the dimensionality of the function space $\mathcal{F}$, and the number of samples, **3)** once this dependency is clear, the bound can be used to determine the order of magnitude of the number of samples needed to achieve a desired accuracy.

Recently, several works have been focused on deriving a finite-sample analysis for different RL algorithms. In the following, we review those that are more strictly related to LSTD and to the results reported in this paper. Antos et al. [2008] analyzed the modified Bellman residual (MBR) minimization algorithm for a finite number of samples, bounded function spaces, and a $\mu$-norm that might be different from the norm induced by $\rho^\pi$. Although MBR minimization was shown to reduce to LSTD in case of linear spaces, it is not straightforward to extend the finite-sample bounds derived by Antos et al. [2008] to unbounded linear spaces considered by LSTD. Farahmand et al. [2008] proposed a $\ell_2$-regularized extension of LSPI and provided finite-sample analysis for the algorithm when the function space is a reproducing kernel Hilbert space (RKHS). In this work, the authors consider the optimization formulation of LSTD (instead of the better known fixed-point formulation) and assume that a generative model of the environment is available. Moreover, the analysis is for $\ell_2$-regularized LSTD (LSPI) and also for the case that the function space $\mathcal{F}$ is a RKHS. Ávila Pires and Szepesvári [2012] also analyzed a regularized version of LSTD reporting performance bounds for both the on-policy and off-policy case. In this paper, we first report a finite-sample analysis of LSTD. To the best of our knowledge, this is the first complete finite-sample analysis of this widely used algorithm. Our analysis is for a specific implementation of LSTD that we call *pathwise LSTD*. Pathwise LSTD has two specific characteristics: **1)** it takes a single trajectory generated by the Markov chain induced by policy $\pi$ as input, and **2)** it uses the pathwise Bellman operator (precisely defined in Section 2.3), which is defined to be a contraction w.r.t. the empirical norm. We first derive a bound on the performance of the pathwise LSTD solution for a setting that we call *Markov design*. In this setting, the performance is evaluated at the points used by the algorithm to learn an estimate of $V^{\pi}$. This bound is general in the sense that no assumption is made on the existence of a stationary distribution for the Markov chain. Then, in the case that the Markov chain admits a stationary distribution $\rho^{\pi}$ and is $\beta$-mixing, we derive generalization bounds w.r.t. the norm induced by $\rho^{\pi}$. Finally, along the lines of Antos et al. [2008], we show how the LSTD error is propagated through the iterations of LSPI, and under suitable assumptions, derive a performance bound for the LSPI algorithm.

Besides providing a full finite-sample analysis of LSPI, the major insights gained by the analysis in the paper may be summarized as follows. The first result is about the existence of the LSTD solution and its performance. In Theorem 3 we show that with a slight modification of the empirical Bellman operator $\widehat{\mathcal{T}}$ (leading to the definition of pathwise LSTD), the operator $\widehat{\Pi}\widehat{\mathcal{T}}$ (where $\widehat{\Pi}$ is an empirical projection operator) always has a fixed point $\hat{v}$, even when the sample-based Gram matrix is not invertible and the Markov chain does not admit a stationary distribution. In this very general setting, it is still possible to derive a bound for the performance of the LSTD solution, $\hat{v}$, evaluated at the states of the trajectory used by the algorithm. Moreover, an analysis of the bound reveals a critical dependency on the smallest strictly positive eigenvalue $\nu_n$ of the sample-based Gram matrix. Then, in the case in which the Markov chain has a stationary distribution $\rho^\pi$, it is possible to relate the value of $\nu_n$ to the smallest eigenvalue of the Gram matrix defined according to $\rho^\pi$. Furthermore, it is possible to generalize the previous performance bound over the entire state space under the measure $\rho^\pi$, when the samples are drawn from a stationary $\beta$-mixing process (Theorem 7). It is important to note that the asymptotic bound obtained by taking the number of samples, $n$, to infinity is equal (up to constants) to the bound in Tsitsiklis and Van Roy [1997] for model-based LSTD. Furthermore, a comparison with the bounds in Antos et al. [2008] shows that we successfully leverage on the specific setting of LSTD: **1)** the space of functions is linear, and **2)** the distribution used to evaluate the performance is the stationary distribution of the Markov chain induced by the policy, and obtain a better bound both in terms of **1)** estimation error, a rate of order $O(1/n)$ instead of $O(1/\sqrt{n})$ for the squared error, and **2)** approximation error, the minimal distance between the value function $V^\pi$ and the space $\mathcal{F}$ instead of the inherent Bellman errors of $\mathcal{F}$. The extension in Theorem 8 to the case in which the samples belong to a trajectory generated by a fast mixing Markov chain shows that it is possible to achieve the same performance as in the case of stationary $\beta$-mixing processes. Finally, the analysis of LSPI reveals the need for several critical assumptions on the stationary distributions of the policies that are greedy w.r.t. to the functions in the linear space $\mathcal{F}$. These assumptions seem unavoidable when an on-policy method is used at each iteration, and whether they can be removed or relaxed in other settings is still an open question. This paper extends and improves over the conference paper by Lazaric et al. [2010c] in the following respects: **1)** we report the full proofs and technical tools for all the theoretical results, thus making the paper self-contained, **2)** we extend the LSTD results to LSPI showing how the approximation errors are propagated through iterations.

The rest of the chapter is organized as follows. In Section 2.2, we set the notation used throughout the paper. In Section 2.3, we introduce pathwise LSTD by a minor modification to the standard LSTD formulation in order to guarantee the existence of at least one solution. In Section 2.4, we introduce the Markov design setting for regression and report an empirical bound for LSTD. In Section 2.5, we show how the Markov design bound of Section 2.4 may be extended when the Markov chain admits a stationary distribution. In Section 2.6, we analyze how the LSTD error is propagated through the iterations of LSPI and derive a performance bound for the LSPI algorithm. Finally in Section 2.7, we draw conclusions and discuss some possible directions for future work.

## 2.2 Preliminaries

For a measurable space with domain $\mathcal{X}$, we let $\mathcal{S}(\mathcal{X})$ and $\mathcal{B}(\mathcal{X}; L)$ denote the set of probability measures over $\mathcal{X}$, and the space of bounded measurable functions with domain $\mathcal{X}$ and bound $0 < L < \infty$, respectively. For a measure $\rho \in \mathcal{S}(\mathcal{X})$ and a measurable function $f : \mathcal{X} \to \mathbb{R}$, we define the $\ell_2(\rho)$-norm of $f$, $||f||_\rho$, and for a set of $n$ points $X_1, \ldots, X_n \in \mathcal{X}$, we define the empirical norm $||f||_n$ as

$$||f||_\rho^2 = \int f(x)^2 \rho(dx) \qquad \text{and} \qquad ||f||_n^2 = \frac{1}{n} \sum_{t=1}^{n} f(X_t)^2.$$

The supremum norm of $f$, $||f||_\infty$, is defined as $||f||_\infty = \sup_{x \in \mathcal{X}} |f(x)|$.

We consider the standard RL framework [Bertsekas and Tsitsiklis, 1996, Sutton and Barto, 1998] in which a learning agent interacts with a stochastic environment and this interaction is modeled as a discrete-time discounted Markov decision process (MDP). A discounted MDP is a tuple $\mathcal{M} = \langle \mathcal{X}, \mathcal{A}, r, P, \gamma \rangle$ where the state space $\mathcal{X}$ is a bounded closed subset of the $s$-dimensional Euclidean space, $\mathcal{A}$ is a finite ($|\mathcal{A}| < \infty$) action space, the reward function $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is uniformly bounded by $R_{\max}$, the transition kernel $P$ is such that for all $x \in \mathcal{X}$ and $a \in \mathcal{A}$, $P(\cdot|x, a)$ is a distribution over $\mathcal{X}$, and $\gamma \in (0, 1)$

is a discount factor. A deterministic policy $\pi : \mathcal{X} \to \mathcal{A}$ is a mapping from states to actions. For a given policy $\pi$, the MDP $\mathcal{M}$ is reduced to a Markov chain $\mathcal{M}^\pi = \langle \mathcal{X}, R^\pi, P^\pi, \gamma \rangle$ with the reward function $R^\pi(x) = r\big(x, \pi(x)\big)$, transition kernel $P^\pi(\cdot|x) = P\big(\cdot\,|x, \pi(x)\big)$, and stationary distribution $\rho^\pi$ (if it admits one). The value function of a policy $\pi$, $V^\pi$, is the unique fixed-point of the Bellman operator $\mathcal{T}^\pi : \mathcal{B}(\mathcal{X}; V_{\max} = \frac{R_{\max}}{1-\gamma}) \to \mathcal{B}(\mathcal{X}; V_{\max})$ defined by

$$(\mathcal{T}^\pi V)(x) = R^\pi(x) + \gamma \int_{\mathcal{X}} P^\pi(dy|x) V(y).$$

We also define the optimal value function $V^*$ as the unique fixed-point of the optimal Bellman operator $\mathcal{T}^* : \mathcal{B}(\mathcal{X}; V_{\max}) \to \mathcal{B}(\mathcal{X}; V_{\max})$ defined by

$$(\mathcal{T}^* V)(x) = \max_{a \in \mathcal{A}} \left[ r(x,a) + \gamma \int_{\mathcal{X}} P(dy|x,a) V(y) \right].$$

In the following sections, to simplify the notation, we remove the dependency to the policy $\pi$ and use $R$, $P$, $V$, $\rho$, and $\mathcal{T}$ instead of $R^\pi$, $P^\pi$, $V^\pi$, $\rho^\pi$, and $\mathcal{T}^\pi$ whenever the policy $\pi$ is fixed and clear from the context.

To approximate the value function $V$, we use a linear approximation architecture with parameters $\alpha \in \mathbb{R}^d$ and basis functions $\varphi_i \in \mathcal{B}(\mathcal{X}; L)$, $i = 1, \ldots, d$. We denote by $\phi : \mathcal{X} \to \mathbb{R}^d$, $\phi(\cdot) = \big(\varphi_1(\cdot), \ldots, \varphi_d(\cdot)\big)^\top$ the feature vector, and by $\mathcal{F}$ the linear function space spanned by the basis functions $\varphi_i$. Thus $\mathcal{F} = \big\{ f_\alpha \mid \alpha \in \mathbb{R}^d \text{ and } f_\alpha(\cdot) = \phi(\cdot)^\top \alpha \big\}$.

Let $(X_1, \ldots, X_n)$ be a sample path (trajectory) of size $n$ generated by the Markov chain $\mathcal{M}^\pi$. Let $v \in \mathbb{R}^n$ and $r \in \mathbb{R}^n$ be such that $v_t = V(X_t)$ and $r_t = R(X_t)$ be the value vector and the reward vector, respectively. Also, let $\Phi = [\phi(X_1)^\top; \ldots; \phi(X_n)^\top]$ be the feature matrix defined at the states, and $\mathcal{F}_n = \{\Phi\alpha,\ \alpha \in \mathbb{R}^d\} \subset \mathbb{R}^n$ be the corresponding vector space. We denote by $\widehat{\Pi} : \mathbb{R}^n \to \mathcal{F}_n$ the orthogonal projection onto $\mathcal{F}_n$, defined as $\widehat{\Pi} y = \arg\min_{z \in \mathcal{F}_n} ||y - z||_n$, where $||y||_n^2 = \frac{1}{n}\sum_{t=1}^n y_t^2$. Note that the orthogonal projection $\widehat{\Pi} y$ for any $y \in \mathbb{R}^n$ exists and is unique. Moreover, $\widehat{\Pi}$ is a non-

```
Input: Linear space $\mathcal{F} = \mathrm{span}\{\varphi_i, 1 \le i \le d\}$, sample trajectory $\{(x_t, r_t)\}_{t=1}^n$
of the Markov chain

Build the feature matrix $\Phi = [\phi(x_1)^\top; \ldots; \phi(x_n)^\top]$
Build the empirical transition matrix $\hat{P} : \hat{P}_{ij} = \mathbb{I}\{j = i + 1, j \neq n\}$
Build matrix $A = \Phi^\top (I - \gamma \widehat{P}) \Phi$
Build vector $b = \Phi^\top r$
Return the pathwise LSTD solution $\widehat{\alpha} = A^+ b$
```

Figure 2.1: A pseudo-code for the batch pathwise LSTD algorithm.

expansive mapping w.r.t. the $\ell_2$-norm: since the projection is orthogonal and using the Cauchy-Schwarz inequality $||\widehat{\Pi} y - \widehat{\Pi} z||_n^2 = \langle y - z, \widehat{\Pi} y - \widehat{\Pi} z \rangle_n \le ||y - z||_n ||\widehat{\Pi} y - \widehat{\Pi} z||_n$, and thus, we obtain $||\widehat{\Pi} y - \widehat{\Pi} z||_n \le ||y - z||_n$.

## 2.3 Pathwise LSTD

Pathwise LSTD (Algorithm 4.3) is a version of LSTD that takes as input a linear function space $\mathcal{F}$ and a single trajectory $X_1, \ldots, X_n$ generated by following the policy, and returns the fixed-point of the empirical operator $\widehat{\Pi} \widehat{\mathcal{T}}$, where $\widehat{\mathcal{T}} : \mathbb{R}^n \to \mathbb{R}^n$ is the *pathwise Bellman operator* defined as

$$(\widehat{\mathcal{T}} y)_t = \begin{cases} r_t + \gamma y_{t+1} & 1 \le t < n, \\ r_t & \quad\; t = n. \end{cases}$$

Note that by defining the operator $\widehat{P} : \mathbb{R}^n \to \mathbb{R}^n$ as $(\widehat{P} y)_t = y_{t+1}$ for $1 \le t < n$ and $(\widehat{P} y)_n = 0$, we have $\widehat{\mathcal{T}} y = r + \gamma \widehat{P} y$. The motivation for using the pathwise Bellman operator is that it is $\gamma$-contraction in $\ell_2$-norm, i.e., for any $y, z \in \mathbb{R}^n$, we have

$$||\widehat{\mathcal{T}} y - \widehat{\mathcal{T}} z||_n^2 = ||\gamma \widehat{P} (y - z)||_n^2 \le \gamma^2 ||y - z||_n^2 \,.$$

Since the orthogonal projection $\widehat{\Pi}$ is non-expansive w.r.t. $\ell_2$-norm, from Banach fixed point theorem, there exists a unique fixed-point $\hat{v}$ of the mapping

$\widehat{\Pi}\widehat{\mathcal{T}}$, i.e., $\hat{v} = \widehat{\Pi}\widehat{\mathcal{T}}\hat{v}$. Since $\hat{v}$ is the unique fixed point of $\widehat{\Pi}\widehat{\mathcal{T}}$, the vector $\hat{v} - \widehat{\mathcal{T}}\hat{v}$ is perpendicular to the space $\mathcal{F}_n$, and thus, $\Phi^\top(\hat{v} - \widehat{\mathcal{T}}\hat{v}) = 0$. By replacing $\hat{v}$ with $\Phi\alpha$, we obtain $\Phi^\top\Phi\alpha = \Phi^\top(r + \gamma\widehat{P}\Phi\alpha)$ and then $\Phi^\top(I - \gamma\widehat{P})\Phi\alpha = \Phi^\top r$. Therefore, by setting $A = \Phi^\top(I - \gamma\widehat{P})\Phi$ and $b = \Phi^\top r$, we recover a $d \times d$ system of equations $A\alpha = b$ similar to the one in the original LSTD algorithm. Note that since the fixed point $\hat{v}$ exists, this system always has at least one solution. We call the solution with minimal norm, $\widehat{\alpha} = A^+ b$, where $A^+$ is the Moore-Penrose pseudo-inverse of $A$, the pathwise LSTD solution.[1]

Finally, notice that the algorithm reported in Figure 4.3 may be easily extended to the incremental version of LSTD by incrementally building the inverse of the matrix $A$ as the samples are collected.

## 2.4 Markov Design Bound

In Section 2.3, we defined the pathwise Bellman operator with a slight modification in the definition of the empirical Bellman operator $\widehat{\mathcal{T}}$, and showed that the operator $\widehat{\Pi}\widehat{\mathcal{T}}$ always has a unique fixed point $\hat{v}$. In this section, we derive a bound for the performance of $\hat{v}$ evaluated at the states of the trajectory used by the pathwise LSTD algorithm. We first state the main theorem and we discuss it in a number of remarks. The proofs are postponed at the end of the section.

**Theorem 3** *Let $X_1, \ldots, X_n$ be a trajectory generated by the Markov chain, and $v, \hat{v} \in \mathbb{R}^n$ be the vectors whose components are the value function and the pathwise LSTD solution at $\{X_t\}_{t=1}^n$, respectively. Then with probability at least $1 - \delta$ (the probability is w.r.t. the random trajectory), we have*

$$||v - \hat{v}||_n \leq \frac{1}{\sqrt{1-\gamma^2}} ||v - \widehat{\Pi}v||_n + \frac{1}{1-\gamma}\left[\gamma V_{\max} L \sqrt{\frac{d}{\nu_n}} \Big(\sqrt{\frac{8\log(2d/\delta)}{n}} + \frac{1}{n}\Big)\right], \tag{2.1}$$

[1] Note that whenever the matrix $A$ is invertible $A^+ = A^{-1}$.

*where the random variable $\nu_n$ is the smallest strictly-positive eigenvalue of the sample-based Gram matrix $\frac{1}{n}\Phi^\top\Phi$.*

**Remark 1** Theorem 3 provides a bound on the prediction error of the LSTD solution $\hat{v}$ w.r.t. the true value function $v$ on the trajectory $X_1, \ldots, X_n$ used as a training set for pathwise-LSTD. The bound contains two main terms. The first term $||v - \widehat{\Pi}v||_n$ is the *approximation* error and it represents the smallest possible error in approximating $v$ with functions in $\mathcal{F}$. This error cannot be avoided. The second term, of order $O(\sqrt{d/n})$, is the *estimation* error and it accounts for the error due to the use of a finite number of noisy samples and it shows what is the influence of the different elements of the problem (e.g., $\gamma$, $d$, $n$) on the prediction error and it provides insights about how to tune some parameters. We first notice that the bound suggests that the number of samples $n$ should be significantly bigger than the number of features $d$ in order to achieve a small estimation error. Furthermore, the bound can be used to estimate the number of samples needed to guarantee a desired prediction error $\epsilon$. In fact, apart from the approximation error, which is unavoidable, we have that $n = O(d/((1-\gamma)^2\epsilon^2))$ samples are enough to achieve an $\epsilon$-accurate approximation of the true value function $v$. We also remark that one might be tempted to reduce the dimensionality $d$, so as to reduce the sample cost of the algorithm. Nonetheless, this is likely to reduce the approximation capability of $\mathcal{F}$ and thus increase the approximation error.

**Remark 2** When the eigenvalues of the sample-based Gram matrix $\frac{1}{n}\Phi^\top\Phi$ are all non-zero, $\Phi^\top\Phi$ is invertible, and thus, $\widehat{\Pi} = \Phi(\Phi^\top\Phi)^{-1}\Phi^\top$. In this case, the uniqueness of $\hat{v}$ implies the uniqueness of $\widehat{\alpha}$ since

$$\hat{v} = \Phi\alpha \implies \Phi^\top\hat{v} = \Phi^\top\Phi\alpha \implies \widehat{\alpha} = (\Phi^\top\Phi)^{-1}\Phi^\top\hat{v}.$$

On the other hand, when the sample-based Gram matrix $\frac{1}{n}\Phi^\top\Phi$ is not invertible, the system $Ax = b$ may have many solutions. Among all the possible solutions, one may choose the one with minimal norm: $\widehat{\alpha} = A^{+}b$.

**Remark 3** Note that in case there exists a constant $\nu > 0$, such that with probability $1 - \delta'$ all the eigenvalues of the sample-based Gram matrix are lower-bounded by $\nu$, Eq. 2.1 (with $\nu_n$ replaced by $\nu$) holds with probability at least $1 - (\delta + \delta')$ (see Section 2.5.1 for a case in which such constant $\nu$ can be computed and it is related to the smallest eigenvalue of the model based Gram matrix).

**Remark 4** Theorem 3 provides a bound without any reference to the stationary distribution of the Markov chain. In fact, the bound of Eq. 2.1 holds even when the chain does not admit a stationary distribution. For example, consider a Markov chain on the real line where the transitions always move the states to the right, i.e., $p(X_{t+1} \in dy | X_t = x) = 0$ for $y \leq x$. For simplicity assume that the value function $V$ is bounded and belongs to $\mathcal{F}$. This Markov chain is not recurrent, and thus, does not have a stationary distribution. We also assume that the feature vectors $\phi(X_1), \ldots, \phi(X_n)$ are sufficiently independent, so that all the eigenvalues of $\frac{1}{n}\Phi^\top \Phi$ are greater than $\nu > 0$. Then according to Theorem 3, pathwise LSTD is able to estimate the value function at the samples at a rate $O(1/\sqrt{n})$. This may seem surprising because at each state $X_t$ the algorithm is only provided with a noisy estimation of the expected value of the next state. However, the estimates are unbiased conditioned on the current state, and we will see in the proof that using a concentration inequality for martingale, pathwise LSTD is able to learn a good estimate of the value function at a state $X_t$ using noisy pieces of information at other states that may be far away from $X_t$. In other words, learning the value function at a given state does not require making an average over many samples close to that state. This implies that LSTD does not require the Markov chain to possess a stationary distribution.

**Remark 5** The most critical part of the bound in Eq. 2.1 is the inverse dependency on the smallest positive eigenvalue $\nu_n$. A similar dependency is shown in the LSTD analysis of Bertsekas [2007]. The main difference is that here we have a more complete finite-sample analysis with an explicit dependency on the number of samples and the other characteristic parameters of the problem. Furthermore, if the Markov chain admits a stationary distribution $\rho$, we are able to relate the existence of the LSTD solution to the smallest eigenvalue of the Gram matrix defined according to $\rho$ (see Section 2.5.1).

In order to prove Theorem 3, we first introduce the regression setting with *Markov design* and then state and prove a lemma about this model. Delattre and Gaïffas [2011] recently analyzed a similar setting in the general case of martingale incremental errors.

**Definition 4** *The model of regression with* ***Markov design*** *is a regression problem where the data $(X_t, Y_t)_{1\le t\le n}$ are generated according to the following model: $X_1, \dots, X_n$ is a sample path generated by a Markov chain, $Y_t = f(X_t) + \xi_t$, where $f$ is the target function, and the noise term $\xi_t$ is a random variable which is adapted to the filtration generated by $X_1, \dots, X_{t+1}$ and is such that*

$$|\xi_t| \le C \quad \textit{and} \quad \mathbb{E}[\xi_t | X_1, \dots, X_t] = 0. \tag{2.2}$$

The next lemma reports a risk bound for the Markov design setting which is of independent interest.

**Lemma 5 (Regression bound for the Markov design setting)** *We consider the model of regression with Markov design in Definition 4. Let $\hat{w} \in \mathcal{F}_n$ be the least-squares estimate of the (noisy) values $Y = \{Y_t\}_{t=1}^n$, i.e., $\hat{w} = \widehat{\Pi} Y$, and $w \in \mathcal{F}_n$ be the least-squares estimate of the (noiseless) values $Z = \{Z_t =$*

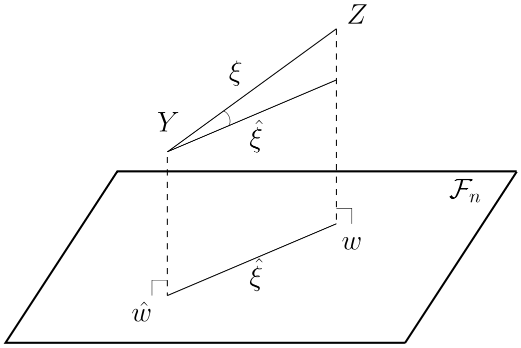


Figure 2.2: This figure shows the components used in Lemma 5 and its proof such as $w$, $\hat{w}$, $\xi$, and $\hat{\xi}$, and the fact that $\langle \hat{\xi}, \xi \rangle_n = ||\hat{\xi}||_n^2$.

*$f(X_t)\}_{t=1}^n$, i.e., $w = \widehat{\Pi} Z$. Then for any $\delta > 0$, with probability at least $1 - \delta$ (the probability is w.r.t. the random sample path $X_1, \ldots, X_n$), we have*

$$||\hat{w} - w||_n \leq CL \sqrt{\frac{2d \log(2d/\delta)}{n \nu_n}} \; , \tag{2.3}$$

*where $\nu_n$ is the smallest strictly-positive eigenvalue of the sample-based Gram matrix $\frac{1}{n}\Phi^\top \Phi$.*

**Proof** [Lemma 5] We define $\xi \in \mathbb{R}^n$ to be the vector with components $\xi_t = Y_t - Z_t$, and $\hat{\xi} = \hat{w} - w = \widehat{\Pi}(Y - Z) = \widehat{\Pi}\xi$. Since the projection is orthogonal we have $\langle \hat{\xi}, \xi \rangle_n = ||\hat{\xi}||_n^2$ (see Figure 2.2). Since $\hat{\xi} \in \mathcal{F}_n$, there exists at least one $\alpha \in \mathbb{R}^d$ such that $\hat{\xi} = \Phi\alpha$, so by Cauchy-Schwarz inequality we have

$$||\hat{\xi}||_n^2 = \langle \hat{\xi}, \xi \rangle_n = \frac{1}{n} \sum_{i=1}^d \alpha_i \sum_{t=1}^n \xi_t \varphi_i(X_t) \leq \frac{1}{n} ||\alpha||_2 \left[ \sum_{i=1}^d \Big( \sum_{t=1}^n \xi_t \varphi_i(X_t) \Big)^2 \right]^{1/2} . \tag{2.4}$$

Now among the vectors $\alpha$ such that $\hat{\xi} = \Phi\alpha$, we define $\widehat{\alpha}$ to be the one with minimal $\ell_2$-norm, i.e., $\widehat{\alpha} = \Phi^+ \hat{\xi}$. Let $K$ denote the null-space of $\Phi$, which is also the null-space of $\frac{1}{n}\Phi^\top\Phi$. Then $\widehat{\alpha}$ may be decomposed as $\widehat{\alpha} = \widehat{\alpha}_K + \widehat{\alpha}_{K^\perp}$, where $\widehat{\alpha}_K \in K$ and $\widehat{\alpha}_{K^\perp} \in K^\perp$, and because the decomposition is orthogonal, we have $||\widehat{\alpha}||_2^2 = ||\widehat{\alpha}_K||_2^2 + ||\widehat{\alpha}_{K^\perp}||_2^2$. Since $\widehat{\alpha}$ is of minimal norm among all the vectors $\alpha$ such that $\hat{\xi} = \Phi\alpha$, its component in $K$ must be zero, thus $\widehat{\alpha} \in K^\perp$.

The Gram matrix $\frac{1}{n}\Phi^\top\Phi$ is positive-semidefinite, thus its eigenvectors corresponding to zero eigenvalues generate $K$ and the other eigenvectors generate its orthogonal complement $K^\perp$. Therefore, from the assumption that the smallest strictly-positive eigenvalue of $\frac{1}{n}\Phi^\top\Phi$ is $\nu_n$, we deduce that since $\widehat{\alpha} \in K^\perp$,

$$||\hat{\xi}||_n^2 = \frac{1}{n}\widehat{\alpha}^\top\Phi^\top\Phi\widehat{\alpha} \geq \nu_n\widehat{\alpha}^\top\widehat{\alpha} = \nu_n||\widehat{\alpha}||_2^2. \tag{2.5}$$

By using the result of Eq. 2.5 in Eq. 2.4, we obtain

$$||\hat{\xi}||_n \leq \frac{1}{n\sqrt{\nu_n}}\left[\sum_{i=1}^{d}\Big(\sum_{t=1}^{n}\xi_t\varphi_i(X_t)\Big)^2\right]^{1/2}. \tag{2.6}$$

Now, from the conditions on the noise in Eq. 2.2, we have that for any $i = 1, \ldots, d$

$$\mathbb{E}[\xi_t\varphi_i(X_t)|X_1, \ldots, X_t] = \varphi_i(X_t)\mathbb{E}[\xi_t|X_1, \ldots, X_t] = 0,$$

and since $\xi_t\varphi_i(X_t)$ is adapted to the filtration generated by $X_1, \ldots, X_{t+1}$, it is a martingale difference sequence w.r.t. that filtration. Thus one may apply Azuma's inequality to deduce that with probability $1 - \delta$,

$$\Big|\sum_{t=1}^{n}\xi_t\varphi_i(X_t)\Big| \leq CL\sqrt{2n\log(2/\delta)}\ ,$$

where we used that $|\xi_t\varphi_i(X_t)| \leq CL$ for any $i$ and $t$. By a union bound over all features, we have that with probability $1-\delta$, for all $1 \leq i \leq d$

$$\Big|\sum_{t=1}^{n} \xi_t \varphi_i(X_t)\Big| \leq CL\sqrt{2n\log(2d/\delta)}\ . \tag{2.7}$$

The result follows by combining Eqs. 2.7 and 2.6. ■

**Remark about Lemma 5** Note that this lemma is an extension of the bound for regression with deterministic design in which the states, $\{X_t\}_{t=1}^n$, are fixed and the noise terms, $\xi_t$'s, are independent. In deterministic design, usual concentration results provide high probability bounds similar to Eq. 2.3 (see e.g., Hsu et al., 2012), but without the dependence on $\nu_n$. An open question is whether it is possible to remove $\nu_n$ in the bound for the Markov design regression setting.

In the Markov design model considered in this lemma, states $\{X_t\}_{t=1}^n$ are random variables generated according to the Markov chain and the noise terms $\xi_t$ may depend on the next state $X_{t+1}$ (but should be centered conditioned on the past states $X_1,\dots,X_t$). This lemma will be used in order to prove Theorem 3, where we replace the target function $f$ with the value function $V$, and the noise term $\xi_t$ with the temporal difference $r(X_t)+\gamma V(X_{t+1})-V(X_t)$.

**Proof** [Theorem 3]

**Step 1:** Using the Pythagorean theorem and the triangle inequality, we have (see Figure 2.3)

$$||v-\hat{v}||_n^2 = ||v-\widehat{\Pi}v||_n^2+||\hat{v}-\widehat{\Pi}v||_n^2 \leq ||v-\widehat{\Pi}v||_n^2+\big(||\hat{v}-\widehat{\Pi}\widehat{\mathcal{T}}v||_n+||\widehat{\Pi}\widehat{\mathcal{T}}v-\widehat{\Pi}v||_n\big)^2. \tag{2.8}$$

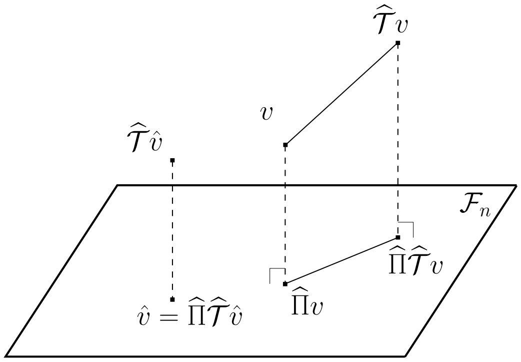

$\widehat{\mathcal{T}}v$
$v$
$\widehat{\mathcal{T}}\hat{v}$
$\mathcal{F}_n$
$\widehat{\Pi}\widehat{\mathcal{T}}v$
$\widehat{\Pi}v$
$\hat{v} = \widehat{\Pi}\widehat{\mathcal{T}}\hat{v}$

Figure 2.3: This figure represents the space $\mathbb{R}^n$, the linear vector subspace $\mathcal{F}_n$ and some vectors used in the proof of Theorem 3.

From the $\gamma$-contraction of the operator $\widehat{\Pi}\widehat{\mathcal{T}}$ and the fact that $\hat{v}$ is its unique fixed point, we obtain

$$||\hat{v} - \widehat{\Pi}\widehat{\mathcal{T}}v||_n = ||\widehat{\Pi}\widehat{\mathcal{T}}\hat{v} - \widehat{\Pi}\widehat{\mathcal{T}}v||_n \leq \gamma||\hat{v} - v||_n, \tag{2.9}$$

Thus from Eq. 2.8 and 2.9, we have

$$||v - \hat{v}||_n^2 \leq ||v - \widehat{\Pi}v||_n^2 + \big(\gamma||v - \hat{v}||_n + ||\widehat{\Pi}\widehat{\mathcal{T}}v - \widehat{\Pi}v||_n\big)^2. \tag{2.10}$$

**Step 2:** We now provide a high probability bound on $||\widehat{\Pi}\widehat{\mathcal{T}}v - \widehat{\Pi}v||_n$. This is a consequence of Lemma 5 applied to the vectors $Y = \widehat{\mathcal{T}}v$ and $Z = v$. Since $v$ is the value function at the points $\{X_t\}_{t=1}^n$, from the definition of the pathwise Bellman operator, we have that for $1 \leq t \leq n-1$,

$$\xi_t = y_t - v_t = r(X_t) + \gamma V(X_{t+1}) - V(X_t) = \gamma\big[V(X_{t+1}) - \int P(dy|X_t)V(y)\big],$$

and $\xi_n = y_n - v_n = -\gamma \int P(dy|X_n)V(y)$. Thus, Eq. 2.2 holds for $1 \leq t \leq n-1$. Here we may choose $C = 2\gamma V_{\max}$ for a bound on $\xi_t,\ 1 \leq t \leq n-1$, and $C = \gamma V_{\max}$ for a bound on $\xi_n$. Azuma's inequality may be applied only to the sequence of $n-1$ terms (the $n$-th term adds a contribution to the bound), thus instead of Eq. 2.7, we obtain

$$\Big|\sum_{t=1}^{n}\xi_t\varphi_i(X_t)\Big| \le \gamma V_{\max} L\big(2\sqrt{2n\log(2d/\delta)}+1\big),$$

with probability $1-\delta$, for all $1 \le i \le d$. Combining with Eq. 2.6, we deduce that with probability $1-\delta$, we have

$$||\widehat{\Pi}\widehat{\mathcal{T}}v - \widehat{\Pi}v||_n \le \gamma V_{\max} L\sqrt{\frac{d}{\nu_n}}\Big(\sqrt{\frac{8\log(2d/\delta)}{n}}+\frac{1}{n}\Big), \tag{2.11}$$

where $\nu_n$ is the smallest strictly-positive eigenvalue of $\frac{1}{n}\Phi^\top\Phi$. The claim follows by solving Eq. 2.10 for $||v-\hat{v}||_n$ and replacing $||\widehat{\Pi}\widehat{\mathcal{T}}v - \widehat{\Pi}v||_n$ from Eq. 2.11. ■

## 2.5 Generalization Bounds

As we pointed out earlier, Theorem 3 makes no assumption on the existence of the stationary distribution of the Markov chain. This generality comes at the cost that the performance is evaluated only at the states visited by the Markov chain and no generalization on other states is possible. However in many problems of interest, the Markov chain has a stationary distribution $\rho$, and thus, the performance may be generalized to the whole state space under the measure $\rho$. Moreover, if $\rho$ exists, it is possible to derive a condition for the existence of the pathwise LSTD solution depending on the number of samples and the smallest eigenvalue of the Gram matrix defined according to $\rho$ ; $G \in \mathbb{R}^{d\times d}$ , $G_{ij} = \int \varphi_i(x)\varphi_j(x)\rho(dx)$. In this section, we assume that the Markov chain $\mathcal{M}^\pi$ is exponentially fast $\beta$-mixing with parameters $\bar{\beta}, b, \kappa$, i.e., its $\beta$-mixing coefficients satisfy $\beta_i \le \bar{\beta}\exp(-bi^\kappa)$ (see Section 2.8.2 in the appendix for a more detailed definition of $\beta$-mixing processes).

Before stating the main results of this section, we introduce some notation. If $\rho$ is the stationary distribution of the Markov chain, we define the orthogonal projection operator $\Pi : \mathcal{B}(\mathcal{X}; V_{\max}) \to \mathcal{F}$ as

$$\Pi V = \arg\min_{f \in \mathcal{F}} ||V - f||_\rho \, .$$

Furthermore, in the rest of the paper with a little abuse of notation, we replace the empirical norm $||v||_n$ defined on states $X_1, \ldots, X_n$ by $||V||_n$, where $V \in \mathcal{B}(\mathcal{X}; V_{\max})$ is such that $V(X_t) = v_t$. Finally, we should guarantee that the pathwise LSTD solution $\widehat{V}$ is uniformly bounded on $\mathcal{X}$. For this reason, we move from $\mathcal{F}$ to the truncated space $\widetilde{\mathcal{F}}$ in which for any function $f \in \mathcal{F}$, a truncated function $\tilde{f}$ is defined as

$$\tilde{f}(x) = \begin{cases} f(x) & \text{if } |f(x)| \le V_{\max} \, , \\ \operatorname{sgn}\big(f(x)\big) V_{\max} & \text{otherwise.} \end{cases} \tag{2.12}$$

In the next sections, we present conditions on the existence of the pathwise LSTD solution and derive generalization bounds under different assumptions on the way the samples $X_1, \ldots, X_n$ are generated.

### 2.5.1 Uniqueness of Pathwise LSTD Solution

In this section, we assume that all the eigenvalues of $G$ are strictly positive; that is, we assume the existence of the model-based solution of LSTD, and derive a condition to guarantee that the sample-based Gram matrix $\frac{1}{n}\Phi^\top\Phi$ is invertible. More specifically, we show that if a large enough number of samples (depending on the smallest eigenvalue of $G$) is available, then the smallest eigenvalue of $\frac{1}{n}\Phi^\top\Phi$ is strictly positive with high probability.

**Lemma 6** *Let $G$ be the Gram matrix defined according to the distribution $\rho$ and $\omega > 0$ be its smallest eigenvalue. Let $X_1, \ldots, X_n$ be a trajectory of length $n$ of a stationary $\beta$-mixing process with parameters $\bar{\beta}, b, \kappa$ and stationary* *distribution $\rho$. If the number of samples $n$ satisfies the following condition*

$$n > \frac{288L^2\Lambda(n,d,\delta)}{\omega} \max\left\{\frac{\Lambda(n,d,\delta)}{b}, 1\right\}^{1/\kappa}, \tag{2.13}$$

*where*[2] $\Lambda(n,d,\delta) = 2(d+1)\log n + \log\frac{e}{\delta} + \log^+\left(\max\{18(6e)^{2(d+1)}, \bar{\beta}\}\right)$*, then with probability $1-\delta$, the family of features $(\varphi_1, \dots, \varphi_d)$ is linearly independent on the states $X_1, \dots, X_n$ (i.e., $||f_\alpha||_n = 0$ implies $\alpha = 0$) and the smallest eigenvalue $\nu_n$ of the sample-based Gram matrix $\frac{1}{n}\Phi^\top\Phi$ satisfies*

$$\sqrt{\nu_n} \geq \sqrt{\nu} = \frac{\sqrt{\omega}}{2} - 6L\sqrt{\frac{2\Lambda(n,d,\delta)}{n}\max\left\{\frac{\Lambda(n,d,\delta)}{b}, 1\right\}^{1/\kappa}} > 0\ . \tag{2.14}$$

**Proof** From the definition of the Gram matrix and the fact that $\omega > 0$ is its smallest eigenvalue, for any function $f_\alpha \in \mathcal{F}$, we have

$$||f_\alpha||_\rho^2 = ||\phi^\top\alpha||_\rho^2 = \alpha^\top G\alpha \geq \omega\alpha^\top\alpha = \omega||\alpha||^2. \tag{2.15}$$

Using the concentration inequality from Corollary 20 in the appendix and the fact that the basis functions $\varphi_i$ are bounded by $L$, thus $f_\alpha$ is bounded by $L||\alpha||$, we have $||f_\alpha||_\rho - 2||f_\alpha||_n \leq \epsilon$ with probability $1-\delta$, where

$$\epsilon = 12L||\alpha||\sqrt{\frac{2\Lambda(n,d,\delta)}{n}\max\left\{\frac{\Lambda(n,d,\delta)}{b}, 1\right\}^{1/\kappa}}\ .$$

Thus we obtain

$$2||f_\alpha||_n + \epsilon \geq \sqrt{\omega}||\alpha||. \tag{2.16}$$

Let $\alpha$ be such that $||f_\alpha||_n = 0$, then if the number of samples $n$ satisfies the condition of Eq. 2.13, we may deduce from Eq. 2.16 and the definition of $\epsilon$ that $\alpha = 0$. This indicates that given Eq. 2.13, with probability $1-\delta$, the family of features $(\varphi_1, \dots, \varphi_d)$ is linearly independent on the states $X_1, \dots, X_n$, and

[2] We define $\log^+ x = \max\{\log x, 0\}$.

thus, $\nu_n > 0$. The inequality in Eq. 2.14 is obtained by choosing $\alpha$ to be the eigenvector of $\frac{1}{n}\Phi^\top\Phi$ corresponding to the smallest eigenvalue $\nu_n$. For this value of $\alpha$, we have $||f_\alpha||_n = \sqrt{\nu_n}||\alpha||$. By using the definition of $\epsilon$ in Eq. 2.16 and reordering we obtain

$$2\sqrt{\nu_n}||\alpha|| + 12L||\alpha||\sqrt{\frac{2\Lambda(n,d,\delta)}{n}\max\left\{\frac{\Lambda(n,d,\delta)}{b},1\right\}^{1/\kappa}} \geq \sqrt{\omega}||\alpha||\ ,$$

and the claim follows. ■

**Remark 1** In order to make the condition on the number of samples and its dependency on the critical parameters of the problem at hand more explicit, let us consider the case of a stationary process with $b = \beta = \kappa = 1$. Then the condition in Eq. 2.13 becomes (up to constant and logarithmic factors)

$$n \geq \widetilde{O}\left(\frac{288L^2}{\omega}\left((d+1)\log\frac{n}{\delta}\right)^2\right).$$

As can be seen, the number of samples needed to have strictly positive eigenvalues in the sample-based Gram matrix has an inverse dependency on the smallest eigenvalue of $G$. As a consequence, the more $G$ is ill-conditioned the more samples are needed for the sample-based Gram matrix $\frac{1}{n}\Phi^\top\Phi$ to be invertible.

### 2.5.2 Generalization Bounds for Stationary $\beta$-mixing Processes

In this section, we show how Theorem 3 may be generalized to the entire state space $\mathcal{X}$ when the Markov chain $\mathcal{M}^\pi$ has a stationary distribution $\rho$. In particular, we consider the case in which the samples $X_1, \ldots, X_n$ are obtained by following a single trajectory in the stationary regime of $\mathcal{M}^\pi$, i.e., when we consider that $X_1$ is drawn from $\rho$.

**Theorem 7** *Let $X_1, \ldots, X_n$ be a path generated by a stationary $\beta$-mixing process with parameters $\bar{\beta}, b, \kappa$ and stationary distribution $\rho$. Let $\omega > 0$ be the smallest eigenvalue of the Gram matrix defined according to $\rho$ and $n$ satisfy the condition in Eq. 2.13. Let $\widetilde{V}$ be the truncation (using Eq. 2.12) of the pathwise LSTD solution, then*

$$||\widetilde{V}-V||_\rho \leq \frac{2}{\sqrt{1-\gamma^2}}\Big(2\sqrt{2}||V-\Pi V||_\rho+\varepsilon_2\Big)+\frac{2}{1-\gamma}\left[\gamma V_{\max} L\sqrt{\frac{d}{\nu}}\Big(\sqrt{\frac{8\log(8d/\delta)}{n}}+\frac{1}{n}\Big)\right]+\varepsilon_1 \tag{2.17}$$

*with probability $1-\delta$, where $\nu$ is a lower-bound on the eigenvalues of the sample-based Gram matrix defined by Eq. 2.14,*

$$\varepsilon_1 = 24V_{\max}\sqrt{\frac{2\Lambda_1(n,d,\delta/4)}{n}\max\left\{\frac{\Lambda_1(n,d,\delta/4)}{b},1\right\}^{1/\kappa}},$$

*with $\Lambda_1(n,d,\delta/4) = 2(d+1)\log n + \log\frac{4e}{\delta} + \log^+\big(\max\{18(6e)^{2(d+1)}, \bar{\beta}\}\big)$, and*

$$\varepsilon_2 = 12\big(V_{\max} + L||\alpha^*||\big)\sqrt{\frac{2\Lambda_2(n,\delta/4)}{n}\max\left\{\frac{\Lambda_2(n,\delta/4)}{b},1\right\}^{1/\kappa}}, \tag{2.18}$$

*with $\Lambda_2(n,\delta/4) = \log\frac{4e}{\delta} + \log\big(\max\{6, n\bar{\beta}\}\big)$ and $\alpha^*$ is such that $f_{\alpha^*} = \Pi V$.*

**Proof** This result is a consequence of applying generalization bounds to both sides of Eq. 2.1 (Theorem 3). We first bound the left-hand side:

$$2||\widehat{V}-V||_n \geq 2||\widetilde{V}-V||_n \geq ||\widetilde{V}-V||_\rho - \varepsilon_1$$

with probability $1-\delta'$. The first step follows from the definition of the truncation operator, while the second step is a straightforward application of Corollary 19 in the appendix.

We now bound the term $||V - \widehat{\Pi}V||_n$ in Eq. 2.1:

$$||V - \widehat{\Pi}V||_n \leq ||V - \Pi V||_n \leq 2\sqrt{2}||V - \Pi V||_\rho + \varepsilon_2$$

with probability $1 - \delta'$. The first step follows from the definition of the operator $\widehat{\Pi}$. The second step is an application of the inequality of Corollary 21 in the appendix for the function $V - \Pi V$.

From Theorem 3, the two generalization bounds, and the lower-bound on $\nu$, each one holding with probability $1 - \delta'$, the statement of the Theorem (Eq. 2.17) holds with probability $1 - \delta$ by setting $\delta = 4\delta'$. ■

**Remark 1** Rewriting the bound in terms of the approximation and estimation error terms (up to constants and logarithmic factors), we obtain

$$||\widetilde{V} - V||_\rho \leq \widetilde{O}\left(\frac{1}{\sqrt{1-\gamma^2}}||V - \Pi V||_\rho + \frac{1}{1-\gamma}\frac{1}{\sqrt{n}}\right).$$

While the first term (*approximation error*) only depends on the target function $V$ and the function space $\mathcal{F}$, the second term (*estimation error*) primarily depends on the number of samples. Thus, when $n$ goes to infinity, the estimation error goes to zero and we obtain the same performance bound (up to a $4\sqrt{2}$ constant) as for the model-based case reported by Tsitsiklis and Van Roy [1997]. The additional multiplicative constant $4\sqrt{2}$ in front of the approximation error is the standard cost to have the improved rate bounds for the squared loss and linear spaces (see e.g., Györfi et al., 2002). In fact, it is possible to derive a bounds with constant 1 but a worse rate $n^{-1/4}$ instead of $n^{-1/2}$. The bound in Theorem 7 is more accurate whenever the approximation error is small and few samples are available.

**Remark 2** Antos et al. [2008] reported a sample-based analysis for the modified Bellman residual (MBR) minimization algorithm. They consider a general setting in which the function space $\mathcal{F}$ is bounded and the performance of the algorithm is evaluated according to an arbitrary measure $\mu$ (possibly different than the stationary distribution of the Markov chain $\rho$). Since Antos et al. [2008] showed that the MBR minimization algorithm is equivalent to LSTD when $\mathcal{F}$ is a linearly parameterized space, it would be interesting to compare the bound in Theorem 7 to the one in Lemma 11 of Antos et al. [2008]. In Theorem 7, similar to Antos et al. [2008], samples are drawn from a stationary $\beta$-mixing process, however, $\mathcal{F}$ is a linear space and $\rho$ is the stationary distribution of the Markov chain. It is interesting to note the impact of these two differences in the final bound. The use of linear spaces has a direct effect on the estimation error and leads to a better convergence rate due to the use of improved functional concentration inequalities (Lemma 18 in the appendix). In fact, while in Antos et al. [2008] the estimation error for the squared error is of order $O(1/\sqrt{n})$, here we achieve a faster convergence rate of order $O(1/n)$. Moreover, although Antos et al. [2008] showed that the solution of MBR minimization coincides with the LSTD solution, its sample-based analysis cannot be directly applied to LSTD. In fact, in Antos et al. [2008] the function space $\mathcal{F}$ is assumed to be bounded, while general linear spaces cannot be bounded. Whether the analysis of Antos et al. [2008] may be extended to the truncated solution of LSTD is an open question that requires further investigation.

### 2.5.3 Generalization Bounds for Markov Chains

The main assumption in the previous section is that the trajectory $X_1, \dots, X_n$ is generated by a stationary $\beta$-mixing process with stationary distribution $\rho$. This is possible if we consider samples of a Markov chain during its stationary regime, i.e., $X_1 \sim \rho$. However in practice, $\rho$ is not known, and the first sample $X_1$ is usually drawn from a given initial distribution and the rest of the sequence is obtained by following the Markov chain from $X_1$ on. As a result, the sequence $X_1, \dots, X_n$ is no longer a realization of a stationary $\beta$-mixing process. Nonetheless, under suitable conditions, after $\tilde{n} < n$ steps, the distribution of $X_{\tilde{n}}$ approaches the stationary distribution $\rho$. In fact, according to the convergence theorem for fast-mixing Markov chains (see e.g., Proposition 22 in the appendix), for any initial distribution $\lambda \in \mathcal{S}(\mathcal{X})$, we have

$$\left|\left| \int_{\mathcal{X}} \lambda(dx) P^n(\cdot|x) - \rho(\cdot) \right|\right|_{TV} \leq \bar{\beta} \exp(-bn^{\kappa}).$$

where $|| \cdot ||_{TV}$ is the total variation.[3]

We now derive a bound for a modification of pathwise LSTD in which the first $\tilde{n}$ samples (that are used to burn the chain) are discarded and the remaining $n - \tilde{n}$ samples are used as training samples for the algorithm.

**Theorem 8** *Let $X_1, \dots, X_n$ be a trajectory generated by a $\beta$-mixing Markov chain with parameters $\bar{\beta}, b, \kappa$ and stationary distribution $\rho$. Let $\tilde{n}$ ($1 \leq \tilde{n} < n$) be such that $n - \tilde{n}$ satisfies the condition of Eq. 2.13, and $X_{\tilde{n}+1}, \dots, X_n$ be the samples actually used by the algorithm. Let $\omega > 0$ be the smallest eigenvalue of the Gram matrix defined according to $\rho$ and $\alpha^* \in \mathbb{R}^d$ be such that $f_{\alpha^*} = \Pi V$. Let $\widetilde{V}$ be the truncation of the pathwise LSTD solution (using Eq. 2.12), then by setting $\tilde{n} = \left(\frac{1}{b} \log \frac{2e\bar{\beta}n}{\delta}\right)^{1/\kappa}$, with probability $1 - \delta$, we have*

$$||\widetilde{V} - V||_{\rho} \leq \frac{2}{\sqrt{1-\gamma^2}} \Big(2\sqrt{2}||V - \Pi V||_{\rho} + \varepsilon_2\Big) + \frac{2}{1-\gamma} \left[\gamma V_{\max} L \sqrt{\frac{d}{\nu}} \Big(\sqrt{\frac{8 \log(8d/\delta)}{n - \tilde{n}}} + \frac{1}{\tilde{n}}\Big)\right] + \varepsilon_1, \tag{2.19}$$

*where $\varepsilon_1$ and $\varepsilon_2$ are defined as in Theorem 7 (with $n - \tilde{n}$ as the number of training samples).*

The proof of this result is a simple consequence of Lemma 26 in the appendix applied to Theorem 7.

[3] We recall that for any two distributions $\mu_1, \mu_2 \in \mathcal{S}(\mathcal{X})$, the total variation norm is defined as $||\mu_1 - \mu_2||_{TV} = \sup_{X \subseteq \mathcal{X}} |\mu_1(X) - \mu_2(X)|$.

**Remark 1** The bound in Eq. 2.19 indicates that in the case of $\beta$-mixing Markov chains, a similar performance to the one for stationary $\beta$-mixing processes is obtained by discarding the first $\tilde{n} = O(\log n)$ samples.

## 2.6 Finite-Sample Analysis of LSPI

In the previous sections we studied the performance of pathwise-LSTD for policy evaluation. Now we move to the analysis of the least-squares policy iteration (LSPI) algorithm [Lagoudakis and Parr, 2003a] in which at each iteration $k$ samples are collected by following a single trajectory of the policy under evaluation, $\pi_k$, and LSTD is used to compute an approximation of $V^{\pi_k}$. In particular, in the next section we report a performance bound by comparing the value of the policy returned by the algorithm after $K$ iterations, $V^{\pi_K}$, and the optimal value function, $V^*$, w.r.t. an arbitrary target distribution $\sigma$. In order to achieve this bound we introduce assumptions on the MDP and the linear space $\mathcal{F}$. In Section 2.6.2 we show that in some cases one of these assumptions does not hold and the performance of LSPI can be arbitrarily bad.

### 2.6.1 Generalization Bound for LSPI

In this section, we provide a performance bound for the LSPI algorithm [Lagoudakis and Parr, 2003a]. We first introduce the *greedy policy* operator $\mathcal{G}$ that maps value functions to their corresponding greedy policies:

$$\big(\mathcal{G}(V)\big)(x) = \arg\max_{a\in\mathcal{A}} \left[ r(x,a) + \gamma \int_{\mathcal{X}} P(dy|x,a)V(y) \right].$$

We use $\mathcal{G}(\mathcal{F})$ to refer to the set of all the greedy policies w.r.t. the functions in $\mathcal{F}$. LSPI is a policy iteration algorithm that uses LSTD for policy evaluation at each iteration. It starts with an arbitrary initial value function $V_{-1} \in \widetilde{\mathcal{F}}$ and its corresponding greedy policy $\pi_0$. At the first iteration, it approximates

$V^{\pi_0}$ using LSTD and returns a function $V_0$ whose truncated version $\widetilde{V}_0$ is used to build the policy $\pi_1$ for the second iteration.[4] More precisely, $\pi_1$ is the greedy policy w.r.t. $\widetilde{V}_0$, i.e., $\pi_1 = \mathcal{G}(\widetilde{V}_0)$. So, at each iteration $k$ of LSPI, a function $V_{k-1}$ is computed as an approximation to $V^{\pi_{k-1}}$, and then truncated, $\widetilde{V}_{k-1}$, and used to build the policy $\pi_k = \mathcal{G}(\widetilde{V}_{k-1})$. Note that the MDP model is needed in order to generate the greedy policy $\pi_k$. To avoid the need for the model, we could simply move from LSTD to LSTD-Q. The analysis of LSTD in the previous sections may be easily extended to action-value function, and thus, to LSTD-Q.[5] For simplicity we use value function in the paper and report the LSPI bound in terms of the distance to the optimal value function.

It is important to note that in general the measure used to evaluate the final performance of LSPI, $\sigma \in \mathcal{S}(\mathcal{X})$, might be different than the distribution used to generate the samples at each iteration. Moreover, the LSTD performance bounds of Section 2.5 require the samples to be collected by following the policy under evaluation. Thus, we make the following assumption.

**Assumption 1** (Lower-bounding distribution) *There exists a distribution $\mu \in \mathcal{S}(\mathcal{X})$ such that for any policy $\pi$ that is greedy w.r.t. a function in the truncated space $\widetilde{\mathcal{F}}$, $\mu \leq C\rho^{\pi}$, where $C < \infty$ is a constant and $\rho^{\pi}$ is the stationary distribution of policy $\pi$.*

**Assumption 2** . (Discounted-average Concentrability of Future-State Distribution [Antos et al., 2008]) *Given the target distribution $\sigma \in \mathcal{S}(\mathcal{X})$ and an*

[4] Unlike in the original formulation of LSPI, here we need to explicitly truncate the function so as to prevent unbounded functions.

[5] We point out that moving to LSTD-Q requires the introduction of some exploration to the current policy. In fact, in the on-policy setting, if the policy under evaluation is deterministic, it does not provide any information about the value of actions $a \neq \pi(\cdot)$ and the policy improvement step would always fail. Thus, we need to consider stochastic policies where the current policy is perturbed by an $\epsilon > 0$ randomization which guarantees that any action has a non-zero probability to be selected in any state.

*arbitrary sequence of policies* $\{\pi_m\}_{m\geq 1}$*, let*

$$c_{\sigma,\mu} = \sup_{\pi_1,\ldots,\pi_m} \left|\left|\frac{d(\mu P^{\pi_1}\ldots P^{\pi_m})}{d\sigma}\right|\right|.$$

*We define the second-order discounted-average concentrability of future-state distributions as*

$$C_{\sigma,\mu} = (1-\gamma)^2 \sum_{m\geq 1} m\gamma^{m-1} c_{\sigma,\mu}(m)$$

*and we assume that* $C_{\sigma,\mu} < \infty$*.*

We also need to guarantee that with high probability a unique LSTD solution exists at each iteration of the LSPI algorithm, thus, we make the following assumption.

**Assumption 3** (Linear independent features) *Let* $\mu \in \mathcal{S}(\mathcal{X})$ *be the lower-bounding distribution from Assumption 1. We assume that the features* $\phi(\cdot)$ *of the function space* $\mathcal{F}$ *are linearly independent w.r.t.* $\mu$*. In this case, the smallest eigenvalue* $\omega_\mu$ *of the Gram matrix* $G_\mu \in \mathbb{R}^{d\times d}$ *w.r.t.* $\mu$ *is strictly positive.*

**Lemma 9** *Under Assumption 3, at each iteration* $k$ *of LSPI, the smallest eigenvalue* $\omega_k$ *of the Gram matrix* $G_k$ *defined according to the stationary distribution* $\rho_k = \rho^{\pi_k}$ *is strictly positive and* $\omega_k \geq \frac{\omega_\mu}{C}$*.*

**Proof** Similar to Lemma 6, for any function $f_\alpha \in \mathcal{F}$, we have $||\alpha|| \leq \frac{||f_\alpha||_\mu}{\sqrt{\omega_\mu}}$. Using Assumption 1, $||f_\alpha||_\mu \leq \sqrt{C}\ ||f_\alpha||_{\rho_k}$, and thus, $||\alpha|| \leq \sqrt{\frac{C}{\omega_\mu}}\ ||f_\alpha||_{\rho_k}$. For the $\alpha$ that is the eigenvector of $G_k$ corresponding to $\rho_k$, we have $||\alpha|| = \frac{||f_\alpha||_{\rho_k}}{\sqrt{\omega_k}}$. For this value of $\alpha$, we may write $\frac{||f_\alpha||_{\rho_k}}{\sqrt{\omega_k}} \leq \sqrt{\frac{C}{\omega_\mu}}\ ||f_\alpha||_{\rho_k}$, and thus, $\omega_k \geq \frac{\omega_\mu}{C}$, which guarantees that $\omega_k$ is strictly positive, because $\omega_\mu$ is strictly positive according to Assumption 3. ■

Finally, we make the following assumption on the stationary $\beta$-mixing processes corresponding to the stationary distributions of the policies encountered at the iterations of the LSPI algorithm.

**Assumption 4** (Slower $\beta$-mixing process) *We assume that there exists a stationary $\beta$-mixing process with parameters $\bar{\beta}, b, \kappa$, such that for any policy $\pi$ that is greedy w.r.t. a function in the truncated space $\widetilde{\mathcal{F}}$, it is slower than the stationary $\beta$-mixing process with stationary distribution $\rho^{\pi}$ (with parameters $\bar{\beta}_{\pi}, b_{\pi}, \kappa_{\pi}$). This means that $\bar{\beta}$ is larger and $b$ and $\kappa$ are smaller than their counterparts $\bar{\beta}_{\pi}$, $b_{\pi}$, and $\kappa_{\pi}$ (see Definition 16).*

Now we may state the main theorem of this section.

**Theorem 10** *Let us assume that at each iteration $k$ of the LSPI algorithm, a path of size $n$ is generated from the stationary $\beta$-mixing process with stationary distribution $\rho_{k-1} = \rho^{\pi_{k-1}}$. Let $n$ satisfy the condition in Eq. 2.13 for the slower $\beta$-mixing process defined in Assumption 4. Let $V_{-1} \in \widetilde{\mathcal{F}}$ be an arbitrary initial value function, $V_0, \ldots, V_{K-1}$ ($\widetilde{V}_0, \ldots, \widetilde{V}_{K-1}$) be the sequence of value functions (truncated value functions) generated by LSPI after $K$ iterations, and $\pi_K$ be the greedy policy w.r.t. the truncated value function $\widetilde{V}_{K-1}$. Then under Assumptions 1- 4, with probability $1-\delta$ (w.r.t. the random samples), we have*

$$||V^* - V^{\pi_K}||_{\sigma} \leq \frac{4\gamma}{(1-\gamma)^2}\Bigg\{(1+\gamma)\sqrt{CC_{\sigma,\mu}}\Bigg[\frac{2}{\sqrt{1-\gamma^2}}\Big(2\sqrt{2}E_0(\mathcal{F}) + E_2\Big) + \frac{2}{1-\gamma}\Big(\gamma V_{\max} L\sqrt{\frac{d}{\nu_{\mu}}}\big(\sqrt{\frac{8\log(8dK/\delta)}{n}} + \frac{1}{n}\big)\Big) + E_1\Bigg] + \gamma^{\frac{K-1}{2}} R_{\max}\Bigg\} ,$$

*where*

1. *$E_0(\mathcal{F}) = \sup_{\pi \in \mathcal{G}(\widetilde{\mathcal{F}})} \inf_{f \in \mathcal{F}} ||f - V^{\pi}||_{\rho^{\pi}}$,*
2. *$E_1$ is $\varepsilon_1$ from Theorem 7 written for the slower $\beta$-mixing process defined in Assumption 4,*

3. *$E_2$ is $\varepsilon_2$ from Theorem 7 written for the slower $\beta$-mixing process defined in Assumption 4 and $||\alpha^*||$ replaced by $\sqrt{\frac{C}{\omega_\mu}}\frac{R_{\max}}{1-\gamma}$, and*

4. *$\nu_\mu$ is $\nu$ from Eq. 2.14 in which $\omega$ is replaced by $\omega_\mu$ defined in Assumption 3, and the second term is written for the slower $\beta$-mixing process defined in Assumption 4.*

**Remark 1** The previous theorem states a bound on the prediction error when LSPI is stopped after a fixed number $K$ of iterations. The structure of the bound resembles the one in Antos et al. [2008]. Unlike policy evaluation, the approximation error $E_0(\mathcal{F})$ now depends on how well the space $\mathcal{F}$ can approximate the target functions $V^\pi$ obtained in the policy improvement step. While the estimation errors are mostly similar to those in policy evaluation, an additional term of order $\gamma^K$ is introduced. Finally, we notice that the concentrability terms may significantly amplify the prediction error (see also next remark). Farahmand et al. [2010] recently performed a refined analysis of the propagation of the error in approximate policy iteration and have interesting insights on the concentrability terms.

**Remark 2** The most critical issue about Theorem 10 is the validity of Assumptions 1–4. The analysis of LSTD explicitly requires that the samples are collected by following the policy under evaluation, $\pi_k$, and the performance is bounded according to its stationary distribution $\rho_k$. Since the performance of LSPI is assessed w.r.t. a target distribution $\sigma$, we need each of the policies encountered through the LSPI process to have a stationary distribution which does not differ too much from $\sigma$. Furthermore, since the policies are random (at each iteration $k$ the new policy $\pi_k$ is greedy w.r.t. the approximation $\widetilde{V}_{k-1}$ which is random because of the sampled trajectory), we need to consider the distance of $\sigma$ and the stationary distribution of any possible policy generated as greedy w.r.t. a function in the truncated space $\widetilde{\mathcal{F}}$, i.e., $\rho^\pi,\ \pi \in \mathcal{G}(\widetilde{\mathcal{F}})$. Thus in Assumption 1 we first assume the existence of a distribution $\mu$ lower-bounding any possible stationary distribution $\rho_k$. The existence of $\mu$ and the value of the constant $C$ depend on the MDP at hand. In Section 2.6.2, we provide an example in which the constant $C$ is infinite. In this case, we show that the LSPI performance, when the samples at each iteration are generated according to the stationary distribution of the policy under evaluation, can be arbitrarily bad. A natural way to relax this assumption would be the use of off-policy LSTD in which the samples are collected by following a behavior policy. Nonetheless, we are not aware of any finite-sample analysis for such an algorithm. Another critical term appearing in the bound of LSPI, inherited from Theorem 7, is the maximum of $||\alpha_k^*||$ over the iterations, where $\alpha_k^*$ is such that $f_{\alpha_k^*} = \Pi_{\rho_k} V^{\pi_k}$. Each term $||\alpha_k^*||$ can be bounded whenever the features of the space $\mathcal{F}$ are linearly independent according to the stationary distribution $\rho_k$. Since $\alpha_k^*$ is a random variable, the features $\{\varphi_i\}_{i=1}^d$ of the space $\mathcal{F}$ should be carefully chosen so as to be linearly independent w.r.t. the lower-bounding distribution $\mu$.

We now prove a lemma that is used in the proof of Theorem 10.

**Lemma 11** *Let $\pi_k$ be the greedy policy w.r.t. $\widetilde{V}_{k-1}$, i.e., $\pi_k = \mathcal{G}(\widetilde{V}_{k-1})$ and $\rho^{\pi_k}$ be the stationary distribution of the Markov chain induced by $\pi_k$. We have*

$$||\widetilde{V}_k - \mathcal{T}^{\pi_k}\widetilde{V}_k||_{\rho^{\pi_k}} \leq (1+\gamma)||\widetilde{V}_k - V^{\pi_k}||_{\rho^{\pi_k}} \ .$$

**Proof** [Lemma 11] We first show that $\widetilde{V}_k - \mathcal{T}^{\pi_k}\widetilde{V}_k = (I - \gamma P^{\pi_k})(\widetilde{V}_k - V^{\pi_k})$

$$\begin{aligned}(I - \gamma P^{\pi_k})(\widetilde{V}_k - V^{\pi_k}) &= \widetilde{V}_k - V^{\pi_k} - \gamma P^{\pi_k}\widetilde{V}_k + \gamma P^{\pi_k} V^{\pi_k} = \widetilde{V}_k - V^{\pi_k} - \mathcal{T}^{\pi_k}\widetilde{V}_k + \mathcal{T}^{\pi_k} V^{\pi_k} \\ &= \widetilde{V}_k - V^{\pi_k} - \mathcal{T}^{\pi_k}\widetilde{V}_k + V^{\pi_k} = \widetilde{V}_k - \mathcal{T}^{\pi_k}\widetilde{V}_k \ .\end{aligned}$$

For any distribution $\sigma \in \mathcal{S}(\mathcal{X})$, we may write

$$\begin{aligned}||\widetilde{V}_k - \mathcal{T}^{\pi_k}\widetilde{V}_k||_\sigma &= ||(I - \gamma P^{\pi_k})(\widetilde{V}_k - V^{\pi_k})||_\sigma \leq ||I - \gamma P^{\pi_k}||_\sigma ||\widetilde{V}_k - V^{\pi_k}||_\sigma \\ &\leq \big(1 + \gamma ||P^{\pi_k}||_\sigma\big)||\widetilde{V}_k - V^{\pi_k}||_\sigma\end{aligned}$$

If $\sigma$ is the stationary distribution of $\pi_k$, i.e., $\sigma = \rho^{\pi_k}$, then $||P^{\pi_k}||_\sigma = 1$ and the claim follows. Note that this theorem holds not only for $\ell_2$-norm, but for any $\ell_p$-norm, $p \geq 1$. ■

**Proof** [Theorem 10] Rewriting Lemma 12 in Antos et al. [2008] for $V$ instead of $Q$, we obtain[6]

$$||V^* - V^{\pi_K}||_\sigma \leq \frac{4\gamma}{(1-\gamma)^2}\Big(\sqrt{C_{\sigma,\mu}} \max_{0 \leq k < K} ||\widetilde{V}_k - \mathcal{T}^{\pi_k}\widetilde{V}_k||_\mu + \gamma^{\frac{K-1}{2}} R_{\max}\Big). \quad (2.20)$$

From Assumption 1, we know that $|| \cdot ||_\mu \leq \sqrt{C}|| \cdot ||_{\rho_k}$ for any $0 \leq k < K$ and thus we may rewrite Eq. 2.20 as

$$||V^* - V^{\pi_K}||_\sigma \leq \frac{4\gamma}{(1-\gamma)^2}\Big(\sqrt{CC_{\sigma,\mu}} \max_{0 \leq k < K} ||\widetilde{V}_k - \mathcal{T}^{\pi_k}\widetilde{V}_k||_{\rho_k} + \gamma^{\frac{K-1}{2}} R_{\max}\Big). \quad (2.21)$$

Using the result of Lemma 11, Eq. 2.21 may be rewritten as

$$||V^* - V^{\pi_K}||_\sigma \leq \frac{4\gamma}{(1-\gamma)^2}\Big((1+\gamma)\sqrt{CC_{\sigma,\mu}} \max_{0 \leq k < K} ||\widetilde{V}_k - V^{\pi_k}||_{\rho_k} + \gamma^{\frac{K-1}{2}} R_{\max}\Big). \quad (2.22)$$

We can now use the result of Theorem 7 (which holds with probability $\delta/K$) and replace $||\widetilde{V}_k - V^{\pi_k}||_{\rho_k}$ with its upper-bound. The next step would be to apply the maximum over $k$ to this upper-bound (the right hand side of Eq. 2.17). There are four terms on the r.h.s. of Eq. 2.17 that depend on $k$ and in following we find a bound for each of them.

1. $||V^{\pi_k} - \Pi_{\rho_k} V^{\pi_k}||_{\rho_k}$: This term can be upper-bounded by $E_0(\mathcal{F})$. This

[6] The slight difference between Eq. 2.20 and the bound in Lemma 12 of Antos et al. [2008] is due to a small error in Eq. 26 of Antos et al. [2008]. It can be shown that the r.h.s. of Eq. 26 in Antos et al. [2008] is not an upper-bound for the r.h.s. of its previous equation. This can be easily fixed by redefining the coefficients $\alpha_k$ while we make sure that they remain positive and still sum to one. This modification causes two small changes in the final bound: the constant 2 in front of the parenthesis becomes 4 and the power of the $\gamma$ in front of $R_{\max}$ changes from $K/p$ to $(K-1)/p$.

quantity, $E_0(\mathcal{F})$, measures the approximation power of the linear function space $\mathcal{F}$.

2. $\varepsilon_1$: This term only depends on the parameters $\bar{\beta}_k, b_k, \kappa_k$ of the stationary $\beta$-mixing process with stationary distribution $\rho_k$. Using Assumption 4, this term can be upper-bounded by $E_1$, which is basically $\varepsilon_1$ written for the slower $\beta$-mixing process from Assumption 4.

3. $\varepsilon_2$: This term depends on the following $k$-related terms.

   - The term under the root-square in Eq. 2.18: This term depends on the parameters $\bar{\beta}_k, b_k, \kappa_k$ of the stationary $\beta$-mixing process with stationary distribution $\rho_k$. Similar to $\varepsilon_1$, this term can be upper-bounded by rewriting it for the slower $\beta$-mixing process from Assumption 4.

   - $\alpha_k^*$: The coefficient vector $\alpha_k^*$ is such that $f_{\alpha_k^*} = \Pi_{\rho_k} V^{\pi_k}$. This term can be upper-bounded as follows:

     $$\begin{aligned} ||\alpha_k^*|| &\overset{\text{(a)}}{\leq} \frac{||f_{\alpha^*}||_\mu}{\sqrt{\omega_\mu}} \overset{\text{(b)}}{\leq} \sqrt{\frac{C}{\omega_\mu}}||f_{\alpha^*}||_{\rho_k} = \sqrt{\frac{C}{\omega_\mu}}||\Pi_{\rho_k} V^{\pi_k}||_{\rho_k} \overset{\text{(c)}}{\leq} \sqrt{\frac{C}{\omega_\mu}}||V^{\pi_k}||_{\rho_k} \\ &\leq \sqrt{\frac{C}{\omega_\mu}}||V^{\pi_k}||_\infty = \sqrt{\frac{C}{\omega_\mu}}\, V_{\max} = \sqrt{\frac{C}{\omega_\mu}}\, \frac{R_{\max}}{1-\gamma} \; . \end{aligned}$$

     **(a)** Similar to Eq. 2.15, this is true for any function $f_\alpha \in \mathcal{F}$.
     **(b)** This is an immediate application of Assumption 1.
     **(c)** We use the fact that the orthogonal projection $\Pi_{\rho_k}$ is non-expansive for norm $|| \cdot ||_{\rho_k}$.

4. $\nu_{\rho_k}$: This term depends on the following $k$-related terms.

   - $\omega_k$: This is the smallest eigenvalue of the Gram matrix $G_k$ defined according to the distribution $\rho_k$. From Lemma 9, this term can be lower-bounded by $\omega_\mu$.

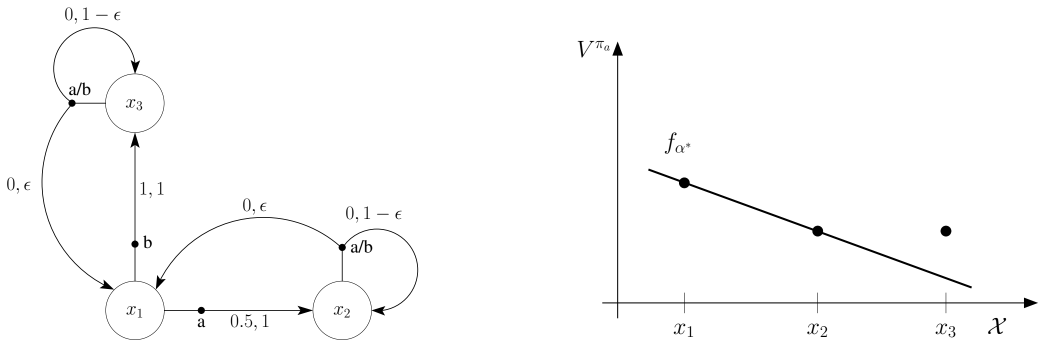


Figure 2.4: *(left)* The MDP used in the example of Section 2.6.2 and *(right)* the value function for policy $\pi_a$ in this MDP.

- The second term on the r.h.s. of Eq. 2.14: This term depends on the parameters $\bar{\beta}_k, b_k, \kappa_k$ of the stationary $\beta$-mixing process with stationary distribution $\rho_k$. Similar to $\varepsilon_1$ and $\varepsilon_2$, this term can be upper-bounded by rewriting it for the slower $\beta$-mixing process from Assumption 4.

By replacing the above lower and upper bounds in Eq. 2.14, we obtain $\nu_\mu$ which is a lower-bound for any $\nu_{\rho_k}$.

The claim follows by replacing the bounds for the above four terms in Eq. 2.22. ■

### 2.6.2 A Negative Result for LSPI

In the previous section we analyzed the performance of LSPI when at each iteration the samples are obtained from a trajectory generated by following the policy under evaluation. In order to bound the performance of LSPI in Theorem 10, we made a strong assumption on all possible stationary distributions that can be obtained at the iterations of the algorithm. Assumption 1 states the existence of a lower-bounding distribution $\mu$ for the stationary distribution $\rho^{\pi}$ of any policy $\pi \in \mathcal{G}(\widetilde{\mathcal{F}})$. If such a distribution does not exist ($C$ is infinite), the LSPI performance can no longer be bounded. In other words, this result states that in some MDPs, even if at each iteration the target function $V^{\pi_k}$ is perfectly approximated by $\widehat{V}_k$ under $\rho_k$-norm, i.e., $||V^{\pi_k} - \widehat{V}_k||_{\rho_k} = 0$, the LSPI performance could be arbitrarily bad. In this section we show a very simple MDP in which this is actually the case.

Let consider a finite MDP with $\mathcal{X} = \{x_1, x_2, x_3\}$, $\mathcal{A} = \{a, b\}$, and the reward function $r$ and transition model $p$ as illustrated in Figure 2.4. As it can be noticed only two policies are available in this MDP: $\pi_a$ which takes action $a$ in state $x_1$ and $\pi_b$ which takes action $b$ in this state. It is easy to verify that the stationary distribution $\rho^{\pi_a}$ assigns probabilities $\frac{\epsilon}{1+\epsilon}$, $\frac{1}{1+\epsilon}$, and 0 to $x_1$, $x_2$, and $x_3$, while $\rho^{\pi_b}$ has probabilities $\frac{\epsilon}{1+\epsilon}$, 0, and $\frac{1}{1+\epsilon}$. Since $\rho^{\pi_a}$ and $\rho^{\pi_b}$ assign a probability 0 to two different states, it is not possible to find a finite constant $C$ such that a distribution $\mu$ is lower-bounding both $\rho^{\pi_a}$ and $\rho^{\pi_b}$, thus, $C = \infty$ and according to Theorem 10 LSPI may have an arbitrary bad performance.

Let initialize LSPI with the suboptimal policy $\pi_a$. The value function $V^{\pi_a}$ is shown in Figure 2.4 (note that the specific values depend on the choice of $\epsilon$ and $\gamma$). Let $\mathcal{F} = \{f_\alpha(x) = \alpha_1 x + \alpha_2, \alpha \in \mathbb{R}^2\}$ be the space of lines in dimension 1. Let $\alpha^*$ be the solution to the following minimization problem $\alpha^* = \arg\inf_{\alpha\in\mathbb{R}} ||V^{\pi_a} - f_\alpha||^2_{\rho^{\pi_a}}$ (the projection of $V^{\pi_a}$ onto space $\mathcal{F}$). Since $\rho^{\pi_a}$ assigns a probability 0 to state $x_3$, the $f_{\alpha^*}$ in Figure 2.4 has a zero loss, i.e., $||V^{\pi_a} - f_{\alpha^*}||_{\rho^{\pi_a}} = 0$. Nonetheless, while the greedy policy w.r.t. $V^{\pi_a}$ is the optimal policy $\pi_b$, the policy improvement step w.r.t. $f_{\alpha^*}$ returns the policy $\pi_a$. As a result, although at each iteration the function space $\mathcal{F}$ may accurately approximate the value function of the current policy $\pi$ w.r.t. its stationary distribution $\rho^{\pi}$, LSPI never improves its performance and returns $\pi_a$ instead of the optimal policy $\pi_b$. By properly setting the rewards we could make the performance of $\pi_a$ arbitrarily worse than $\pi_b$.

## 2.7 Conclusions

In this paper we presented a finite-sample analysis of the least-squares policy iteration (LSPI) algorithm [Lagoudakis and Parr, 2003a]. This paper substantially extends the analysis in Lazaric et al. [2010c] by reporting all the lemmas used to prove the performance bounds of LSTD in the case of $\beta$-mixing and Markov chain processes and by analyzing how the performance of LSTD is propagated through iterations in LSPI.

More in detail, we first studied a version of LSTD, called pathwise LSTD, for policy evaluation. We considered a general setting where we do not make any assumption on the Markov chain. We derived an empirical performance bound that indicates how close the LSTD solution is to the value function at the states along a trajectory generated by following the policy and used by the algorithm. The bound is expressed in terms of the best possible approximation of the value function in the selected linear space (approximation error), and an estimation error which depends on the number of samples and the smallest strictly-positive eigenvalue of the sample-based Gram matrix. We then showed that when the Markov chain possesses a stationary distribution, one may deduce generalization performance bounds using the stationary distribution of the chain as the generalization measure. In particular, we considered two cases, where the sample trajectory is generated by stationary and non-stationary $\beta$-mixing Markov chains, and derived the corresponding bounds. Finally, we considered the whole policy iteration algorithm (LSPI) and showed that under suitable conditions it is possible to bound the error cumulated through the iterations.

The techniques used for the analysis of LSTD have also been recently employed for the development of the finite-sample analysis of a number of novel algorithms such as LSTD with random projections [Ghavamzadeh et al., 2010], LassoTD or LSTD with $\ell_1$ regularization [Ghavamzadeh et al., 2011], Classification-based Policy Iteration with a Critic [Gabillon et al., 2011b], and temporal-difference learning with Dantzig selector [Geist et al., 2012].

**Technical issues.** From a technical point of view there are two main open issues.

1. *Dependency on $\nu_n$ in the bound of Theorem 1.* In Section 2.4 we introduced the Markov design setting for regression in which the samples are obtained by following a Markov chain and the noise is a zero-mean martingale. By comparing the bound in Lemma 5 with the bounds for least-squares regression in deterministic design (see e.g., Theorem 11.1 in Györfi et al., 2002), the main difference is the inverse dependency on the eigenvalue $\nu_n$ of the empirical Gram matrix. It is not clear whether this dependency is intrinsic in the process generating the samples or whether it can be removed. Abbasi-Yadkori et al. [2011] recently developed improved Azuma's inequalities for self-normalizing process (see also e.g., de la Peña et al., 2007, de la Peña and Pang, 2009) which suggest that the bound can be improved by removing the dependency from $\nu_n$ and, thus, also from the $L_\infty$-norm $L$ of the features.

2. *The* $\log n$ *dependency in the generalization bounds.* Chaining techniques [Talagrand, 2005] can be successfully applied to remove the $\log n$ dependency in Pollard's inequalities for regression in bounded spaces. An interesting question is whether similar techniques can be applied to the refined analysis for squared losses and linear spaces (see e.g., Lemma 12) used in our theorems.

**Extensions.** Some extensions to the current work are possible.

1. *LSTD($\lambda$).* A popular improvement to LSTD is the use of eligibility traces, thus obtaining LSTD($\lambda$). The extension of the results presented in this paper to this setting does not seem to be straightforward since the regression problem solved in LSTD($\lambda$) does not match the Markov design setting introduced in Definition 4. Hence, it is an open question how a finite-sample analysis of LSTD($\lambda$) could be derived.

2. *Off-policy LSTD.* Yu and Bertsekas [2010] derived new bounds for projected linear equations substituting the $\frac{1}{\sqrt{1-\gamma^2}}$ term in front of the approximation error with a much sharper term depending on the spectral radius of some matrices defined by the problem. An open question is whether these new bounds can be effectively reused in the finite-sample analysis derived in this paper, thus obtaining much sharper bounds.

3. *Joint analysis of BRM and LSTD.* Scherrer [2010] recently proposed a unified view of Bellman residual minimization (BRM) [Schweitzer and Seidmann, 1985, Baird, 1995] and temporal difference methods through the notion of oblique projections. This suggests the possibility that the finite-sample analysis of LSTD could be extended to BRM through this unified view over the two methods.

## 2.8 Appendix

In this appendix we report a series of lemmata which are used throughout the paper. In particular, we derive concentration of measures inequalities for linear spaces and squared loss when samples are generated from different stochastic processes. We start with the traditional setting of independent and identically distributed samples in Section 2.8.1, then move to samples generated from mixing processes in Section 2.8.2, and finally consider the more general case of samples obtained by simulating a fast mixing Markov chain starting from an arbitrary distribution in Section 2.8.3.

As a general rule, we use *proposition* to indicate results which are copied from other sources, while *lemma* refers to completely or partially new results.

### 2.8.1 IID Samples

Although in the setting considered in the paper the samples are non-i.i.d., we first report functional concentration inequalities for i.i.d. samples which will be later extended to stationary and non-stationary $\beta$-mixing processes. We first recall the definition of expected and empirical $\ell_2$-norms for a function $f : \mathcal{X} \to \mathbb{R}$

$$\|f\|^2_{X_1^n} = \frac{1}{n}\sum_{t=1}^{n} |f(X_t)|^2 \qquad , \qquad \|f\|^2 = \mathbb{E}\left[|f(X_1)|^2\right].$$

**Lemma 12** *Let $\mathcal{F}$ be a class of functions $f : \mathcal{X} \to \mathbb{R}$ bounded in absolute value by $B$. Let $X_1^n = \{X_1, \dots, X_n\}$ be a sequence of i.i.d. samples. For any $\epsilon > 0$*

$$\mathbb{P}\left[\exists f \in \mathcal{F} : \|f\| - 2\|f\|_{X_1^n} > \epsilon\right] \le 3\mathbb{E}\left[\mathcal{N}_2\left(\frac{\sqrt{2}}{24}\epsilon, \mathcal{F}, X_1^{2n}\right)\right] \exp\left(-\frac{n\epsilon^2}{288B^2}\right),$$

*and*

$$\mathbb{P}\left[\exists f \in \mathcal{F} : \|f\|_{X_1^n} - 2\|f\| > \epsilon\right] \le 3\mathbb{E}\left[\mathcal{N}_2\left(\frac{\sqrt{2}}{24}\epsilon, \mathcal{F}, X_1^{2n}\right)\right] \exp\left(-\frac{n\epsilon^2}{288B^2}\right),$$

*where $\mathcal{N}_2(\epsilon, \mathcal{F}, X_1^n)$ is the $(L_2, \epsilon)$-cover number of the function space $\mathcal{F}$ on the samples $X_1^n$ (see Györfi et al. 2002).*

**Proof** The first statement is proved in Györfi et al. [2002] and the second one can be proved similarly. ■

**Proposition 13** *Let $\mathcal{F}$ be a class of linear functions $f : \mathcal{X} \to \mathbb{R}$ of dimension $d$ and $\widetilde{\mathcal{F}}$ be the class of functions obtained by truncating functions $f \in \mathcal{F}$ at a threshold $B$. Then for any sample $X_1^n = \{X_1, \dots, X_n\}$ and $\epsilon > 0$*

$$\mathcal{N}_2\left(\epsilon, \widetilde{\mathcal{F}}, X_1^n\right) \le 3\left(\frac{3e(2B)^2}{\epsilon^2}\right)^{2(d+1)}.$$

**Proof** Using Theorem 9.4. in Györfi et al. [2002] and the fact that the pseudo-dimension of $\widetilde{\mathcal{F}}$ is the same as $\mathcal{F}$, we have

$$\mathcal{N}_2\left(\epsilon, \widetilde{\mathcal{F}}, X_1^n\right) \leq 3\left(\frac{2e(2B)^2}{\epsilon^2}\log\frac{3e(2B)^2}{\epsilon^2}\right)^{d+1} \leq 3\left(\frac{3e(2B)^2}{\epsilon^2}\right)^{2(d+1)}.$$

■

We now use Proposition 13 to invert the bound in Lemma 12 for truncated linear spaces.

**Corollary 14** *Let $\mathcal{F}$ be a class of linear functions $f : \mathcal{X} \to \mathbb{R}$ of dimension $d$, $\widetilde{\mathcal{F}}$ be the class of functions obtained by truncating functions $f \in \mathcal{F}$ at a threshold $B$, and $X_1^n = \{X_1, \dots, X_n\}$ be a sequence of i.i.d. samples. By inverting the bound of Lemma 12, for any $\tilde{f} \in \widetilde{\mathcal{F}}$, we have*

$$||\tilde{f}|| - 2||\tilde{f}||_{X_1^n} \leq \epsilon(\delta), \qquad ||\tilde{f}||_{X_1^n} - 2||\tilde{f}|| \leq \epsilon(\delta),$$

*with probability $1-\delta$, where*

$$\epsilon(\delta) = 12B\sqrt{\frac{2\Lambda(n,d,\delta)}{n}}\,, \tag{2.23}$$

*and $\Lambda(n,d,\delta) = 2(d+1)\log n + \log\frac{e}{\delta} + \log\left(9(12e)^{2(d+1)}\right)$.*

**Proof** In order to prove the corollary it is sufficient to verify that the following inequality holds for the $\epsilon$ defined in Eq. 2.23

$$3\mathbb{E}\left[\mathcal{N}_2\left(\frac{\sqrt{2}}{24}\epsilon, \widetilde{\mathcal{F}}, X_1^{2n}\right)\right]\exp\left(-\frac{n\epsilon^2}{288B^2}\right) \leq \delta.$$

Using Proposition 13, we bound the first term as

$$\mathbb{E}\left[\mathcal{N}_2\left(\frac{\sqrt{2}}{24}\epsilon, \widetilde{\mathcal{F}}, X_1^{2n}\right)\right] \leq 3\left(\frac{C_1}{\epsilon^2}\right)^{2(d+1)},$$

with $C_1 = 3456eB^2$. Next we notice that $\Lambda(n,d,\delta) \geq 1$ and thus $\epsilon \geq \sqrt{1/(nC_2)}$ with $C_2 = (288B^2)^{-1}$. Using these bounds in the original inequality and some algebra we obtain

$$\begin{aligned}
3\mathbb{E}\left[\mathcal{N}_2\left(\frac{\sqrt{2}}{24}\epsilon, \widetilde{\mathcal{F}}, X_1^{2n}\right)\right]\exp\left(-\frac{n\epsilon^2}{288B^2}\right) &\leq 9\left(\frac{C_1}{\epsilon^2}\right)^{2(d+1)}\exp\left(-nC_2\epsilon^2\right)\\
&\leq 9\left(nC_1C_2\right)^{2(d+1)}\exp\left(-C_2 n\frac{\Lambda(n,d,\delta)}{nC_2}\right)\\
&= 9\left(nC_1C_2\right)^{2(d+1)} n^{-2(d+1)}\frac{\delta}{e}\frac{1}{9(C_1C_2)^{2(d+1)}}\\
&= \frac{\delta}{e} \leq \delta.
\end{aligned}$$

■

Non-functional versions of Corollary 14 can be simply obtained by removing the covering number from the statement of Lemma 12.

**Corollary 15** *Let $f : \mathcal{X} \to \mathbb{R}$ be a function bounded in absolute value by $B$ and $X_1^n = \{X_1, \ldots, X_n\}$ be a sequence of i.i.d. samples. Then*

$$\|f\| - 2\|f\|_{X_1^n} \leq \epsilon(\delta), \qquad \|f\|_{X_1^n} - 2\|f\| \leq \epsilon(\delta),$$

*with probability $1-\delta$, where*

$$\epsilon(\delta) = 12B\sqrt{\frac{2}{n}\log\frac{3}{\delta}}\,.$$

### 2.8.2 Stationary $\beta$-mixing Processes

We first introduce $\beta$-mixing stochastic processes and $\beta$-mixing coefficients.

**Definition 16** *Let $\{X_t\}_{t\geq 1}$ be a stochastic process. Let $X_i^j = \{X_i, X_{i+1}, \ldots, X_j\}$ and $\sigma(X_i^j)$ denote the sigma-algebra generated by $X_i^j$. The i-th $\beta$-mixing co-*

*efficient of the stochastic process is defined by*

$$\beta_i = \sup_{t \geq 1} \mathbb{E}\left[ \sup_{B \in \sigma(X_{t+i}^{\infty})} |\mathbb{P}(B|X_1^t) - \mathbb{P}(B)| \right].$$

*The process $\{X_t\}_{t\geq 1}$ is said to be $\beta$-mixing if $\beta_i \to 0$ as $i \to \infty$. In particular, $\{X_t\}_{t\geq 1}$ mixes at an exponential rate with parameters $\bar{\beta}, b, \kappa$ if $\beta_i \leq \bar{\beta} \exp(-bi^{\kappa})$. Finally, $\{X_t\}_{t\geq 1}$ is strictly stationary if $X_t \sim \nu$ for any $t > 0$.*

Let $X_1, \ldots, X_n$ be a sequence of samples drawn from a stationary $\beta$-mixing process with coefficients $\{\beta_i\}$. We first introduce the blocking technique of Yu [1994]. Let us divide the sequence of samples into blocks of size $k_n$. For simplicity we assume $n = 2m_n k_n$ with $2m_n$ be the number of blocks.[7] For any $1 \leq j \leq m_n$ we define the set of indexes in an odd and even block respectively as

$$H_j = \{t : 2(j-1)k_n + 1 \leq t \leq (2j-1)k_n\}, \quad \text{and}$$
$$E_j = \{t : (2j-1)k_n + 1 \leq t \leq (2j)k_n\}.$$

Let $H = \cup_{j=1}^{m_n} H_j$ and $E = \cup_{j=1}^{m_n} E_j$ be the set of all indexes in the odd and even blocks, respectively. We use $X(H_j) = \{X_t : t \in H_j\}$ and $X(H) = \{X_t : t \in H\}$. We now introduce a ghost sample $X'$ (the size of the ghost sample $X'$ is equal to the number of samples in each block $k_n$) in each of the odd blocks such that the joint distribution of $X'(H_j)$ is the same as $X(H_j)$ but independent from any other block. In the following, we also use another ghost sample $X''$ independently generated from the same distribution as $X'$.

**Proposition 17** [Yu, 1994] *Let $X_1, \ldots, X_n$ be a sequence of samples drawn from a stationary $\beta$-mixing process with coefficients $\{\beta_i\}$. Let $Q$, $Q'$ be the distributions of $X(H)$ and $X'(H)$, respectively. For any measurable function*

[7] The extension to the general case is straightforward.

*$h : \mathcal{X}^{m_n k_n} \to \mathbb{R}$ bounded by $B$*

$$|\mathbb{E}_Q \left[h(X(H))\right] - \mathbb{E}_{Q'} \left[h(X'(H))\right]| \leq B m_n \beta_{k_n}.$$

Before moving to the extension of Propsition 12 to $\beta$ mixing processes, we report this technical lemma.

**Lemma 18** *Let $\mathcal{F}$ be a class of functions $f : \mathcal{X} \to \mathbb{R}$ bounded in absolute value by $B$ and $X_1, \ldots, X_n$ be a sequence of samples drawn from a stationary $\beta$-mixing process with coefficients $\{\beta_i\}$. For any $\epsilon > 0$*

$$\mathbb{P}\Big[\exists f \in \mathcal{F} : \|f\| - 2\|f\|_{X_1^n} > \epsilon\Big] \leq 2\delta(\sqrt{2}\epsilon) + 2m_n\beta_{k_n}, \tag{2.24}$$

$$\mathbb{P}\Big[\exists f \in \mathcal{F} : \|f\|_{X_1^n} - 2\sqrt{2}\|f\| > \epsilon\Big] \leq 2\delta(\sqrt{2}\epsilon) + 2m_n\beta_{k_n}, \tag{2.25}$$

*where*

$$\delta(\epsilon) = 3\mathbb{E}\left[\mathcal{N}_2\left(\frac{\sqrt{2}}{24}\epsilon, \mathcal{F}, X'(H) \cup X''(H)\right)\right] \exp\left(-\frac{m_n\epsilon^2}{288B^2}\right).$$

**Proof** Similar to Meir [2000], we first introduce $\overline{\mathcal{F}}$ as the class of block functions $\bar{f} : \mathcal{X}^{k_n} \to \mathbb{R}$ defined as

$$\bar{f}\big(X(H_j)\big)^2 = \frac{1}{k_n} \sum_{t \in H_j} f(X_t)^2.$$

It is interesting to notice that block functions have exactly the same norms as the functions in $\mathcal{F}$. In fact

$$\|\bar{f}\|^2_{X(H)} = \frac{1}{m_n} \sum_{j=1}^{m_n} |\bar{f}(X(H_j))|^2 = \frac{1}{m_n} \sum_{j=1}^{m_n} \frac{1}{k_n} \sum_{t \in H_j} |f(X_t)|^2 = \|f\|_{X(H)}, \tag{2.26}$$

and

$$\|\bar{f}\|^2 = \mathbb{E}\left[|\bar{f}(X(H_1))|^2\right] = \frac{1}{k_n}\sum_{t\in H_1}\mathbb{E}\left[|f(X_t)|^2\right] = \mathbb{E}\left[|f(X_1)|^2\right] = \|f\|, \tag{2.27}$$

where in Eq. 2.27, we used the fact that the process is stationary. We now focus on Eq. 2.24

$$\begin{aligned}
&\mathbb{P}\left[\exists f\in\mathcal{F} : \|f\| - 2\|f\|_{X_1^n} > \epsilon\right]\\
&\quad\overset{\text{(a)}}{\leq} \mathbb{P}\left[\exists f\in\mathcal{F} : \|f\| - \left(\|f\|_{X(H)} + \|f\|_{X(E)}\right) > \epsilon\right]\\
&\quad\overset{\text{(b)}}{=} \mathbb{P}\left[\exists f\in\mathcal{F} : \frac{1}{2}\left(\|f\| - 2\|f\|_{X(H)}\right) + \frac{1}{2}\left(\|f\| - 2\|f\|_{X(E)}\right) > \epsilon\right]\\
&\quad\overset{\text{(c)}}{\leq} \mathbb{P}\left[\exists f\in\mathcal{F} : \|f\| - 2\|f\|_{X(H)} > 2\epsilon\right] + \mathbb{P}\left[\exists f\in\mathcal{F} : \|f\| - 2\|f\|_{X(E)} > 2\epsilon\right]\\
&\quad\overset{\text{(d)}}{=} 2\mathbb{P}\left[\exists \bar{f}\in\overline{\mathcal{F}} : \|\bar{f}\| - 2\|\bar{f}\|_{X(H)} > 2\epsilon\right]\\
&\quad\overset{\text{(e)}}{\leq} 2\left(\mathbb{P}\left[\exists \bar{f}\in\overline{\mathcal{F}} : \|\bar{f}\| - 2\|\bar{f}\|_{X'(H)} > 2\epsilon\right] + m_n\beta_{k_n}\right)\\
&\quad\overset{\text{(f)}}{\leq} 2\delta'(2\epsilon) + 2m_n\beta_{k_n}.
\end{aligned}$$

**(a)** We used the inequality $\sqrt{a+b} \geq \frac{1}{\sqrt{2}}(\sqrt{a}+\sqrt{b})$ to split the norm $\|f\|_{X_1^n} \geq \frac{1}{2}\left(\|f\|_{X(H)} + \|f\|_{X(E)}\right)$.
**(b)** Algebra.
**(c)** Split the probability.
**(d)** (1) Since the process is stationary the distribution over the even blocks is the same as the distribution over the odd blocks. (2) From Eqs. 2.26 and 2.27.
**(e)** Using Proposition 17 with $h$ equals to the indicator function of the event inside the bracket, and the fact that the indicator function is bounded by $B = 1$ and its expected value is equal to the probability of the event.

**(f)** Lemma 12 on space $\overline{\mathcal{F}}$ where

$$\delta'(\epsilon) = 3\mathbb{E}\left[\mathcal{N}_2\left(\frac{\sqrt{2}}{24}\epsilon, \overline{\mathcal{F}}, \{X'(H_j), X''(H_j)\}_{j=1}^{m_n}\right)\right] \exp\left(-\frac{m_n\epsilon^2}{288B^2}\right),$$

where $X''$ is a ghost sample independently generated from the same distribution as $X'$. Now we relate the $\ell_2$-covering number of $\overline{\mathcal{F}}$ to the covering number of $\mathcal{F}$. Using the definition of $\bar{f}$ we have

$$\begin{aligned}
||\bar{f} - \bar{g}||^2_{X(H)} &= \frac{1}{m_n}\sum_{j=1}^{m_n}\Big(\bar{f}\big(X(H_j)\big) - \bar{g}\big(X(H_j)\big)\Big)^2 \\
&= \frac{1}{m_n k_n}\sum_{j=1}^{m_n}\left[\Big(\sum_{t\in H_j} f(X_t)^2\Big)^{\frac{1}{2}} - \Big(\sum_{t'\in H_j} g(X_{t'})^2\Big)^{\frac{1}{2}}\right]^2 .
\end{aligned}$$

Taking the square and using the Cauchy-Schwarz inequality, each element of the outer summation may be written as

$$\begin{aligned}
&\sum_{t\in H_j}\big(f(X_t)^2 + g(X_t)^2\big) - 2\Big(\sum_{t\in H_j} f(X_t)^2\Big)^{\frac{1}{2}}\Big(\sum_{t'\in H_j} g(X_{t'})^2\Big)^{\frac{1}{2}} \\
&\qquad \le \sum_{t\in H_j}\big(f(X_t)^2 + g(X_t)^2 - 2f(X_t)g(X_t)\big) = \sum_{t\in H_j}\big(f(X_t) - g(X_t)\big)^2.
\end{aligned}$$

By taking the sum over all the odd blocks we obtain

$$||\bar{f} - \bar{g}||^2_{X(H)} \le ||f - g||^2_{X(H)} ,$$

which indicates that $\mathcal{N}_2\big(\epsilon, \overline{\mathcal{F}}, \{X'(H_j), X''(H_j)\}_{j=1}^{m_n}\big) \le \mathcal{N}_2\big(\epsilon, \mathcal{F}, X'(H) \cup X''(H)\big)$. Therefore, we have $\delta'(2\epsilon) \le \delta(2\epsilon) \le \delta(\sqrt{2}\epsilon)$, which concludes the proof.

With a similar approach, we can prove Eq. 2.25

$$
\begin{aligned}
&\mathbb{P}\left[\exists f \in \mathcal{F} : \|f\|_{X_1^n} - 2\sqrt{2}\|f\| > \epsilon\right] \\
&\overset{\text{(a)}}{\leq} \mathbb{P}\left[\exists f \in \mathcal{F} : \frac{\sqrt{2}}{2}\big(\|f\|_{X(H)} + \|f\|_{X(E)}\big) - 2\sqrt{2}\|f\| > \epsilon\right] \\
&\overset{\text{(b)}}{=} \mathbb{P}\left[\exists f \in \mathcal{F} : \left(\frac{\sqrt{2}}{2}\|f\|_{X(H)} - \sqrt{2}\|f\|\right) + \left(\frac{\sqrt{2}}{2}\|f\|_{X(E)} - \sqrt{2}\|f\|\right) > \epsilon\right] \\
&\overset{\text{(c)}}{\leq} \mathbb{P}\left[\exists f \in \mathcal{F} : \|f\|_{X(H)} - 2\|f\| > \sqrt{2}\epsilon\right] + \mathbb{P}\left[\exists f \in \mathcal{F} : \|f\|_{X(E)} - 2\|f\| > \sqrt{2}\epsilon\right] \\
&\overset{\text{(d)}}{=} 2\mathbb{P}\left[\exists \bar{f} \in \overline{\mathcal{F}} : \|\bar{f}\|_{X(H)} - 2\|\bar{f}\| > \sqrt{2}\epsilon\right] \\
&\overset{\text{(e)}}{\leq} 2\left(\mathbb{P}\left[\exists \bar{f} \in \overline{\mathcal{F}} : \|\bar{f}\|_{X'(H)} - 2\|\bar{f}\| > \sqrt{2}\epsilon\right] + m_n\beta_{k_n}\right) \\
&\overset{\text{(f)}}{\leq} 2\delta'(\sqrt{2}\epsilon) + 2m_n\beta_{k_n} \leq 2\delta(\sqrt{2}\epsilon) + 2m_n\beta_{k_n}.
\end{aligned}
$$

**(a)** We used the inequality $\sqrt{a+b} \leq (\sqrt{a} + \sqrt{b})$ to split the norm $\|f\|_{X_1^n} \leq \frac{\sqrt{2}}{2}\big(\|f\|_{X(H)} + \|f\|_{X(E)}\big)$.
**(b)-(f)** use the same arguments as before. ■

**Corollary 19** *Let $\mathcal{F}$ be a class of linear functions $f : \mathcal{X} \to \mathbb{R}$ of dimension $d$, $\widetilde{\mathcal{F}}$ be the class of functions obtained by truncating functions $f \in \mathcal{F}$ at a threshold $B$, and $X_1^n = \{X_1, \ldots, X_n\}$ be a sequence of samples drawn from a stationary exponentially fast $\beta$-mixing process with coefficients $\{\beta_i\}$. By inverting the bound of Lemma 18, for any $\tilde{f} \in \widetilde{\mathcal{F}}$ we have*

$$
\|\tilde{f}\| - 2\|\tilde{f}\|_{X_1^n} \leq \epsilon(\delta), \qquad \|\tilde{f}\|_{X_1^n} - 2\sqrt{2}\|\tilde{f}\| \leq \epsilon(\delta),
$$

*with probability $1 - \delta$, where*

$$
\epsilon(\delta) = 12B\sqrt{\frac{2\Lambda(n,d,\delta)}{n}\max\left\{\frac{\Lambda(n,d,\delta)}{b}, 1\right\}^{1/\kappa}}, \tag{2.28}
$$

*and* $\Lambda(n,d,\delta) = 2(d+1)\log n + \log\frac{e}{\delta} + \log^{+}\left(\max\{18(6e)^{2(d+1)}, \bar{\beta}\}\right)$.

**Proof** In order to prove the statement, we need to verify that $\epsilon$ in Eq. 2.28 satisfies

$$\delta' = 6\mathbb{E}\left[\mathcal{N}_2\left(\frac{1}{12}\epsilon, \widetilde{\mathcal{F}}, X'(H) \cup X''(H)\right)\right] \exp\left(-\frac{m_n\epsilon^2}{144B^2}\right) + 2m_n\beta_{k_n} \leq \delta \;.$$

Using Proposition 13 the covering number can be bounded by

$$\mathbb{E}\left[\mathcal{N}_2\left(\frac{1}{12}\epsilon, \widetilde{\mathcal{F}}, X'(H) \cup X''(H)\right)\right] \leq 3\left(\frac{1728eB^2}{\epsilon^2}\right)^{2(d+1)} .$$

By recalling the definition of the $\beta$-coefficients $\{\beta_i\}$ and $k_n \geq 1$ we have

$$2m_n\beta_{k_n} \leq \frac{n}{k_n}\bar{\beta}\exp(-bk_n^{\kappa}) \leq n\bar{\beta}\exp(-bk_n^{\kappa}) \;.$$

From the last two inequalities, $m_n = n/2k_n$, setting $C_1 = 1728eB^2$ and $D = 2(d+1)$ we obtain

$$\delta' \leq 18\left(\frac{C_1}{\epsilon^2}\right)^D \exp\left(-\frac{n\epsilon^2}{144B^2}\frac{1}{2k_n}\right) + n\bar{\beta}\exp(-bk_n^{\kappa}).$$

By equalizing the arguments of the two exponential we obtain the definition of $k_n$ as

$$k_n = \left\lceil \left(\frac{nC_2\epsilon^2}{b}\right)^{\frac{1}{\kappa+1}} \right\rceil ,$$

where $C_2 = (576B^2)^{-1}$, which implies

$$\max\left\{\left(\frac{nC_2\epsilon^2}{b}\right)^{\frac{1}{\kappa+1}}, 1\right\} \leq k_n \leq \max\left\{\left(\frac{2nC_2\epsilon^2}{b}\right)^{\frac{1}{\kappa+1}}, 1\right\}.$$

Thus we have the bound

$$\frac{1}{2k_n} \geq \frac{1}{4}\min\left\{\left(\frac{b}{nC_2\epsilon^2}\right)^{\frac{1}{\kappa+1}}, 2\right\} \geq \frac{1}{4}\min\left\{\left(\frac{b}{nC_2\epsilon^2}\right)^{\frac{1}{\kappa+1}}, 1\right\}.$$

Using the above inequalities, we may write $\delta'$ as

$$\delta' \leq 18\left(\frac{C_1}{\epsilon^2}\right)^D \exp\left(-\min\left\{\frac{b}{nC_2\epsilon^2}, 1\right\}^{\frac{1}{\kappa+1}} nC_2\epsilon^2\right) + n\bar{\beta}\exp\left(-b\max\left\{\frac{nC_2\epsilon^2}{b}, 1\right\}^{\frac{\kappa}{\kappa+1}}\right).$$

The objective now is to make the arguments of the two exponential equal. For the second argument we have

$$b\max\left\{\frac{nC_2\epsilon^2}{b}, 1\right\}^{\frac{\kappa}{\kappa+1}} = b\max\left\{\frac{nC_2\epsilon^2}{b}, 1\right\}\min\left\{\frac{b}{nC_2\epsilon^2}, 1\right\}^{\frac{1}{\kappa+1}} \geq nC_2\epsilon^2\min\left\{\frac{b}{nC_2\epsilon^2}, 1\right\}^{\frac{1}{\kappa+1}}.$$

Thus

$$\delta' \leq \left(18\left(\frac{C_1}{\epsilon^2}\right)^D + n\bar{\beta}\right)\exp\left(-\min\left\{\frac{b}{nC_2\epsilon^2}, 1\right\}^{\frac{1}{\kappa+1}} nC_2\epsilon^2\right).$$

Now we plug in $\epsilon$ from Eq. 2.28. Using the fact that $\Lambda \geq 1$, we know that $\epsilon^2 \geq (nC_2)^{-1}$, and thus

$$\delta' \leq \left(18\left(nC_1C_2\right)^D + n\bar{\beta}\right)\exp\left(-\Lambda\right).$$

Using the definition of $\Lambda$, we obtain

$$\delta' \leq \left(18\left(nC_1C_2\right)^D + n\bar{\beta}\right) n^{-D}\max\{18(C_1C_2)^D, \bar{\beta}\}^{-1}\frac{\delta}{e} \leq (1+n^{1-D})\frac{\delta}{e} \leq (1+1)\frac{\delta}{e} \leq \delta\ ,$$

which concludes the proof. ■

In order to understand better the shape of the estimation error, we consider a simple $\beta$-mixing process with parameters $\bar{\beta} = b = \kappa = 1$. Eq. 2.28 reduces to

$$\epsilon(\delta) = \sqrt{\frac{288B^2\Lambda(n,d,\delta)^2}{n}},$$

with $\Lambda(n,d,\delta) = 2(d+1)\log n + \log\frac{e}{\delta} + \log\left(18(6e)^{2(d+1)}\right)$. It is interesting to notice that the shape of the bound in this case resembles the structure of the bound in Corollary 14 for i.i.d. samples. Finally, we report the non-functional version of the previous corollary.

**Corollary 20** *Let $\mathcal{F}$ be a class of linear functions $f : \mathcal{X} \to \mathbb{R}$ of dimension $d$ such that its features $\varphi_i : \mathcal{X} \to \mathbb{R}$ are bounded in absolute value by $L$ for any $i = 1, \ldots, d$ and $X_1^n = \{X_1, \ldots, X_n\}$ be a sequence of samples drawn from a stationary exponentially fast $\beta$-mixing process with coefficients $\{\beta_i\}$. For any $f \in \mathcal{F}$ we have*

$$\|f\| - 2\|f\|_{X_1^n} \leq \epsilon(\delta), \qquad \|f\|_{X_1^n} - 2\sqrt{2}\|f\| \leq \epsilon(\delta),$$

*with probability $1-\delta$, where*

$$\epsilon(\delta) = 12||\alpha||L\sqrt{\frac{2\Lambda(n,d,\delta)}{n}\max\left\{\frac{\Lambda(n,d,\delta)}{b}, 1\right\}^{1/\kappa}},$$

*and $\Lambda(n,d,\delta) = 2(d+1)\log n + \log\frac{e}{\delta} + \log^+\left(\max\{18(6e)^{2(d+1)}, \bar{\beta}\}\right)$.*

**Proof** Let $\mathcal{G} = \left\{g_\alpha = \frac{f_\alpha}{L||\alpha||}\right\}$ so that

$$||g_\alpha||_\infty = \frac{1}{L||\alpha||}||f_\alpha||_\infty \leq \frac{1}{L||\alpha||}||\alpha|| \sup_i ||\varphi_i(x)||_\infty \leq 1.$$

We can thus apply Lemma 18 to the bounded space $\mathcal{G}$ with $B = 1$. By using a similar inversion as in Corollary 19, we thus obtain that with probability

$1-\delta$, for any function $g_\alpha \in \mathcal{G}$

$$\|g_\alpha\| - 2\|g_\alpha\|_{X_1^n} \leq \epsilon(\delta), \qquad \|g_\alpha\|_{X_1^n} - 2\sqrt{2}\|g_\alpha\| \leq \epsilon(\delta),$$

with

$$\epsilon(\delta) = 12\sqrt{\frac{2\Lambda(n,d,\delta)}{n}\max\left\{\frac{\Lambda(n,d,\delta)}{b},1\right\}^{1/\kappa}}\,.$$

Finally, we notice that $||g_\alpha|| = \frac{1}{L||\alpha||}||f_\alpha||$ and $||g_\alpha||_{X_1^n} = \frac{1}{L||\alpha||}||f_\alpha||_{X_1^n}$ and the statement follows. ■

**Corollary 21** *Let $f : \mathcal{X} \to \mathbb{R}$ be a linear function, $\tilde{f}$ be its truncation at a threshold $B$, and $X_1^n = \{X_1, \dots, X_n\}$ be a sequence of samples drawn from a stationary exponentially fast $\beta$-mixing process with coefficients $\{\beta_i\}$. Then*

$$\|\tilde{f}\| - 2\|\tilde{f}\|_{X_1^n} \leq \epsilon(\delta), \qquad \|\tilde{f}\|_{X_1^n} - 2\sqrt{2}\|\tilde{f}\| \leq \epsilon(\delta),$$

*with probability $1-\delta$, where*

$$\epsilon(\delta) = 12B\sqrt{\frac{2\Lambda(n,\delta)}{n}\max\left\{\frac{\Lambda(n,\delta)}{b},1\right\}^{1/\kappa}}\,,$$

$\Lambda(n,\delta) = \log\frac{e}{\delta} + \log\left(\max\{6, n\bar{\beta}\}\right)$.

**Proof** The proof follows the same steps as in Corollary 19. We have the following sequence of inequalities

$$\begin{aligned}\delta' &\leq 6\exp\left(-\frac{nC_2\epsilon^2}{k_n}\right) + \frac{n}{k_n}\bar{\beta}\exp(-bk_n^\kappa) \leq (6+n\bar{\beta})\exp(-\Lambda) \\ &= (6+n\bar{\beta})\max\{6,n\bar{\beta}\}^{-1}\,\frac{\delta}{e} \leq (1+1)\frac{\delta}{e} \leq \delta\,,\end{aligned}$$

where $C_2 = (576B^2)^{-1}$. ■

### 2.8.3 Markov Chains

We first review the conditions for the convergence of Markov chains (Theorem 13.3.3. in Meyn and R 1993).

**Proposition 22** *Let $\mathcal{M}$ be an ergodic and aperiodic Markov chain defined on $\mathcal{X}$ with stationary distribution $\rho$. If $P(A|x)$ is the transition kernel of $\mathcal{M}$ with $A \subseteq \mathcal{X}$ and $x \in \mathcal{X}$, then for any initial distribution $\lambda$*

$$\lim_{i\to\infty} \Big|\Big| \int_{\mathcal{X}} \lambda(dx) P^i(\cdot|x) - \rho(\cdot) \Big|\Big|_{TV} = 0,$$

*where $||\cdot||_{TV}$ is the total variation norm.*

**Definition 23** *Let $\mathcal{M}$ be an ergodic and aperiodic Markov chain with stationary distribution $\rho$. $\mathcal{M}$ is mixing with an exponential rate with parameters $\bar{\beta}, b, \kappa$, if its $\beta$-mixing coefficients $\{\beta_i\}$ satisfy $\beta_i \leq \bar{\beta}\exp(-bi^{\kappa})$. Then for any initial distribution $\lambda$*

$$\Big|\Big| \int_{\mathcal{X}} \lambda(dx) P^i(\cdot|x) - \rho(\cdot) \Big|\Big|_{TV} \leq \bar{\beta}\exp(-bi^{\kappa}).$$

**Lemma 24** *Let $\mathcal{M}$ be an ergodic and aperiodic Markov chain with a stationary distribution $\rho$. Let $X_1, \ldots, X_n$ be a sequence of samples drawn from the stationary distribution of the Markov chain $\rho$ and $X'_1, \ldots, X'_n$ be a sequence of samples such that $X'_1 \sim \rho'$ and $X'_{1<t\leq n}$ are generated by simulating $\mathcal{M}$ from $X'_1$. Let $\eta$ be an event defined on $\mathcal{X}^n$, then*

$$|\mathbb{P}\left[\eta(X_1, \ldots, X_n)\right] - \mathbb{P}\left[\eta(X'_1, \ldots, X'_n)\right]| \leq \|\rho' - \rho\|_{TV}$$

**Proof** We prove one side of the inequality. Let $Q$ be the conditional joint distribution of $(X_{1<t\leq n}|X_1 = x)$ and $Q'$ be the conditional joint distribution of $(X'_{1<t\leq n}|X'_1 = x)$. We first notice that $Q$ is exactly the same as $Q'$. In fact, the first sequence $(X_{1<t\leq n})$ is generated by drawing $X_1$ from the stationary

distribution $\rho$ and then following the Markov chain. Similarly, the second sequence $(X'_{1<t\leq n})$ is obtained following the Markov chain from $X'_1 \sim \rho'$. As a result, the conditional distributions of the two sequences is exactly the same and just depend on the Markov chain. As a result, we obtain the following sequence of inequalities

$$\begin{aligned}
\mathbb{P}\Big[\eta(X_1,\ldots,X_n)\Big] &= \mathbb{E}_{X_1,\ldots,X_n}\left[\mathbb{I}\left\{\eta(X_1,\ldots,X_n)\right\}\right] \\
&= \mathbb{E}_{X_1\sim\rho}\left[\mathbb{E}_{X_2,\ldots,X_n}\left[\mathbb{I}\left\{\eta(X_1,X_2\ldots,X_n)\right\}|X_1\right]\right] \\
&= \mathbb{E}_{X_1\sim\rho}\left[\mathbb{E}_{X'_2,\ldots,X'_n}\left[\mathbb{I}\left\{\eta(X_1,X'_2\ldots,X'_n)\right\}|X_1\right]\right] \\
&\overset{(a)}{\leq} \mathbb{E}_{X_1\sim\rho'}\left[\mathbb{E}_{X'_2,\ldots,X'_n}\left[\mathbb{I}\left\{\eta(X_1,X'_2\ldots,X'_n)\right\}|X_1\right]\right] + \|\rho'-\rho\|_{TV} \\
&\overset{(b)}{=} \mathbb{E}_{X'_1\sim\rho'}\left[\mathbb{E}_{X'_2,\ldots,X'_n}\left[\mathbb{I}\left\{\eta(X'_1,X'_2\ldots,X'_n)\right\}|X'_1\right]\right] + \|\rho'-\rho\|_{TV} \\
&= \mathbb{P}\Big[\eta(X'_1,\ldots,X'_n)\Big] + \|\rho'-\rho\|_{TV}.
\end{aligned}$$

Note that $\mathbb{I}\left\{\cdot\right\}$ is the indicator function.
**(a)** simply follows from

$$\begin{aligned}
\mathbb{E}_{X\sim\rho}\left[f(X)\right] - \mathbb{E}_{X\sim\rho'}\left[f(X)\right] &= \int_{\mathcal{X}} f(x)\rho(dx) - \int_{\mathcal{X}} f(x)\rho'(dx) \\
&\leq ||f||_\infty \int_{\mathcal{X}} \big(\rho(dx)-\rho'(dx)\big) \leq ||f||_\infty ||\rho-\rho'||_{TV}.
\end{aligned}$$

**(b)** From the fact that $X_1 = X'_1 = x$. ■

**Lemma 25** *Let $\mathcal{F}$ be a class of functions $f : \mathcal{X} \to \mathbb{R}$ bounded in absolute value by $B$, $\mathcal{M}$ be a an ergodic and aperiodic Markov chain with a stationary distribution $\rho$. Let $\mathcal{M}$ be mixing with an exponential rate with parameters $\bar{\beta}, b, \kappa$. Let $\lambda$ be an initial distribution over $\mathcal{X}$ and $X_1,\ldots,X_n$ be a sequence of samples such that $X_1 \sim \lambda$ and $X_{1<t\leq n}$ obtained by following $\mathcal{M}$ from $X_1$.*

*For any $\epsilon > 0$,*

$$\mathbb{P}\Big[\exists f \in \mathcal{F} : \|f\| - 2\|f\|_{X_1^n} > \epsilon\Big] \leq \|\lambda - \rho\|_{TV} + 2\delta(\sqrt{2}\epsilon) + 2m_n\beta_{k_n},$$

*and*

$$\mathbb{P}\Big[\exists f \in \mathcal{F} : \|f\|_{X_1^n} - 2\sqrt{2}\|f\| > \epsilon\Big] \leq \|\lambda - \rho\|_{TV} + 2\delta(\sqrt{2}\epsilon) + 2m_n\beta_{k_n},$$

*where*

$$\delta(\epsilon) = 3\mathbb{E}\left[\mathcal{N}_2\left(\frac{\sqrt{2}}{24}\epsilon, \mathcal{F}, X(H) \cup X'(H)\right)\right] \exp\left(-\frac{m_n\epsilon^2}{288B^2}\right).$$

**Proof** The proof is an immediate consequence of Lemma 18 and Lemma 24 by defining $\eta(X_1, \ldots, X_n)$ as

$$\eta(X_1, \ldots, X_n) = \{\exists f \in \mathcal{F} : \|f\| - 2\|f\|_{X_1^n} > \epsilon\},$$

and

$$\eta(X_1, \ldots, X_n) = \{\exists f \in \mathcal{F} : \|f\|_{X_1^n} - 2\sqrt{2}\|f\| > \epsilon\},$$

respectively. ■

Finally, we consider a special case in which out of the $n$ total number of samples, $\tilde{n}$ $(1 \leq \tilde{n} < n)$ are used to "burn" the chain and $n - \tilde{n}$ are actually used as training samples.

**Lemma 26** *Let $\mathcal{F}$ be a class of linear functions $f : \mathcal{X} \to \mathbb{R}$ of dimension $d$ and $\widetilde{\mathcal{F}}$ be the class of functions obtained by truncating functions $f \in \mathcal{F}$ at a threshold $B$. Let $\mathcal{M}$ be an ergodic and aperiodic Markov chain with a stationary distribution $\rho$. Let $\mathcal{M}$ be mixing with an exponential rate with parameters $\bar{\beta}, b, \kappa$. Let $\mu$ be the initial distribution and $X_1, \ldots, X_n$ be a sequence of samples such that $X_1 \sim \mu$ and $X_{1<t\leq n}$ obtained by following $\mathcal{M}$*

*from $X_1$. If the first $\tilde{n}$ $(1 \leq \tilde{n} < n)$ samples are used to burn the chain and $n - \tilde{n}$ are actually used as training samples, by inverting Lemma 25, for any $\tilde{f} \in \widetilde{\mathcal{F}}$, we obtain*

$$\|\tilde{f}\| - 2\|\tilde{f}\|_{X_1^n} \leq \epsilon(\delta), \qquad \|\tilde{f}\|_{X_1^n} - 2\sqrt{2}\|\tilde{f}\| \leq \epsilon(\delta),$$

*with probability $1 - \delta$, where*

$$\epsilon(\delta) = 12B\sqrt{\frac{2\Lambda(n-\tilde{n},d,\delta)}{(n-\tilde{n})}\max\left\{\frac{\Lambda(n-\tilde{n},d,\delta)}{b},1\right\}^{1/\kappa}} \quad ,$$

*and $\Lambda(n,d,\delta) = 2(d+1)\log n + \log\frac{e}{\delta} + \log^{+}\left(\max\{18(6e)^{2(d+1)}, \bar{\beta}\}\right)$, and $\tilde{n} = \left(\frac{1}{b}\log\frac{2e\bar{\beta}n}{\delta}\right)^{1/\kappa}$.*

**Proof** After $\tilde{n}$ steps, the first sample used in the training set $(X_{\tilde{n}+1})$ is drawn from the distribution $\lambda = \mu P^{\tilde{n}}$. Using Proposition 22 and Definition 23 we have

$$||\lambda - \rho||_{TV} \leq \bar{\beta}\exp(-b\tilde{n}^{\kappa}). \tag{2.29}$$

We first substitute the total variation in Lemma 25 with the bound in Eq. 2.29, and then verify that $\epsilon$ in Eq. 26 satisfies the following inequality.

$$\begin{aligned}\delta' &= \|\lambda - \rho\|_{TV} + 2\delta(\sqrt{2}\epsilon) + 2m_{n-\tilde{n}}\beta_{k_{n-\tilde{n}}} \\ &\leq \bar{\beta}\exp(-b\tilde{n}^{\kappa}) + 18\left(\frac{C_1}{\epsilon^2}\right)^{D}\exp\left(-\frac{(n-\tilde{n})C_2\epsilon^2}{k_{n-\tilde{n}}}\right) + (n-\tilde{n})\bar{\beta}\exp(-bk_{n-\tilde{n}}^{\kappa}) \\ &\leq \big(\frac{1}{2n} + 1 + (n-\tilde{n})^{1-D}\big)\frac{\delta}{e} \leq (\frac{1}{2} + 1 + 1)\frac{\delta}{e} \leq \delta,\end{aligned}$$

where $C_1 = 1728eB^2$ and $C_2 = (288B^2)^{-1}$. The above inequality can be verified by following the same steps as in Corollary 19 and by optimizing the bound for $\tilde{n}$. ■

# Chapter 3

# Analysis of Classification-based Policy Iteration Algorithms [MGH2, MGH5, MGH10, MGH11, MGH13, MGH14, MGH17]

We introduce a variant of the classification-based approach to policy iteration which uses a cost-sensitive loss function weighting each classification mistake by its actual *regret*, i.e., the difference between the action-value of the greedy action and of the action chosen by the classifier. For this algorithm, we provide a full finite-sample analysis. Our results state a performance bound in terms of the number of policy improvement steps, the number of rollouts used in each iteration, the capacity of the considered policy space (classifier), and a capacity measure which indicates how well the policy space can approximate policies that are greedy w.r.t. any of its members. The analysis reveals a tradeoff between the estimation and approximation errors in this classification-based policy iteration setting. Furthermore it confirms the intuition that classification-based policy iteration algorithms could be favorably compared to value-based approaches when the policies can be approximated more easily than their corresponding value functions. We also study the consistency of the algorithm when there exists a sequence of policy spaces with increasing capacity.

## 3.1 Introduction

*Policy iteration* [Howard, 1960] is a method of computing an optimal policy for any given Markov decision process (MDP). It is an iterative procedure that discovers a deterministic optimal policy by generating a sequence of monotonically improving policies. Each iteration $k$ of this algorithm consists of two phases: *policy evaluation* in which the action-value function $Q^{\pi_k}$ of the current policy $\pi_k$ is computed, and *policy improvement* in which the new (improved) policy $\pi_{k+1}$ is generated as the greedy policy w.r.t. $Q^{\pi_k}$, i.e., $\pi_{k+1}(x) = \arg\max_{a \in \mathcal{A}} Q^{\pi_k}(x, a)$. Unfortunately, in MDPs with large (or continuous) state and action spaces, the policy evaluation problem cannot be solved exactly and approximation techniques are required. In approximate policy iteration (API), a function approximation scheme is usually employed in the policy evaluation phase. The most common approach is to find a good approximation of the value function of $\pi_k$ in a real-valued function space (see e.g., Bradtke and Barto 1996, Lagoudakis and Parr 2003a). The main drawbacks of this approach are: **1)** the action-value function, $Q^{\pi_k}$, is not known in advance and its high quality samples are often very expensive to obtain, if this option is possible at all, **2)** it is often difficult to find a function space rich enough to represent the action-value function accurately, and thus, careful hand-tuning is needed to achieve satisfactory results, **3)** for the success of policy iteration, it is not necessary to estimate $Q^{\pi_k}$ accurately at every state-action pair, what is important is to have an approximation of the action-value function whose greedy policy has a performance similar to the greedy policy w.r.t. the actual action-value function, and **4)** this method may not be the right choice in domains where good policies are easier to represent and learn than the corresponding value functions.

To address the above issues, mainly **3** and **4**,[1] variants of API have been proposed that replace the usual value function learning step (approximating

[1] The first drawback is shared by all reinforcement learning algorithms and the second one is common to all practical applications of machine learning methods.

the action-value function over the entire state-action space) with a learning step in a policy space [Lagoudakis and Parr, 2003b, Fern et al., 2004]. The main idea is to cast the policy improvement step as a *classification* problem. The training set is generated using rollout estimates of $Q^\pi$ over a finite number of states $\mathcal{D} = \{x_i\}_{i=1}^N$, called the *rollout set*, and for any action $a \in \mathcal{A}$.[2] For each $x \in \mathcal{D}$, if the estimated value $\widehat{Q}^\pi(x, a^*)$ is greater than the estimated value of all other actions with *high confidence*, the state-action pair $(x, a^*)$ is added to the training set with a positive label. In this case, $(x, a)$ for the rest of the actions are labeled negative and added to the training set. The policy improvement step thus reduces to solving a classification problem to find a policy in a given hypothesis space that best predicts the greedy action at every state. Although whether selecting a suitable policy space is any easier than a value function space is highly debatable, we can argue that the classification-based API methods can be advantageous in problems where good policies are easier to represent and learn than their value functions.

The classification-based API algorithms can be viewed as a type of reduction from reinforcement learning (RL) to classification, i.e., solving a MDP by generating and solving a series of classification problems. There have been other proposals for reducing RL to classification. Langford and Zadrozny [2005] provided a formal reduction from RL to classification, showing that $\epsilon$-accurate classification implies near optimal RL. This approach uses an optimistic variant of sparse sampling to generate $h$ classification problems, one for each horizon time step. The main limitation of this work is that it does not provide a practical method for generating training examples for these classification problems. Bagnell et al. [2003] introduced an algorithm for learning non-stationary policies in RL. For a specified horizon $h$, their approach learns a sequence of $h$ policies. At each iteration, all policies are fixed except for one, which is optimized by forming a classification problem via policy roll-

[2] It is worth stressing that $Q^\pi$ is estimated just on states in $\mathcal{D}$ and not over the entire state-action space.

out. Perhaps the closest approach to the classification-based API methods proposed and analyzed in this paper is the group of algorithms that are introduced and analyzed in Kakade and Langford [2002] and Kakade [2003] under the name *conservative policy iteration* (CPI).[3] The main algorithmic difference between CPI and the classification-based API methods studied in this paper is while the output of the classifier is directly assigned to the next policy in our algorithms, CPI algorithms perform a more conservative policy update in which the new policy $\pi_{k+1}$ is a mixture distribution of the current policy $\pi_k$ and the output of the classifier (policies might be stochastic). This conservative update gives CPI two desirable properties: **1)** it guarantees to improve the policy at each iteration, i.e., the value function of $\pi_{k+1}$ is larger on average than the value function of $\pi_k$, and **2)** it has a stopping condition based on the quality of the generated policy (it stops whenever it cannot guarantee that the new policy has a better performance than the previous one). These properties can potentially make CPI a very appealing API algorithm, mainly because other API methods have no guarantee to generate monotonically improving policies. This includes both value function based API algorithms such as LSPI [Lagoudakis and Parr, 2003a] and classification-based API methods. However, Ghavamzadeh and Lazaric [2012] showed that CPI's desirable properties do not come for free. The analysis of Ghavamzadeh and Lazaric [2012] reveals that in order to achieve the same level of accuracy, CPI requires more iterations, and thus, more samples than the classification-based API algorithms proposed in this paper. This indicates that although CPI's conservative update allows it to have a monotonically improving behavior, it slows down the algorithm and increases its sample complexity. Furthermore, CPI may converge to suboptimal policies whose performance is not better than those returned by the algorithms studied in this paper.

Although the classification-based API algorithms have been successfully

[3] While in Kakade and Langford [2002] the algorithm is presented as a rollout value function based approach, in the more detailed description and analysis of CPI found in Kakade [2003], the algorithm is presented as a classification-based API method.

applied to benchmark problems [Lagoudakis and Parr, 2003b, Fern et al., 2004] and have been modified to become more computationally efficient [Dimitrakakis and Lagoudakis, 2008b], a full theoretical understanding of them is still lacking. Fern et al. [2006] and Dimitrakakis and Lagoudakis [2008a] provide a preliminary theoretical analysis of their algorithm. In particular, they both bound the difference in performance at each iteration between the learned policy and the true greedy policy. Their analysis is limited to one step policy update (they do not show how the error in the policy update is propagated through the iterations of the API algorithm) and either to finite class of policies (in Fern et al., 2006) or to a specific architecture (a uniform grid in Dimitrakakis and Lagoudakis, 2008a). Moreover, the bound reported in Fern et al. [2006] depends inversely on the minimum $Q$-value gap between a greedy and a sub-greedy action over the state space. In some classes of MDPs this gap can be arbitrarily small so that the learned policy can be arbitrarily worse than the greedy policy. In order to deal with this problem Dimitrakakis and Lagoudakis [2008a] assume the action-value functions to be smooth and the probability of states with a small $Q$-value gap to be small.

In this paper, we derive a full finite-sample analysis of a classification-based API algorithm, called *direct policy iteration* (DPI). It is based on a cost-sensitive loss function weighting each classification error by its actual *regret*, i.e., the difference between the action-value of the greedy action and of the action chosen by DPI. Using this loss, we are able to derive a performance bound with no dependency on the minimum $Q$-value gap and no assumption on the probability of states with small Q-value gap. Our analysis further extends those in Fern et al. [2006] and Dimitrakakis and Lagoudakis [2008a] by considering arbitrary policy spaces, and by showing how the error at each step is propagated through the iterations of the API algorithm. We also analyze the consistency of DPI when there exists a sequence of policy spaces with increasing capacity. We first use a counterexample and show that

DPI is not consistent in general, and then prove its consistency for the class of Lipschitz MDPs. We conclude the paper with a discussion on different theoretical and practical aspects of DPI.

The rest of the paper is organized as follows. In Section 3.2, we define the basic concepts and set up the notation used in the paper. Section 3.3 introduces the general classification-based approach to policy iteration and details the DPI algorithm. In Section 3.4, we provide a finite-sample analysis for the DPI algorithm. The approximation error and the consistency of the algorithm are discussed in Section 3.5. While all the main results are derived in case of two actions, i.e., $|\mathcal{A}| = 2$, in Section 3.6 we show how they can be extended to the general case of multiple actions. In Section 3.7, we conclude the paper and discuss the obtained results.

## 3.2 Preliminaries

In this section, we set the notation used throughout the paper. A discounted Markov decision process (MDP) $\mathcal{M}$ is a tuple $\langle \mathcal{X}, \mathcal{A}, r, p, \gamma \rangle$, where the state space $\mathcal{X}$ is a bounded closed subset of a Euclidean space $\mathbb{R}^d$, the set of actions $\mathcal{A}$ is finite ($|\mathcal{A}| < \infty$), the reward function $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is uniformly bounded by $R_{\max}$, the transition model $p(\cdot|x, a)$ is a distribution over $\mathcal{X}$, and $\gamma \in (0, 1)$ is a discount factor. Let $\mathcal{B}^V(\mathcal{X}; V_{\max})$ and $\mathcal{B}(\mathcal{X} \times \mathcal{A}; Q_{\max})$ be the space of Borel measurable value and action-value functions bounded by $V_{\max}$ and $Q_{\max}$ ($V_{\max} = Q_{\max} = \frac{R_{\max}}{1-\gamma}$), respectively. We also use $\mathcal{B}^\pi(\mathcal{X})$ to denote the space of deterministic policies $\pi : \mathcal{X} \to \mathcal{A}$. The value function of a policy $\pi$, $V^\pi$, is the unique fixed-point of the Bellman operator $\mathcal{T}^\pi : \mathcal{B}^V(\mathcal{X}; V_{\max}) \to \mathcal{B}^V(\mathcal{X}; V_{\max})$ defined by

$$(\mathcal{T}^\pi V)(x) = r\big(x, \pi(x)\big) + \gamma \int_{\mathcal{X}} p\big(dy|x, \pi(x)\big) V(y).$$

The action-value function $Q^\pi$ is defined as

$$Q^\pi(x,a) = r(x,a) + \gamma \int_{\mathcal{X}} p(dy|x,a) V^\pi(y).$$

Similarly, the optimal value function, $V^*$, is the unique fixed-point of the optimal Bellman operator $\mathcal{T} : \mathcal{B}^V(\mathcal{X}; V_{\max}) \to \mathcal{B}^V(\mathcal{X}; V_{\max})$ defined as

$$(\mathcal{T}V)(x) = \max_{a \in \mathcal{A}} \left[ r(x,a) + \gamma \int_{\mathcal{X}} p(dy|x,a) V(y) \right],$$

and the optimal action-value function $Q^*$ is defined by

$$Q^*(x,a) = r(x,a) + \gamma \int_{\mathcal{X}} p(dy|x,a) V^*(y).$$

We say that a deterministic policy $\pi \in \mathcal{B}^\pi(\mathcal{X})$ is *greedy* w.r.t. an action-value function $Q$, if $\pi(x) \in \arg\max_{a \in \mathcal{A}} Q(x,a), \forall x \in \mathcal{X}$. Greedy policies are important because any greedy policy w.r.t. $Q^*$ is optimal. We define the greedy policy operator $\mathcal{G} : \mathcal{B}^\pi(\mathcal{X}) \to \mathcal{B}^\pi(\mathcal{X})$ as[4]

$$(\mathcal{G}\pi)(x) = \arg\max_{a \in \mathcal{A}} Q^\pi(x,a). \tag{3.1}$$

In the analysis of this paper, $\mathcal{G}$ plays a role similar to the one played by the optimal Bellman operator, $\mathcal{T}$, in the analysis of the fitted value iteration algorithm (Munos and Szepesvári 2008, Section 5).

## 3.3 The DPI Algorithm

In this section, we outline the direct policy iteration (DPI) algorithm. DPI shares the same structure as the algorithms in Lagoudakis and Parr [2003b] and Fern et al. [2004]. Although it can benefit from improvements in **1)**

[4] In Eq. 3.1, the tie among the actions maximizing $Q^\pi(x,a)$ is broken in an arbitrary but consistent manner.

```
Input: policy space $\Pi \subseteq \mathcal{B}^{\pi}(\mathcal{X})$, state distribution $\rho$, number of rollout states
$N$, number of rollouts per state-action pair $M$
Initialize: Let $\pi_0 \in \Pi$ be an arbitrary policy
for $k = 0, 1, 2, \dots$ do
  Construct the rollout set $\mathcal{D}_k = \{x_i\}_{i=1}^{N},\ x_i \overset{\text{iid}}{\sim} \rho$
  for all states $x_i \in \mathcal{D}_k$ and actions $a \in \mathcal{A}$ do
    for $j = 1$ to $M$ do
      Perform a rollout according to policy $\pi_k$ and return $R_j^{\pi_k}(x_i, a) = r(x_i, a) + \sum_{t=1}^{H-1} \gamma^t r(x^t, \pi_k(x^t))$, $x^t \sim p\big(\cdot\,|x^{t-1}, \pi_k(x^{t-1})\big)$ and $x^1 \sim p(\cdot|x_i, a)$
    end for
    $\widehat{Q}^{\pi_k}(x_i, a) = \frac{1}{M} \sum_{j=1}^{M} R_j^{\pi_k}(x_i, a)$
  end for
  $\pi_{k+1} = \arg\min_{\pi \in \Pi} \widehat{\mathcal{L}}_{\pi_k}(\widehat{\rho}; \pi)$          (classifier)
end for
```

Figure 3.1: The Direct Policy Iteration (DPI) algorithm.

selecting states for the rollout set $\mathcal{D}$, **2)** the criteria used to add a sample to the training set, and **3)** the rollout strategy, as discussed in Lagoudakis and Parr [2003b] and Dimitrakakis and Lagoudakis [2008b], here we consider its basic form in order to ease the analysis.

In DPI, at each iteration $k$, a new policy $\pi_{k+1}$ is computed from $\pi_k$, as the best approximation of $\mathcal{G}\pi_k$, by solving a cost-sensitive classification problem. More formally, DPI is based on the following loss function.

**Definition 27** *The loss function at iteration $k$ for a policy $\pi$ is denoted by $\ell_{\pi_k}(\cdot;\pi)$ and is defined as*

$$\ell_{\pi_k}(x;\pi) = \max_{a \in \mathcal{A}} Q^{\pi_k}(x, a) - Q^{\pi_k}\big(x, \pi(x)\big), \quad \forall x \in \mathcal{X}.$$

*Given a distribution $\rho$ over $\mathcal{X}$, we define the expected error as the expectation* *of the loss function $\ell_{\pi_k}(\cdot;\pi)$ according to $\rho$,*[5]

$$\mathcal{L}_{\pi_k}(\rho;\pi) = \int_{\mathcal{X}} \ell_{\pi_k}(x;\pi)\rho(dx) = \int_{\mathcal{X}} \Big[\max_{a\in\mathcal{A}} Q^{\pi_k}(x,a) - Q^{\pi_k}\big(x,\pi(x)\big)\Big]\rho(dx). \tag{3.2}$$

While in Lagoudakis and Parr [2003b] the goal is to minimize the number of misclassifications, i.e., they use a 0/1 loss function, DPI learns a policy which aims at minimizing the error $\mathcal{L}_{\pi_k}$. Similar to other classification-based RL algorithms [Fern et al., 2004, Langford and Zadrozny, 2005, Li et al., 2007], DPI does not focus on finding a uniformly accurate approximation of the actions taken by the greedy policy, but rather on finding actions leading to a similar performance. This is consistent with the final objective of policy iteration, which is to obtain a policy with similar performance to an optimal policy, and not necessarily one that takes actions similar to an optimal policy.[6]

As illustrated in Figure 4.3, for each state $x_i \in \mathcal{D}_k$ and for each action $a \in \mathcal{A}$, an estimate of the action-value function of the current policy is computed through $M$ independent rollouts. A $H$-horizon rollout of a policy $\pi_k$ for a state-action pair $(x_i, a)$ is

$$R^{\pi_k}(x_i,a) = r(x_i,a) + \sum_{t=1}^{H-1} \gamma^t r\big(x^t, \pi_k(x^t)\big), \tag{3.3}$$

where $x^t \sim p\big(\cdot\,|x^{t-1}, \pi_k(x^{t-1})\big)$ and $x^1 \sim p(\cdot|x_i,a)$. The action-value function estimation is then obtained by averaging $M$ independent rollouts $\{R_j^{\pi_k}(x_i,a)\}_{1\le j\le M}$ as

$$\widehat{Q}^{\pi_k}(x_i,a) = \frac{1}{M}\sum_{j=1}^{M} R_j^{\pi_k}(x_i,a). \tag{3.4}$$

[5] The expected error $\mathcal{L}_{\pi_k}(\rho;\pi)$ can be seen as the $L_{1,\rho}$-norm of the loss function.

[6] We refer the readers to Li et al. [2007] for a simple example in which a good approximation (in terms of the number of mismatch in selecting actions) of the greedy policy has a very poor performance w.r.t. it.

Given the outcome of the rollouts, the empirical loss is defined as follows.

**Definition 28** *For any $x \in \mathcal{D}_k$, the empirical loss function at iteration $k$ for a policy $\pi$ is*

$$\widehat{\ell}_{\pi_k}(x;\pi) = \max_{a\in\mathcal{A}} \widehat{Q}^{\pi_k}(x,a) - \widehat{Q}^{\pi_k}\big(x,\pi(x)\big),$$

*where $\widehat{Q}^{\pi_k}(x,a)$ is a $H$-horizon rollout estimation of the action-value of $\pi_k$ in $(x,a)$ as defined by Eqs. 4.15 and 3.4. Similar to Definition 27, the empirical error is defined as the average over states in $\mathcal{D}_k$ of the empirical loss,*[7]

$$\widehat{\mathcal{L}}_{\pi_k}(\widehat{\rho};\pi) = \frac{1}{N}\sum_{t=1}^{n}\Big[\max_{a\in\mathcal{A}} \widehat{Q}^{\pi_k}(x_i,a) - \widehat{Q}^{\pi_k}\big(x_i,\pi(x_i)\big)\Big],$$

*where $\widehat{\rho}$ is the empirical distribution induced by the samples in $\mathcal{D}_k$.*

Finally, DPI makes use of a classifier which returns a policy that minimizes the empirical error $\widehat{\mathcal{L}}_{\pi_k}(\widehat{\rho};\pi)$ over the policy space $\Pi$.

## 3.4 Finite-sample Analysis of DPI

In this section, we first provide a finite-sample analysis of the error incurred at each iteration of DPI in Theorem 31, and then show how this error is propagated through the iterations of the algorithm in Theorem 33. In the analysis, we explicitly assume that the action space contains only two actions, i.e., $\mathcal{A} = \{a_1, a_2\}$ and $|\mathcal{A}| = 2$. We will discuss this assumption and other theoretical and practical aspects of DPI in Section 3.6.

### 3.4.1 Error Bound at Each Iteration

Here we study the error incurred at each iteration $k$ of the DPI algorithm. Comparing the definition of the expected and empirical errors, we notice that

[7] Alternatively, the empirical error can be seen as the $L_{1,\widehat{\rho}}$-norm of the empirical loss.

there are three sources of error in the algorithm of Figure 4.3. The first one depends on the use of a finite number of samples, i.e., $N$ states in the rollout set, to approximate the expectation w.r.t. the distribution $\rho$. The second one is due to using rollouts with finite horizon $H$ to approximate the action-value function $Q^{\pi_k}$ of the current policy $\pi_k$. Finally, the third one depends on the use of $M$ rollouts to approximate the action-value function of the current policy for any of the $N$ states in the rollout set $\mathcal{D}_k$ and any action in the action space $\mathcal{A}$. Before stating our main result, i.e., Theorem 31, we prove bounds for the first and third sources of errors in Lemmas 62 and 63, and have a discussion on the effect of finite horizon rollouts to approximate the action-value function. Lemma 62 shows that the difference between the approximation obtained by averaging over the samples in the rollout set and the true expectation can be controlled and reduces to zero as the number of states in the rollout set $N$ grows.

**Lemma 29** *Let $\Pi$ be a policy space with finite VC-dimension $h = VC(\Pi) < \infty$ and $n > 0$ be the number of states in the rollout set $\mathcal{D}_k$, drawn i.i.d. from the state distribution $\rho$, then*

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

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

**Proof** Let $\mathcal{F}_k$ be the space of the loss functions at iteration $k$ induced by the policies in $\Pi$, i.e., $\mathcal{F}_k = \{\ell_{\pi_k}(\cdot;\pi)|\ \pi \in \Pi\}$. Note that all the functions $\ell_{\pi_k}(\cdot;\pi) \in \mathcal{F}_k$ are uniformly bounded by $2Q_{\max}$. By Pollard's inequality [Pollard, 1984], for the bounded space $\mathcal{F}_k$, we have

$$\mathbb{P}_{\mathcal{D}_k}\left[\sup_{\ell_{\pi_k}\in\mathcal{F}_k}\left|\frac{1}{N}\sum_{i=1}^{N}\ell_{\pi_k}(x_i) - \int\ell_{\pi_k}(x)\rho(dx)\right| > \epsilon\right] \leq 8\mathbb{E}\left[\mathcal{N}_1\left(\frac{\epsilon}{8},\mathcal{F}_k,X_1^N\right)\right]\exp\left(-\frac{N\epsilon^2}{128(2Q_{\max})^2}\right).$$

Note that at each iteration $k$, the policy $\pi_k$ is a random variable because it is the minimizer of the empirical error $\widehat{\mathcal{L}}_{\pi_{k-1}}(\widehat{\rho};\pi)$. However, $\pi_k$ depends only on the previous policies and rollout sets up to $\mathcal{D}_{k-1}$, and is completely independent of the samples in $\mathcal{D}_k$, thus Pollard's inequality applies. We now show how the covering number of the space $\mathcal{F}_k$ can be directly related to the VC-dimension of $\Pi$. First we rewrite the loss function as $\ell_{\pi_k}(x;\pi) = \mathbb{I}\left\{(\mathcal{G}\pi_k)(x) \neq \pi(x)\right\}\Delta^{\pi_k}(x)$, where

$$\Delta^{\pi_k}(x) = \max_{a\in\mathcal{A}} Q^{\pi_k}(x,a) - \min_{a'\in\mathcal{A}} Q^{\pi_k}(x,a') \qquad (3.5)$$

is the gap between the two actions (i.e., the regret of choosing the wrong action). Let $\bar{\Pi}$ be an $\frac{\epsilon}{2Q_{\max}}$-cover of $\Pi$ using the empirical distance defined by the number of different actions at the states $\{x_i\}_{1\le i\le N}$, then $\bar{\mathcal{F}}_k = \{\bar{\ell}_{\pi_k}(\cdot) = \ell_{\pi_k}(\cdot;\bar{\pi})|\bar{\pi}\in\bar{\Pi}\}$ is an $\epsilon$-cover of $\mathcal{F}_k$. In fact for any $\ell_{\pi_k}\in\mathcal{F}_k$, there exist a $\bar{\ell}_{\pi_k}\in\bar{\mathcal{F}}_k$ such that

$$\begin{aligned}\frac{1}{N}\sum_{i=1}^{N}\left|\ell_{\pi_k}(x_i) - \bar{\ell}_{\pi_k}(x_i)\right| &= \frac{1}{N}\sum_{i=1}^{N}\big|\mathbb{I}\left\{(\mathcal{G}\pi_k)(x_i) \neq \pi(x_i)\right\}\Delta^{\pi_k}(x_i) \\ &\quad - \mathbb{I}\left\{(\mathcal{G}\pi_k)(x_i) \neq \bar{\pi}(x_i)\right\}\Delta^{\pi_k}(x_i)\big| \\ &\le 2Q_{\max}\frac{1}{N}\sum_{i=1}^{N}\left|\mathbb{I}\left\{(\mathcal{G}\pi_k)(x_i) \neq \pi(x_i)\right\} - \mathbb{I}\left\{(\mathcal{G}\pi_k)(x_i) \neq \bar{\pi}(x_i)\right\}\right| \\ &= 2Q_{\max}\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left\{(\pi(x_i) \neq \bar{\pi}(x_i)\right\} \le 2Q_{\max}\frac{\epsilon}{2Q_{\max}} = \epsilon.\end{aligned}$$

Thus, we can now relate the covering number of $\mathcal{F}_k$ to the VC-dimension of $\Pi$

$$\mathcal{N}_1\left(\frac{\epsilon}{8},\mathcal{F}_k,X_1^N\right) \le \mathcal{N}_1\left(\frac{\epsilon}{16Q_{\max}},\Pi,X_1^N\right) \le S_\Pi(n) \le \left(\frac{en}{h}\right)^h,$$

where $S_\Pi(n)$ is the growth function of $\Pi$ and the last inequality follows from Sauer's lemma. Since $\mathcal{L}_{\pi_k}(\widehat{\rho};\pi) = \frac{1}{N}\sum_{i=1}^{N}\ell_{\pi_k}(x_i;\pi)$ and $\mathcal{L}_{\pi_k}(\rho;\pi) = \int \ell_{\pi_k}(x;\pi)\rho(dx)$, the final statement is obtained by inverting the Pollard's bound. ■

The second source of error in the algorithm of Figure 4.3 is due to the use of finite horizon rollout estimates of the action-value function on the states in the rollout set. We define the true action-value for a state-action pair $(x, a)$ with a finite horizon $H$ as

$$Q_H^{\pi_k}(x,a) = \mathbb{E}\left[r(x,a) + \sum_{t=1}^{H-1} \gamma^t r\big(x^t, \pi_k(x^t)\big)\right].$$

It is easy to see that the $H$-horizon rollout estimates are stochastic estimations of $Q_H^{\pi_k}(x,a)$ which in turn satisfy

$$|Q^{\pi_k}(x,a) - Q_H^{\pi_k}(x,a)| = \left|\mathbb{E}\left[\sum_{t=H}^{\infty} \gamma^t r\big(x^t, \pi_k(x^t)\big)\right]\right| \leq \gamma^H Q_{\max}. \tag{3.6}$$

In the proof of the main theorem we also need to bound the difference between the action values (of the $N$ states in the rollout set $\mathcal{D}_k$ and all the actions in the action space $\mathcal{A}$) estimated with $M$ rollouts and their true values. We thus report the following lemma to bound this source of error.

**Lemma 30** *Let* $\Pi$ *be a policy space with finite VC-dimension* $h = VC(\Pi) < \infty$ *and* $x_1, \dots, x_N$ *be an arbitrary sequence of states. In each state we simulate* $M$ *independent truncated rollouts, then*

$$\mathbb{P}\left[\sup_{\pi\in\Pi}\Big|\frac{1}{n}\sum_{t=1}^{n}\frac{1}{M}\sum_{j=1}^{M} R_j^{\pi_k}(x_i,\pi(x_i)) - \frac{1}{n}\sum_{t=1}^{n} Q_H^{\pi_k}(x_i,\pi(x_i))\Big| > \epsilon\right] \leq \delta\ ,$$

*with* $\epsilon = 8(1-\gamma^H)Q_{\max}\sqrt{\frac{2}{MN}\left(h\log\frac{eMN}{h} + \log\frac{8}{\delta}\right)}$ .

**Proof** Similar to the proof of Lemma 62, we rely on the Pollard's inequality to prove the statement. We first introduce a sequence of random events $\omega_{ij}$ such that for any $i = 1, \dots, N$ the event $\omega_{ij}$ is independently drawn from a suitable distribution $\nu_i$. As a result, we may rewrite the rollout random variables as $R_j^{\pi_k}\big(x_i,\pi(x_i)\big) = R^{\pi_k}(\omega_{ij};\pi)$ and the statement of the theorem as

$$\mathbb{P}\left[\sup_{\pi\in\Pi}\Big|\frac{1}{MN}\sum_{i,j}R^{\pi_k}(\omega_{ij};\pi) - \frac{1}{MN}\sum_{i,j}\mathbb{E}_{\nu_i}\big[R^{\pi_k}(\omega_{ij};\pi)\big]\Big| > \epsilon\right] \leq \delta.$$

Let $\mathcal{H}_k$ be the space of the rollout functions induced by the policies in $\Pi$ at iteration $k$, i.e., $\mathcal{H}_k = \{R^{\pi_k}(\cdot;\pi)|\ \pi\in\Pi\}$. Note that all the functions $R^{\pi_k}(\cdot;\pi)\in\mathcal{H}_k$ are uniformly bounded by $(1-\gamma^H)Q_{\max}$. By Pollard's inequality [Pollard, 1984], for the bounded space $\mathcal{H}_k$, we have[8]

$$\mathbb{P}\left[\sup_{\pi\in\Pi}\Big|\frac{1}{MN}\sum_{i,j}R^{\pi_k}(\omega_{ij};\pi) - \frac{1}{MN}\sum_{i,j}\mathbb{E}_{\nu_i}[R^{\pi_k}(\omega_{ij};\pi)]\Big| > \epsilon\right] \\ \leq 8\mathbb{E}\left[\mathcal{N}_1\left(\frac{\epsilon}{8},\mathcal{H}_k,\omega_1^{MN}\right)\right]\exp\left(-\frac{MN\epsilon^2}{128(1-\gamma^H)^2Q_{\max}^2}\right).$$

We now show how the covering number of the space $\mathcal{H}_k$ is related to the VC-dimension of $\Pi$. Let $\bar{\Pi}$ be an $\frac{\epsilon}{2(1-\gamma^H)Q_{\max}}$-cover of $\Pi$ using the empirical distance defined at the states $\{x_i\}_{1\leq i\leq N}$, then $\bar{\mathcal{H}}_k = \{\bar{R}^{\pi_k}(\cdot) = R^{\pi_k}(\cdot;\bar{\pi})|\bar{\pi}\in\bar{\Pi}\}$ is an $\epsilon$-cover of $\mathcal{H}_k$. In fact for any $R^{\pi_k}\in\mathcal{H}_k$, there exist a $\bar{R}^{\pi_k}\in\bar{\mathcal{H}}_k$ such that

$$\frac{1}{MN}\sum_{i,j}\big|R^{\pi_k}(\omega_{ij}) - \bar{R}^{\pi_k}(\omega_{ij})\big| = \frac{1}{MN}\sum_{t=1}^{n}\sum_{j=1}^{M}\big|R_j^{\pi_k}\big(x_i,\pi(x_i)\big) - R_j^{\pi_k}\big(x_i,\bar{\pi}(x_i)\big)\big| \\ \leq 2(1-\gamma^H)Q_{\max}\frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left\{\pi(x_i)\neq\bar{\pi}(x_i)\right\} \leq 2(1-\gamma^H)Q_{\max}\frac{\epsilon}{2(1-\gamma^H)Q_{\max}} = \epsilon.$$

[8] Note that since here the samples are independent but not identically distributed, we use a slight variation of the standard Pollard's inequality. We refer the reader to the proof of Pollard's inequality (e.g., Pollard 1984 or Devroye et al. 1996) to see that the standard proof can be easily extended to this case.

We can now relate the covering number of $\mathcal{F}_k$ to the VC-dimension of $\Pi$

$$\mathcal{N}_1\left(\frac{\epsilon}{8}, \mathcal{F}_k, \omega_1^{MN}\right) \leq \mathcal{N}_1\left(\frac{\epsilon}{16(1-\gamma^H)Q_{\max}}, \Pi, \omega_1^{MN}\right) \leq S_\Pi(MN) \leq \left(\frac{eMN}{h}\right)^h,$$

where $S_\Pi(n)$ is the growth function of $\Pi$ and the last inequality follows from Sauer's lemma. The final statement is obtained by inverting the Pollard's bound. ■

We are now ready to prove the main result of this section. We show a high probability bound on $\mathcal{L}_{\pi_k}(\rho; \pi_{k+1})$, the expected error at each iteration $k$ of the DPI algorithm.

**Theorem 31** *Let $\Pi$ be a policy space with finite VC-dimension $h = VC(\Pi) < \infty$ and $\rho$ be a distribution over the state space $\mathcal{X}$. Let $n$ be the number of states in $\mathcal{D}_k$ drawn i.i.d. from $\rho$ at each iteration, $H$ be the horizon of the rollouts, and $M$ be the number of rollouts per state-action pair used in the estimation of the action-value functions. Let $\pi_{k+1} = \arg\min_{\pi\in\Pi} \widehat{\mathcal{L}}_{\pi_k}(\widehat{\rho}; \pi)$ be the policy computed at the k'th iteration of DPI . Then, for any $\delta > 0$, we have*

$$\mathcal{L}_{\pi_k}(\rho; \pi_{k+1}) \leq \inf_{\pi\in\Pi} \mathcal{L}_{\pi_k}(\rho; \pi) + 2(\epsilon_1 + \epsilon_2 + \gamma^H Q_{\max}), \qquad (3.7)$$

*with probability $1 - \delta$, where*

$$\epsilon_1 = 16Q_{\max}\sqrt{\frac{2}{n}\left(h\log\frac{en}{h} + \log\frac{32}{\delta}\right)} \quad \textit{and} \quad \epsilon_2 = 8(1-\gamma^H)Q_{\max}\sqrt{\frac{2}{Mn}\left(h\log\frac{eMN}{h} + \log\frac{32}{\delta}\right)}.$$

**Remark 1** The bound in Eq. 3.7 can be decomposed into an approximation error $\inf_{\pi\in\Pi} \mathcal{L}_{\pi_k}(\rho; \pi)$ and an estimation error consisting of three terms $\epsilon_1$, $\epsilon_2$, and $\gamma^H Q_{\max}$. This is similar to generalization bounds in classification, where the approximation error is the distance between the target function (here the greedy policy w.r.t. $\pi_k$) and the function space $\Pi$. The first estimation term, $\epsilon_1$, grows with the capacity of $\Pi$, measured by its VC-dimension $h$, and decreases with the number of sampled states $n$. Thus in order to avoid overfitting, we should have $n \gg h$. The second estimation term, $\epsilon_2$, comes from the error in the estimation of the action-values due to the finite number of rollouts $M$. It is important to note the nice rate of $1/\sqrt{Mn}$ instead of $1/\sqrt{M}$. This is due to the fact that we do not need a uniformly good estimation of the action-value function at all sampled states, but only an averaged estimation of those values at the sampled points. An important consequence of this is that the algorithm works perfectly well if we consider only $M = 1$ rollout per state-action. Therefore, given a fixed budget (number of rollouts per iteration) and a fixed rollout horizon $H$, the best allocation of $M$ and $n$ would be to choose $M = 1$ and sample as many states as possible, thus, reducing the risk of overfitting. The third estimation term, $\gamma^H Q_{\max}$, is due to the fact that we consider a finite horizon $H$ for the rollouts. This term decreases as the rollout horizon $H$ grows.

**Remark 2** In Remark 1, we considered the tradeoff between the number of states, $N$, and the number of rollouts at each state-action pair, $M$, when a finite budget (number of rollouts per iteration) is given. It is also interesting to analyze the tradeoff with the rollout horizon, $H$, when the number of interactions with the generative model is fixed to a maximum value $S = N \times M \times H$. The term $\gamma^H$ decreases exponentially with a rate depending on $\gamma$, thus, it easy to see that by setting $M = 1$, a rough optimization of the bound in Theorem 31 leads to $H = O(\frac{\log S}{\log 1/\gamma})$ and $N = O(S/H)$. Similar to the tradeoff between $M$ and $N$, this suggests that most of the resources should be allocated so as to have a large number of states, while the rollouts may have a fairly short horizon. Nonetheless, it is clear from the value of $H$ that the discount factor is critical, and when it approaches 1 the horizon increases correspondingly.

**Proof** Let $a^*(x) = \arg\max_{a\in\mathcal{A}} Q^{\pi_k}(x,a)$ be the greedy action in state $x$.[9] We prove the following series of inequalities:

$$
\begin{aligned}
\mathcal{L}_{\pi_k}(\rho;\pi_{k+1}) &\overset{\text{(a)}}{\leq} \mathcal{L}_{\pi_k}(\widehat{\rho};\pi_{k+1}) + \epsilon_1 && \text{w.p. } 1-\delta' \\
&= \frac{1}{n}\sum_{t=1}^{n}\Big[Q^{\pi_k}(x_i,a^*) - Q^{\pi_k}\big(x_i,\pi_{k+1}(x_i)\big)\Big] + \epsilon_1 \\
&\overset{\text{(b)}}{\leq} \frac{1}{n}\sum_{t=1}^{n}\Big[Q^{\pi_k}(x_i,a^*) - Q^{\pi_k}_H\big(x_i,\pi_{k+1}(x_i)\big)\Big] + \epsilon_1 + \gamma^H Q_{\max} && \text{w.p. } 1-\delta' \\
&\overset{\text{(c)}}{\leq} \frac{1}{n}\sum_{t=1}^{n}\Big[Q^{\pi_k}(x_i,a^*) - \widehat{Q}^{\pi_k}\big(x_i,\pi_{k+1}(x_i)\big)\Big] + \epsilon_1 + \epsilon_2 + \gamma^H Q_{\max} && \text{w.p. } 1-2\delta' \\
&\overset{\text{(d)}}{\leq} \frac{1}{n}\sum_{t=1}^{n}\Big[Q^{\pi_k}(x_i,a^*) - \widehat{Q}^{\pi_k}\big(x_i,\pi^*(x_i)\big)\Big] + \epsilon_1 + \epsilon_2 + \gamma^H Q_{\max} \\
&\overset{\text{(e)}}{\leq} \frac{1}{n}\sum_{t=1}^{n}\Big[Q^{\pi_k}(x_i,a^*) - Q^{\pi_k}_H\big(x_i,\pi^*(x_i)\big)\Big] + \epsilon_1 + 2\epsilon_2 + \gamma^H Q_{\max} && \text{w.p. } 1-3\delta' \\
&\overset{\text{(f)}}{\leq} \frac{1}{n}\sum_{t=1}^{n}\Big[Q^{\pi_k}(x_i,a^*) - Q^{\pi_k}\big(x_i,\pi^*(x_i)\big)\Big] + \epsilon_1 + 2(\epsilon_2 + \gamma^H Q_{\max}) && \text{w.p. } 1-3\delta' \\
&= \mathcal{L}_{\pi_k}(\widehat{\rho};\pi^*) + \epsilon_1 + 2(\epsilon_2 + \gamma^H Q_{\max}) \\
&\overset{\text{(g)}}{\leq} \mathcal{L}_{\pi_k}(\rho;\pi^*) + 2(\epsilon_1 + \epsilon_2 + \gamma^H Q_{\max}) && \text{w.p. } 1-4\delta' \\
&= \inf_{\pi'\in\Pi} \mathcal{L}_{\pi_k}(\rho;\pi') + 2(\epsilon_1 + \epsilon_2 + \gamma^H Q_{\max}).
\end{aligned}
$$

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

**(a)** It is an immediate application of Lemma 62, bounding the difference between $\mathcal{L}_{\pi_k}(\rho;\pi)$ and $\mathcal{L}_{\pi_k}(\widehat{\rho};\pi)$ for any policy $\pi \in \Pi$.
**(b)** We use the inequality in Eq. 3.6.

[9] To simplify the notation, we remove the dependency of $a^*$ on states and use $a^*$ instead of $a^*(x)$ in the following.

**(c)** Here we introduce the estimated action-value function $\widehat{Q}^{\pi_k}$ by bounding

$$\sup_{\pi\in\Pi}\left[\frac{1}{n}\sum_{t=1}^{n}\widehat{Q}^{\pi_k}\big(x_i,\pi(x_i)\big)-\frac{1}{n}\sum_{t=1}^{n}Q_H^{\pi_k}\big(x_i,\pi(x_i)\big)\right],$$

the maximum over all the policies in the policy space[10] of the difference between the true action-value function with horizon $H$ and its rollout estimates averaged over the states in the rollout set $\mathcal{D}_k=\{x_i\}_{i=1}^n$. We bound this term using the result of Lemma 63.

**(d)** From the definition of $\pi_{k+1}$ in the DPI algorithm (see Figure 4.3), we have

$$\pi_{k+1}=\operatorname*{arg\,min}_{\pi\in\Pi}\widehat{\mathcal{L}}_{\pi_k}(\widehat{\rho};\pi)=\operatorname*{arg\,max}_{\pi\in\Pi}\frac{1}{n}\sum_{t=1}^{n}\widehat{Q}^{\pi_k}\big(x_i,\pi(x_i)\big),$$

thus, $-\frac{1}{n}\sum_{t=1}^{n}\widehat{Q}^{\pi_k}\big(x_i,\pi_{k+1}(x_i)\big)$ can be maximized by replacing $\pi_{k+1}$ with any other policy, particularly with

$$\pi^*=\operatorname*{arg\,inf}_{\pi'\in\Pi}\int_{\mathcal{X}}\Big(\max_{a\in\mathcal{A}}Q^{\pi_k}(x,a)-Q^{\pi_k}\big(x,\pi'(x)\big)\Big)\rho(dx).$$

**(e)-(g)** The final result follows by using Definition 32 and by applying the Chernoff-Hoeffding inequality, the inequality of Eq. 3.6, and the regression generalization bound. ■

### 3.4.2 Error Propagation

In this section, we first show how the expected error is propagated through the iterations of DPI. We then analyze the error between the value function of

[10] The supremum over all the policies in the policy space $\Pi$ is due to the fact that $\pi_{k+1}$ is a random object, whose randomness comes from all the randomly generated samples at the $k$'th iteration (i.e., the states in the rollout set and all the generated rollouts).

the policy obtained by DPI after $K$ iterations and the optimal value function in $\mu$-norm, where $\mu$ is a distribution used to assess the performance of the algorithm which might be different from the sampling distribution $\rho$.

Before stating the main result, we define the *inherent greedy error* of a policy space $\Pi$.

**Definition 32** *We define the inherent greedy error of a policy space* $\Pi \subseteq \mathcal{B}^{\pi}(\mathcal{X})$ *as*

$$d(\Pi, \mathcal{G}\Pi) = \sup_{\pi \in \Pi} \inf_{\pi' \in \Pi} \mathcal{L}_{\pi}(\rho; \pi').$$

In other words, the inherent greedy error is the worst expected error that a error-minimizing policy $\pi' \in \Pi$ can incur in approximating the greedy policy $\mathcal{G}\pi$, $\pi \in \Pi$. This measures how well $\Pi$ is able to approximate policies that are greedy w.r.t. any policy in $\Pi$.

Let $P^{\pi}$ be the transition kernel for policy $\pi$, i.e., $P^{\pi}(dy|x) = p\big(dy|x, \pi(x)\big)$. It defines two related operators: a right-linear operator, $P^{\pi}\cdot$, which maps any $V \in \mathcal{B}^{V}(\mathcal{X}; V_{\max})$ to $(P^{\pi}V)(x) = \int V(y)P^{\pi}(dy|x)$, and a left-linear operator, $\cdot P^{\pi}$, that returns $(\mu P^{\pi})(dy) = \int P^{\pi}(dy|x)\mu(dx)$ for any distribution $\mu$ over $\mathcal{X}$.

From the definitions of $\ell_{\pi_k}$, $\mathcal{T}^{\pi}$, and $\mathcal{T}$, we have $\ell_{\pi_k}(\pi_{k+1}) = \mathcal{T}V^{\pi_k} - \mathcal{T}^{\pi_{k+1}}V^{\pi_k}$. We deduce the following pointwise inequalities:

$$\begin{aligned} V^{\pi_k} - V^{\pi_{k+1}} &= \mathcal{T}^{\pi_k}V^{\pi_k} - \mathcal{T}^{\pi_{k+1}}V^{\pi_k} + \mathcal{T}^{\pi_{k+1}}V^{\pi_k} - \mathcal{T}^{\pi_{k+1}}V^{\pi_{k+1}} \\ &\leq \ell_{\pi_k}(\pi_{k+1}) + \gamma P^{\pi_{k+1}}(V^{\pi_k} - V^{\pi_{k+1}}) \;, \end{aligned}$$

which gives us $V^{\pi_k} - V^{\pi_{k+1}} \leq (I - \gamma P^{\pi_{k+1}})^{-1}\ell_{\pi_k}(\pi_{k+1})$. Since $\mathcal{T}V^{\pi_k} \geq \mathcal{T}^{\pi^*}V^{\pi_k}$, we also have

$$\begin{aligned} V^{*} - V^{\pi_{k+1}} &= \mathcal{T}V^{*} - \mathcal{T}V^{\pi_k} + \mathcal{T}V^{\pi_k} - \mathcal{T}^{\pi_{k+1}}V^{\pi_k} + \mathcal{T}^{\pi_{k+1}}V^{\pi_k} - \mathcal{T}^{\pi_{k+1}}V^{\pi_{k+1}} \\ &\leq \gamma P^{*}(V^{*} - V^{\pi_k}) + \ell_{\pi_k}(\pi_{k+1}) + \gamma P^{\pi_{k+1}}(V^{\pi_k} - V^{\pi_{k+1}}) \;, \end{aligned}$$

which yields

$$\begin{aligned} V^* - V^{\pi_{k+1}} &\leq \gamma P^*(V^* - V^{\pi_k}) + \left[\gamma P^{\pi_{k+1}}(I - \gamma P^{\pi_{k+1}})^{-1} + I\right]\ell_{\pi_k}(\pi_{k+1}) \\ &= \gamma P^*(V^* - V^{\pi_k}) + (I - \gamma P^{\pi_{k+1}})^{-1}\ell_{\pi_k}(\pi_{k+1}) \ . \end{aligned}$$

Finally, by defining the operator $E_k = (I - \gamma P^{\pi_{k+1}})^{-1}$, which is well defined since $P^{\pi_{k+1}}$ is a stochastic kernel and $\gamma < 1$, and by induction, we obtain

$$V^* - V^{\pi_K} \leq (\gamma P^*)^K(V^* - V^{\pi_0}) + \sum_{k=0}^{K-1}(\gamma P^*)^{K-k-1}E_k\ell_{\pi_k}(\pi_{k+1}) \ . \tag{3.8}$$

Eq. 3.8 shows how the error at each iteration $k$ of DPI, $\ell_{\pi_k}(\pi_{k+1})$, is propagated through the iterations and appears in the final error of the algorithm: $V^* - V^{\pi_K}$. Since we are interested in bounding the final error in $\mu$-norm, which might be different than the sampling distribution $\rho$, we use one of the following assumptions:

**Assumption 5** *For any policy $\pi$ and any non-negative integers $s$ and $t$, there exists a constant $C_{\mu,\rho}(s,t) < \infty$ such that $\mu(P^*)^s(P^\pi)^t \leq C_{\mu,\rho}(s,t)\rho$. We define $C_{\mu,\rho} = (1-\gamma)^2\sum_{s=0}^{\infty}\sum_{t=0}^{\infty}\gamma^{s+t}C_{\mu,\rho}(s,t)$.*

**Assumption 6** *For any $x \in \mathcal{X}$ and any $a \in \mathcal{A}$, there exist a constant $C_\rho < \infty$ such that $p(\cdot|x,a) \leq C_\rho\rho(\cdot)$.*

Note that *concentrability coefficients* similar to $C_{\mu,\rho}$ and $C_\rho$ were previously used in the $L_p$-analysis of fitted value iteration [Munos, 2007, Munos and Szepesvári, 2008] and approximate policy iteration [Antos et al., 2008].[11] We now state our main result.

**Theorem 33** *Let $\Pi$ be a policy space with finite VC-dimension $h$ and $\pi_K$ be the policy generated by DPI after $K$ iterations. Let $M$ be the number of*

[11] See also Farahmand et al. [2010] for a more refined analysis.

*rollouts per state-action and $n$ be the number of samples drawn i.i.d. from a distribution $\rho$ over $\mathcal{X}$ at each iteration of DPI. Then, for any $\delta > 0$, we have*

$$||V^* - V^{\pi_K}||_{1,\mu} \leq \frac{C_{\mu,\rho}}{(1-\gamma)^2}\Big[d(\Pi, \mathcal{G}\Pi) + 2(\epsilon_1 + \epsilon_2 + \gamma^H Q_{\max})\Big] + \frac{2\gamma^K R_{\max}}{1-\gamma}, \quad \textit{(Assumption 1)}$$

$$||V^* - V^{\pi_K}||_{\infty} \leq \frac{C_{\rho}}{(1-\gamma)^2}\Big[d(\Pi, \mathcal{G}\Pi) + 2(\epsilon_1 + \epsilon_2 + \gamma^H Q_{\max})\Big] + \frac{2\gamma^K R_{\max}}{1-\gamma}, \quad \textit{(Assumption 2)}$$

*with probability $1-\delta$, where*

$$\epsilon_1 = 16 Q_{\max}\sqrt{\frac{2}{n}\left(h\log\frac{en}{h} + \log\frac{32K}{\delta}\right)} \quad \textit{and} \quad \epsilon_2 = 8(1-\gamma^H)Q_{\max}\sqrt{\frac{2}{Mn}\left(h\log\frac{eMN}{h} + \log\frac{32K}{\delta}\right)}\,.$$

**Proof** We have $C_{\mu,\rho} \leq C_\rho$ for any $\mu$. Thus, if the $L_1$-bound holds for any $\mu$, choosing $\mu$ to be a Dirac at each state implies that the $L_\infty$-bound holds as well. Hence, we only need to prove the $L_1$-bound. By taking the absolute value point-wise in Eq. 3.8 we obtain

$$|V^* - V^{\pi_K}| \leq (\gamma P^*)^K |V^* - V^{\pi_0}| + \sum_{k=0}^{K-1} (\gamma P^*)^{K-k-1}(I - \gamma P^{\pi_{k+1}})^{-1} |\ell_{\pi_k}(\pi_{k+1})|\,.$$

From the fact that $|V^* - V^{\pi_0}| \leq \frac{2}{1-\gamma} R_{\max}\mathbf{1}$, and by integrating both sides w.r.t. $\mu$, and using Assumption 5 we have

$$||V^* - V^{\pi_K}||_{1,\mu} \leq \frac{2\gamma^K}{1-\gamma} R_{\max} + \sum_{k=0}^{K-1}\sum_{t=0}^{\infty} \gamma^{K-k-1}\gamma^t C_{\mu,\rho}(K-k-1, t)\mathcal{L}_{\pi_k}(\rho; \pi_{k+1})\,.$$

From the definition of $C_{\mu,\rho}$ we obtain

$$||V^* - V^{\pi_K}||_{1,\mu} \leq \frac{2\gamma^K}{1-\gamma} R_{\max} + \frac{C_{\mu,\rho}}{(1-\gamma)^2} \max_{0 \leq k \leq K} \mathcal{L}_{\pi_k}(\rho; \pi_{k+1})\,.$$

By bounding $\mathcal{L}_{\pi_k}(\rho; \pi_{k+1})$ using Theorem 31 with a union bound argument over the $K$ iterations and the definition of the inherent greedy error the claim follows. ■

## 3.5 Approximation Error

In Section 3.4.2, we analyzed how the expected error at each iteration $k$ of DPI, $\mathcal{L}_{\pi_k}(\rho, \pi_{k+1})$, propagates through iterations. The final approximation error term in Theorem 33 is the inherent greedy error of Definition 32, $d(\Pi, \mathcal{G}\Pi)$, which depends on the MDP and the richness of the policy space $\Pi$. The main question in this section is whether this approximation error can be made small by increasing the capacity of the policy space $\Pi$. The answer is not obvious because when the space of policies, $\Pi$, grows, it can better approximate any greedy policy w.r.t. a policy in $\Pi$, however, the number of such greedy policies grows as well. We start our analysis of this approximation error by introducing the notion of *universal family of policy spaces.*

**Definition 34** *A sequence of policy spaces $\{\Pi_n\}$ is a universal family of policy spaces, if there exists a sequence of real numbers $\{\beta_n\}$ with $\lim_{n\to\infty} \beta_n = 0$, such that for any $n > 0$, $\Pi_n$ is induced by a partition $P_n = \{\mathcal{X}_i\}_{i=1}^{S_n}$ over the state space $\mathcal{X}$ (i.e., for each $S_n$-tuple $(b_1, \dots, b_{S_n})$ with $b_i \in \{0, 1\}$, there exists a policy $\pi \in \Pi_n$ such that $\pi(x) = b_i$ for all $x \in \mathcal{X}_i$ and for all $i \in \{1, \dots, S_n\}$) such that*

$$\max_{1\leq i\leq S_n} \max_{x,y\in\mathcal{X}_i} ||x - y|| \leq \beta_n.$$

This definition requires that for any $n > 0$, $\Pi_n$ be the space of policies induced by a partition $P_n$, and the diameters of the elements $\mathcal{X}_i$ of this partition shrink to zero as $n$ goes to infinity. The main property of such a sequence of spaces is that any fixed policy $\pi$ can be approximated arbitrary well by policies of $\Pi_n$ when $n \to \infty$. Although other definitions of universality could be used, Definition 34 seems natural and it is satisfied by widely-used classifiers such as $k$-nearest neighbor, uniform grid, and histogram.

In the next section, we first show that the universality of a policy space (Definition 34) does not guarantee that $d(\Pi_n, \mathcal{G}\Pi_n)$ converges to zero in a general MDP. In particular, we present a MDP in which $d(\Pi_n, \mathcal{G}\Pi_n)$ is constant (does not depend on $n$) even when $\{\Pi_n\}$ is a universal family of classifiers. We then prove that in Lipschitz MDPs, $d(\Pi_n, \mathcal{G}\Pi_n)$ converges to zero for a universal family of policy spaces.

### 3.5.1 Counterexample

In this section, we illustrate a simple example in which $d(\Pi_n, \mathcal{G}\Pi_n)$ does not go to zero, even when $\{\Pi_n\}$ is a universal family of classifiers. We consider a MDP with state space $\mathcal{X} = [0, 1]$, action space $\mathcal{A} = \{0, 1\}$, and the following transitions and rewards

$$x_{t+1} = \begin{cases} \min(x_t + 0.5, 1) & \text{if } \ a = 1, \\ x_t & \text{otherwise,} \end{cases} \qquad r(x, a) = \begin{cases} 0 & \text{if } \ x = 1, \\ R_1 & \text{else if } \ a = 1, \\ R_0 & \text{otherwise,} \end{cases}$$

$$\text{where } \ (1 - \gamma^2)R_1 < R_0 < R_1 \ . \tag{3.9}$$

We consider the policy space $\Pi_n$ of piecewise constant policies obtained by uniformly partitioning the state space $\mathcal{X}$ into $n$ intervals. This family of policy spaces is universal. The inherent greedy error of $\Pi_n$, $d(\Pi_n, \mathcal{G}\Pi_n)$, can be decomposed into the sum of the expected errors at each interval

$$d(\Pi_n, \mathcal{G}\Pi_n) = \sup_{\pi \in \Pi_n} \inf_{\pi' \in \Pi_n} \sum_{i=1}^{n} \mathcal{L}_\pi^{(i)}(\rho; \pi') \ ,$$

where $\mathcal{L}_\pi^{(i)}(\rho; \pi')$ is the same as $\mathcal{L}_\pi(\rho; \pi')$, only the integral is over the $i$'th interval instead of the entire state space $\mathcal{X}$. In the following we show that for the MDP and the universal class of policies considered here, $d(\Pi_n, \mathcal{G}\Pi_n)$ does not converge to zero as $n$ grows.

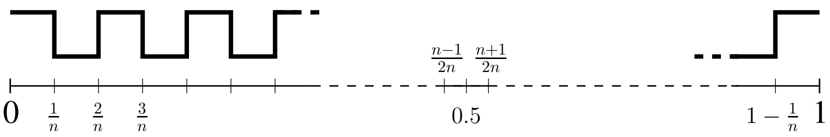


Figure 3.2: The policy used in the counterexample. It is one in odd and zero in even intervals. Note that the number of intervals, $n$, is assumed to be odd.

Let $n$ be odd and $\pi \in \Pi_n$ be one in odd and zero in even intervals (see Figure 3.2). For any $x > 0.5$, the agent either stays in the same state forever by taking action 0, or goes out of bound in one step by taking action 1. Thus, given the assumption of Eq. 3.9, it can be shown that for any $x$ belonging to the intervals $i \geq \frac{n+1}{2}$ (the interval containing 0.5 and above), $(\mathcal{G}\pi)(x) = 0$. This means that there exists a policy $\pi' \in \Pi_n$ such that $\mathcal{L}_\pi^{(i)}(\rho; \pi') = 0$ for all the intervals $i \geq \frac{n+1}{2}$. However, $\mathcal{G}\pi$ does not remain constant in the intervals $i \leq \frac{n-1}{2}$, and changes its value in the middle of the interval. Using Eq. 3.9, we can show that

$$\inf_{\pi' \in \Pi_n} \sum_{i=1}^{n} \mathcal{L}_\pi^{(i)}(\rho; \pi') = C\Big(1 + \frac{1}{1-\gamma}\Big)\frac{n-1}{8n} \geq \frac{C}{16}\Big(1 + \frac{1}{1-\gamma}\Big) \ ,$$

where $C = \min\{(1-\gamma)(R_1 - R_0), R_0 - (1-\gamma^2)R_1\}$. This means that for any odd $n$, it is always possible to find a policy $\pi \in \Pi_n$ such that $\inf_{\pi' \in \Pi_n} \mathcal{L}_\pi(\rho; \pi')$ is lower bounded by a constant independent of $n$, thus $\lim_{n\to\infty} d(\Pi_n, \mathcal{G}\Pi_n) \neq 0$.

### 3.5.2 Lipschitz MDPs

In this section, we prove that for Lipschitz MDPs, $d(\Pi_n, \mathcal{G}\Pi_n)$ goes to zero when $\{\Pi_n\}$ is a universal family of classifiers. We start by defining a Lipschitz MDP.

**Definition 35** *A MDP is Lipschitz if both its transition probability and re-*

*ward functions are Lipschitz, i.e.,* $\forall (B, x, x', a) \in \mathcal{B}(\mathcal{X}) \times \mathcal{X} \times \mathcal{X} \times \mathcal{A}$

$$\begin{aligned} |r(x,a) - r(x',a)| &\leq L_r ||x - x'||, \\ |p(B|x,a) - p(B|x',a)| &\leq L_p ||x - x'||, \end{aligned}$$

*with* $L_r$ *and* $L_p$ *being the Lipschitz constants of the transitions and reward, respectively.*

An important property of Lipschitz MDPs is that for any function $Q \in \mathcal{B}(\mathcal{X} \times \mathcal{A}; Q_{\max})$, the function obtained by applying the Bellman operator $\mathcal{T}^\pi$ to $Q(\cdot, a)$, $(\mathcal{T}^\pi Q)(\cdot, a)$, is Lipschitz with constant $L = (L_r + \gamma Q_{\max} L_p)$, for any action $a \in \mathcal{A}$. As a result, the function $Q^\pi(\cdot, a)$, which is the unique fixed point of the Bellman operator $\mathcal{T}^\pi$, is Lipschitz with constant $L$, for any policy $\pi \in \mathcal{B}^\pi(\mathcal{X})$ and any action $a \in \mathcal{A}$.

**Theorem 36** *Let* $\mathcal{M}$ *be a Lipschitz MDP with* $|\mathcal{A}| = 2$ *and* $\{\Pi_n\}$ *be a universal family of policy spaces (Definition 34). Then* $\lim_{n \to \infty} d(\Pi_n, \mathcal{G}\Pi_n) = 0$.

**Proof**

$$\begin{aligned}
d(\Pi_n, \mathcal{G}\Pi_n) &= \sup_{\pi \in \Pi_n} \inf_{\pi' \in \Pi_n} \int_{\mathcal{X}} \ell_\pi(x; \pi')\rho(dx) \\
&\overset{\text{(a)}}{=} \sup_{\pi \in \Pi_n} \inf_{\pi' \in \Pi_n} \int_{\mathcal{X}} \mathbb{I}\left\{(\mathcal{G}\pi)(x) \neq \pi'(x)\right\} \Delta^\pi(x)\rho(dx) \\
&\overset{\text{(b)}}{=} \sup_{\pi \in \Pi_n} \inf_{\pi' \in \Pi_n} \sum_{i=1}^{S_n} \int_{\mathcal{X}_i} \mathbb{I}\left\{(\mathcal{G}\pi)(x) \neq \pi'(x)\right\} \Delta^\pi(x)\rho(dx) \\
&\overset{\text{(c)}}{=} \sup_{\pi \in \Pi_n} \sum_{i=1}^{S_n} \min_{a \in \mathcal{A}} \int_{\mathcal{X}_i} \mathbb{I}\left\{(\mathcal{G}\pi)(x) \neq a\right\} \Delta^\pi(x)\rho(dx) \\
&\overset{\text{(d)}}{\leq} \sup_{\pi \in \Pi_n} \sum_{i=1}^{S_n} \min_{a \in \mathcal{A}} \int_{\mathcal{X}_i} \mathbb{I}\left\{(\mathcal{G}\pi)(x) \neq a\right\} 2L \inf_{y: \Delta^\pi(y)=0} ||x - y|| \, \rho(dx) \\
&\overset{\text{(e)}}{\leq} 2L \sup_{\pi \in \Pi_n} \sum_{i=1}^{S_n} \min_{a \in \mathcal{A}} \int_{\mathcal{X}_i} \mathbb{I}\left\{(\mathcal{G}\pi)(x) \neq a\right\} \beta_n\rho(dx) \\
&\overset{\text{(f)}}{\leq} 2L\beta_n \sum_{i=1}^{S_n} \int_{\mathcal{X}_i} \rho(dx) = 2L\beta_n.
\end{aligned}$$

**(a)** We rewrite Definition 32, where $\Delta^\pi$ is the regret of choosing the wrong action defined by Eq. 3.5.
**(b)** Since $\Pi_n$ contains piecewise constants policies induced by the partition $P_n = \{\mathcal{X}_i\}$, we split the integral as the sum over the regions.
**(c)** Since the policies in $\Pi_n$ can take any action in each possible region, the policy $\pi'$ minimizing the loss is the one which takes the best action in each region.
**(d)** Since $\mathcal{M}$ is Lipschitz, both $\max_{a \in \mathcal{A}} Q^\pi(\cdot, a)$ and $\min_{a' \in \mathcal{A}} Q^\pi(\cdot, a')$ are Lipschitz, and thus, $\Delta^\pi(\cdot)$ is $2L$-Lipschitz. Furthermore, $\Delta^\pi$ is zero in all the states in which the policy $\mathcal{G}\pi$ changes (see Figure 3.3). Thus, for any state $x$ the value $\Delta^\pi(x)$ can be bounded using the Lipschitz property by taking $y$ as the closest state to $x$ in which $\Delta^\pi(y) = 0$.

**(e)** If $\mathcal{G}\pi$ is constant in a region $\mathcal{X}_i$, the integral can be made zero by setting $a$ to the greedy action (thus making $\mathbb{I}\{(\mathcal{G}\pi)(x) \neq a\} = 0$ for any $x \in \mathcal{X}_i$). Otherwise if $\mathcal{G}\pi$ changes in a state $y \in \mathcal{X}_i$, then $\Delta^\pi(y) = 0$ and we can replace $||x - y||$ by the diameter of the region which is bounded by $\beta_n$ according to the definition of the universal family of spaces (Definition 34).
**(f)** We simply take $\mathbb{I}\{(\mathcal{G}\pi)(x) \neq a\} = 1$ in each region.
The claim follows using the definition of the universal family of policy spaces.
■

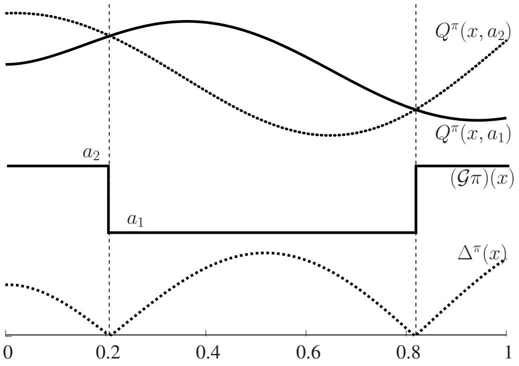


Figure 3.3: This figure is used as an illustrative example in the proof of Theorem 36. It shows the action-value function of a Lipschitz MDP for a policy $\pi$, $Q^\pi(\cdot, a_1)$ and $Q^\pi(\cdot, a_2)$ (top), the corresponding greedy policy $\mathcal{G}\pi$ (middle), and the regret of selecting the wrong action, $\Delta^\pi$, (bottom).

Theorem 36 together with the counter-example in Section 3.5.1 show that the assumption on the policy space is not enough to guarantee a small approximation error and additional assumptions on the smoothness of the MDP (e.g., Lipschitz condition) must be satisfied.

### 3.5.3 Consistency of DPI

A highly desirable property of any learning algorithm is *consistency*, i.e., as the number of samples grows to infinity, the error of the algorithm converges to zero. It can be seen that as the number of samples $N$ and the rollout horizon $H$ grow in Theorem 31, $\epsilon_1$ and $\epsilon_2$ become arbitrarily small, and thus, the expected error at each iteration, $\mathcal{L}_{\pi_k}(\rho; \pi_{k+1})$, is bounded by the inherent greedy error $d(\Pi, \mathcal{G}\Pi)$. We can conclude from the results of this section that DPI is not consistent in general, but it is consistent for the class of Lipschitz MDPs, when a universal family of policy spaces is used. However, it is important to note that as we increase the index $n$ also the capacity of the policy space $\Pi$ (its VC-dimension $h$) grows as well, and thus, when the number of samples $N$ goes to infinity, in order to still have a vanishing the estimation error ($\epsilon_1$ in Theorem 31), we should guarantee that $N$ grows faster than $VC(\Pi)$. We deduce the following result.

**Corollary 37** *Let $\mathcal{M}$ be a Lipschitz MDP with $|\mathcal{A}| = 2$, $\{\Pi_n\}$ be a universal family of policy spaces (Definition 34), $h(n) = VC(\Pi_n)$, and $\lim_{n,N\to\infty} \frac{h(n)}{N} = 0$. Then DPI is consistent:*

$$\lim_{\substack{n,\,N,\,H,\,K\,\to\,\infty \\ \delta\,\to\,0}} V^{\pi_K} = V^* \,, \qquad\qquad w.p.\ 1.$$

## 3.6 Extension to Multiple Actions

The analysis of Sections 3.4 and 3.5 are for the case that the action space contains only two actions. In Section 3.6.1 we extend the previous theoretical analysis to the general case of an action space with $|\mathcal{A}| > 2$. While the theoretical analysis is completely independent from the specific algorithm used to solve the empirical error minimization problem (see DPI algorithm of Figure 4.3), in Section 3.6.2 we discuss which algorithms could be employed to solve this problem in the case of multiple actions.

### 3.6.1 Theoretical Analysis

From the theoretical point of view, the extension of the previous results to multiple actions is straightforward. The definitions of loss and error functions do not change and we just need to use an alternative complexity measure for multi-class classification. We rely on the following definitions from Ben-David et al. [1995].

**Definition 38** *Let $\Pi \subseteq \mathcal{B}^{\pi}(\mathcal{X})$ be a set of deterministic policies and $\Psi = \big\{\psi : \mathcal{A} \to \{0, 1, *\}\big\}$ be a set of mappings from the action space to the set $\{0, 1, *\}$. A finite set of $N$ states $\mathcal{X}_N = \{x_i\}_{i=1}^N \subseteq \mathcal{X}$ is $\Psi$-shattered by $\Pi$ if there exists a vector of mappings $\psi^N = \big(\psi^{(1)}, \dots, \psi^{(N)}\big)^\top \in \Psi^N$ such that for any vector $v \in \{0, 1\}^N$, there exist a policy $\pi \in \Pi$ such that $\psi^{(i)} \circ \pi(x_i) = v_i, \quad 1 \leq i \leq N$. The $\Psi$-dimension of $\Pi$ is the maximal cardinality of a subset of $\mathcal{X}$, $\Psi$-shattered by $\Pi$.*

**Definition 39** *Let $\Pi \subseteq \mathcal{B}^{\pi}(\mathcal{X})$ be a set of deterministic policies and $\Psi = \big\{\psi_{k,l} : \mathcal{A} \to \{0, 1, *\},\ 1 \leq k \neq l \leq L\big\}$ be a set of possible mappings such that*

$$\psi_{k,l}(a) = \begin{cases} 1 & \textit{if} \ \ a = k, \\ 0 & \textit{if} \ \ a = l, \\ * & \textit{otherwise}, \end{cases}$$

*then the Natarajan dimension of $\Pi$, N-dim$(\Pi)$, is the $\Psi$-dimension of $\Pi$.*

By using a policy space with finite Natarajan dimension, we derive the following corollary to Theorem 31.

**Corollary 40** *Let $\Pi \subseteq \mathcal{B}^{\pi}(\mathcal{X})$ be a policy space with finite Natarajan dimension $h =$ N-dim$(\Pi) < \infty$. Let $\rho$ be a distribution over the state space $\mathcal{X}$, $n$ be the number of states in $\mathcal{D}_k$ drawn i.i.d. from $\rho$, and $M$ be the number* *of rollouts per state-action pair used by DPI in the estimation of the action-value functions. Let* $\pi_{k+1} = \arg\min_{\pi\in\Pi} \widehat{\mathcal{L}}_{\pi_k}(\widehat{\rho};\pi)$ *be the policy computed at the k'th iteration of DPI. Then, for any* $\delta > 0$*, we have*

$$\mathcal{L}_{\pi_k}(\rho;\pi_{k+1}) \leq \inf_{\pi\in\Pi} \mathcal{L}_{\pi_k}(\rho;\pi) + 2(\epsilon_1 + \epsilon_2 + \gamma^H Q_{\max}), \tag{3.10}$$

*with probability* $1-\delta$*, where*

$$\epsilon_1 = 16Q_{\max}\sqrt{\frac{2}{n}\left(h\log\frac{|\mathcal{A}|e(n+1)^2}{h} + \log\frac{32}{\delta}\right)} \qquad \textit{and} \qquad \epsilon_2 = (1-\gamma^H)Q_{\max}\sqrt{\frac{2}{MN}\log\frac{4|\mathcal{A}|}{\delta}}\ .$$

**Proof** In order to prove this corollary we just need a minor change in Lemma 62, which now becomes a concentration of measures inequality for a space of multi-class classifiers $\Pi$ with finite Natarajan dimension. By using similar steps as in the proof of Lemma 62 and by recalling the Sauer's lemma for finite Natarajan dimension spaces [Ben-David et al., 1995], we obtain

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

with $\epsilon = 16Q_{\max}\sqrt{\frac{2}{n}\left(h\log\frac{|\mathcal{A}|e(n+1)^2}{h} + \log\frac{8}{\delta}\right)}$. The rest of the proof is exactly the same as in Theorem 31. ■

Similarly, the consistency analysis in case of Lipschitz MDPs remains mostly unaffected by the introduction of multiple actions.

**Corollary 41** *Let* $\{\Pi_n\}$ *be a universal family of policy spaces (Definition 34), and* $\mathcal{M}$ *be a Lipschitz MDP (Definition 35). Then* $\lim_{n\to\infty} d(\Pi_n, \mathcal{G}\Pi_n) = 0$.

**Proof** The critical part in the proof is the definition of the gap function, which now compares the performance of the greedy action to the performance of the action chosen by the policy $\pi'$:

$$\Delta^{\pi,\pi'}(x) = \max_{a\in\mathcal{A}} Q^\pi(x,a) - Q^\pi\big(x,\pi'(x)\big).$$

Note that $\Delta^{\pi,\pi'}(\cdot)$ is no longer a Lipschitz function because it is a function of $x$ through the policy $\pi'$. However, $\Delta^{\pi,\pi'}(x)$ is Lipschitz in each region $\mathcal{X}_i,\ i = 1\ldots,S_n$, because in each region $\mathcal{X}_i$, by the definition of the policy space, $\pi'$ is forced to be constant. Therefore, in a region $\mathcal{X}_i$ in which $\pi'(x) = a,\ \forall x\in\mathcal{X}_i$, $\Delta^{\pi,\pi'}(x)$ may be written as

$$\Delta^{\pi,\pi'}(x) = \Delta^{\pi,a}(x) = \max_{a'\in\mathcal{A}} Q^\pi(x,a') - Q^\pi(x,a).$$

The proof here is exactly the same as in Theorem 36 up to step **(c)**, and then we have

$$\begin{aligned}
d(\Pi_n,\mathcal{G}\Pi_n) &= \sup_{\pi\in\Pi_n}\inf_{\pi'\in\Pi_n}\int_{\mathcal{X}} \ell_\pi(x;\pi')\rho(dx)\\
&= \sup_{\pi\in\Pi_n}\inf_{\pi'\in\Pi_n}\int_{\mathcal{X}} \mathbb{I}\left\{(\mathcal{G}\pi)(x)\neq\pi'(x)\right\}\Delta^{\pi,\pi'}(x)\rho(dx)\\
&= \sup_{\pi\in\Pi_n}\inf_{\pi'\in\Pi_n}\sum_{i=1}^{S_n}\int_{\mathcal{X}_i} \mathbb{I}\left\{(\mathcal{G}\pi)(x)\neq\pi'(x)\right\}\Delta^{\pi,\pi'}(x)\rho(dx)\\
&= \sup_{\pi\in\Pi_n}\sum_{i=1}^{S_n}\min_{a\in\mathcal{A}}\int_{\mathcal{X}_i} \mathbb{I}\left\{(\mathcal{G}\pi)(x)\neq a\right\}\Delta^{\pi,a}(x)\rho(dx)\\
&\leq \sup_{\pi\in\Pi_n}\sum_{i=1}^{S_n}\min_{a\in\mathcal{A}}\int_{\mathcal{X}_i} \Delta^{\pi,a}(x)\rho(dx).
\end{aligned} \tag{3.11}$$

If the greedy action does not change in a region $\mathcal{X}_i$, i.e., $\forall x\in\mathcal{X}_i,\ (\mathcal{G}\pi)(x) = a'$, for an action $a'\in\mathcal{A}$, then the minimizing policy $\pi'$ must select action $a'$ in $\mathcal{X}_i$, and thus, the loss will be zero in $\mathcal{X}_i$. Now let assume that the greedy action changes at a state $y\in\mathcal{X}_i$ and the action $b_i\in\arg\max_{a\in\mathcal{A}} Q^\pi(y,a)$. In this case, we have

$$\min_{a\in\mathcal{A}}\int_{\mathcal{X}_i}\Delta^{\pi,a}(x)\rho(dx)\leq\int_{\mathcal{X}_i}\Delta^{\pi,b_i}(x)\rho(dx)\leq\int_{\mathcal{X}_i}\big(\Delta^{\pi,b_i}(y)+2L\|x-y\|\big)\rho(dx),$$

since the function $x\mapsto\Delta^{\pi,b_i}(x)$ is $2L$-Lipschitz. Now since $\Delta^{\pi,b_i}(y)=0$, we deduce from Eq. 3.11 that

$$d(\Pi_n,\mathcal{G}\Pi_n)\leq\sup_{\pi\in\Pi_n}\sum_{i=1}^{S_n}\int_{\mathcal{X}_i}2L||x-y||\rho(dx)\leq\sup_{\pi\in\Pi_n}\sum_{i=1}^{S_n}\int_{\mathcal{X}_i}2L\beta_n\rho(dx)=2L\beta_n$$

The claim follows using the definition of the universal family of policy spaces.
■

### 3.6.2 Algorithmic Approaches

From an algorithmic point of view, the most critical part of the DPI algorithm (Figure 4.3) is minimizing the empirical error, which in the case of $|\mathcal{A}|>2$ is in the following form:

$$\min_{\pi\in\Pi}\widehat{\mathcal{L}}_{\pi_k}(\widehat{\rho};\pi)=\min_{\pi\in\Pi}\frac{1}{N}\sum_{t=1}^{n}\Big[\max_{a\in\mathcal{A}}\widehat{Q}^{\pi_k}(x_i,a)-\widehat{Q}^{\pi_k}\big(x_i,\pi(x_i)\big)\Big]$$

$$=\min_{\pi\in\Pi}\sum_{t=1}^{n}\mathbb{I}\left\{\arg\max_{a\in\mathcal{A}}\widehat{Q}^{\pi_k}(x_i,a)\neq\pi(x_i)\right\}\Big[\max_{a\in\mathcal{A}}\widehat{Q}^{\pi_k}(x_i,a)-\widehat{Q}^{\pi_k}\big(x_i,\pi(x_i)\big)\Big].$$

Unlike the two-action case, this is a multi-class cost-sensitive (MCCS) classification problem in which any classification mistake is weighted by a cost function which depends on the action taken by policy $\pi$. It is important to note that here the main difference with regression is that the goal is not to have a good approximation of the action-value function over the entire state and action space. The main objective is to have a good enough estimate of the action-value function to find the greedy action in each state. A thorough discussion on the possible approaches to MCCS classification is out of the scope of this paper, thus, we mention only a few recent methods that could be suitable for our problem. The reduction methods proposed by Beygelzimer et al. [2005, 2009] reduce the MCCS classification problem to a series of weighted binary classification problems (which can be in turn reduced to binary classification as in Zadrozny et al. 2003), whose solutions can be combined to obtain a multi-class classifier. The resulting multi-class classifier is guaranteed to have a performance which is upper-bounded by the performance of each binary classifier used in solving the weighted binary problems. Another common approach to MCCS classification is to use boosting-based methods (e.g., Lozano and Abe 2008, Busa-Fekete and Kégl 2010). Finally, a recent regression-based approach has been proposed by Tu and Lin [2010], which reduces the MCCS classification to a one-sided regression problem that can be effectively solved by a variant of SVM.

## 3.7 Conclusions

In this paper, we presented a variant of the classification-based approach to approximate policy iteration (API) called direct policy iteration (DPI) and provided its finite-sample performance bounds. To the best of our knowledge, this is the first complete finite-sample analysis for this class of API algorithms. The main difference of DPI with the existing classification-based API algorithms [Lagoudakis and Parr, 2003b, Fern et al., 2004] is in weighting each classification error by its actual regret, i.e., the difference between the action-values of the greedy action and the action selected by DPI. Our results extend the only theoretical analysis of a classification-based API algorithm [Fern et al., 2006] by **1)** having a performance bound for the full API algorithm instead of being limited to one step policy update, **2)** considering any policy space instead of finite class of policies, and **3)** deriving a bound which does not depend on the Q-advantage, i.e., the minimum Q-value gap between a greedy and a sub-greedy action over the state space, which can be arbitrarily small in a large class of MDPs. Note that the final bound in Fern et al. [2006] depends inversely on the Q-advantage. We also analyzed the consistency of DPI and showed that although it is not consistent in general, it is consistent for the class of Lipschitz MDPs. This is similar to the consistency results for fitted value iteration in Munos and Szepesvári [2008].

One of the main motivations of this work is to have a better understanding of how the classification-based API methods can be compared with their widely-used regression-based counterparts. It is interesting to note that the bound of Eq. 3.7 shares the same structure as the error bounds for the API algorithm in Antos et al. [2008] and the fitted value iteration in Munos and Szepesvári [2008]. The error at each iteration can be decomposed into an approximation error, which depends on the MDP and the richness of the hypothesis space – the inherent greedy error in Eq. 3.7 and the inherent Bellman error in Antos et al. [2008] and Munos and Szepesvári [2008], and an estimation error which mainly depends on the number of samples and rollouts. The difference between the approximation error of the two approaches depends on how well the hypothesis space fits the MDP at hand. This confirms the intuition that whenever the policies generated by policy iteration are easier to represent and learn than their value functions, a classification-based approach can be preferable to regression-based methods.

The performance of DPI is directly related to **1)** the quality of the classifier (the richness of the selected policy space), **2)** the accuracy of the generated training set, which in turn depends on the accuracy of the action-value function estimates, and finally **3)** the sampling distribution used to generate the rollout set. Possible directions for future work are related to these three issues. In the following, we mention several directions at which we have recently made some progress:

- **Cost-sensitive Multiclass Classification Problem:** As discussed in Section 3.6.2 the main issue in the implementation of DPI is the solution of the multi-class cost-sensitive classification problem at each iteration. Although some existing algorithms might be applied to this problem, further investigation is needed to identify which one is better suited for DPI. In particular, the main challenge is to solve the classification problem without first solving a regression problem on the cost function which would eliminate the main advantage of classification-based approaches (i.e., no approximation of the action-value function over the whole state-action space). On this front, we have studied cost-sensitive multiclass classification and derived risk bounds for this problem [Ávila Pires et al., 2013], which in turn will allow us to derive bounds for DPI that take into account the classification error. Note that in this chapter, we did not discuss about the implementation of the classifier. We assumed that we are capable of finding the policy that minimizes the empirical error $\widehat{\mathcal{L}}_{\pi_k}(\widehat{\rho}, \pi)$ in the policy space defined by the classifier. A commonly used approach to multiclass classification is to replace the 0/1 loss with a convex surrogate so as to make empirical risk minimization computationally tractable. Previous work has uncovered sufficient and necessary conditions for the consistency of the resulting procedures. In our work [Ávila Pires et al., 2013], we strengthen these results by showing how the 0/1 excess loss of a predictor can be upper bounded as a function of the excess loss of the predictor measured using the convex surrogate. The bound is developed for the case of cost-sensitive multiclass classification and a convex surrogate loss that goes back to the work of Lee et al. [2004].

- **Bias-Variance Tradeoff in Rollout Estimates:** The rollout estimates of the action-value functions are unbiased (if the rollouts are long enough), but may suffer from high variance (the variance increases with the length of the rollout). This raises an important question that: *Given a fixed budget of samples, how could we generate accurate action-value function estimates in DPI?* We have addressed this question by proposing several different methods with theoretical guarantees, that use a value function approximator that together with the outcome of the rollout return an estimate of the action-value function [Gabillon et al., 2011b, Scherrer et al., 2012, 2013, Farahmand et al., 2013b]. To be more precise, this value function approximator returns an estimate of the value function at the state at which we truncate the rollout. The idea is similar to the *actor-critic* algorithms that are are among the earliest studied in RL [Barto et al., 1983, Sutton, 1984]. We will discuss one of these methods, called classification-based modified policy iteration (CBMPI), in more details in Chapter 4. We obtained the best results in the literature in the game of Tetris using this class of algorithms [Gabillon et al., 2013].

- **Rollout Allocation Strategy:** In DPI, the rollouts are performed the same number of times for each state in the rollout set and each action in $\mathcal{A}$. It is natural to think that it would be more difficult to detect the greedy action at some states than the others, and thus, uniform allocation could be wasteful. Basically, the question is: *Given the rollout set, how shall we allocate a fixed budget of samples (or rollouts) to these states and the actions in the action space in order to have an accurate training set for the classifier?* This question was studied by Dimitrakakis and Lagoudakis [2008b] and some preliminary results were reported. We started a more fundamental approach to this question by first formulating it as a spacial class of bandit problems, called *pure exploration* [Gabillon et al., 2010]. This class of bandit problems is directly related to the important problem of *adaptive resource allocation* that has application in a number of different fields from marketing and advertisement to clinical studies and communication networks. We then developed several algorithms with theoretical guarantees for this class of bandit problems [Gabillon et al., 2011a, 2012]. Below is a brief description of our contribution on this topic.

The accuracy of the training set depends on how successful we are in detecting the greedy action at the states in the rollout set. Note that every time we generate a rollout at a state-action pair, we observe a random sample from a distribution, whose mean is the action-value function at that state-action pair. So, at each state of the rollout set, it is natural to think that we have a number of unknown distributions (equal to the number of possible actions at that state), and the goal is to sample them in a way to detect the one with the highest mean as fast as possible. This problem has been studied in the multi-armed bandit framework under the name *best arm identification* [Maron and Moore, 1993, Bubeck et al., 2009], and several efficient algorithms have been designed for it [Audibert et al., 2010]. In this view, each state in the rollout set is a bandit; each available action in that state in an arm; when we pull an arm, we run a rollout and receive a sample from a distribution whose mean is the action-value function of that state-action pair; and the goal is to allocate the available budget (defined in terms of the number of rollouts or pulls) in a way to detect the arm with the largest mean with high probability. However, what is important for us is to detect the greedy action at all the states in the rollout set as accurate as possible, and not just at one. Therefore, we need to extend the existing bandit algorithms for best arm identification to multiple bandits. We showed that this extension is not straight forward, meaning that it is not enough to divide the total budget equally over the bandits, and then run a best arm identification algorithm at each bandit [Gabillon et al., 2010, 2011a]. We then develop the first algorithms for multi-bandit best arm identification with theoretical guarantees and show their performance in a number of synthetic problems as well as in a problem with clinical data [Gabillon et al., 2011a, 2012]. Despite our results, there are still open problems in this front that are mainly related to the fact that we consider cost-sensitive classification in DPI, which is related to the notion of *simple regret* in pure exploration bandit setting, but the algorithms that we have developed (similar to all the pure exploration bandit algorithms) target the probability of error, and not the simple regret.

# Chapter 4

# Analysis of Approximate Modified Policy Iteration [MGH3, MGH6, MGH8]

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 implementations of approximate MPI (AMPI) that are extensions of the well-known approximate DP algorithms: fitted-value iteration, fitted-Q iteration, and classification-based policy iteration. We provide error propagation analysis that unify those for approximate policy and value iteration. We develop the finite-sample analysis of these algorithms, which highlights the influence of their free parameters. In the classification-based version of the algorithm (CBMPI), the analysis shows that MPI's main parameter controls the balance between the estimation error of the classifier and the overall value function approximation. We illustrate and evaluate the behavior of these new algorithms in the Mountain Car and Tetris problems. Remarkably, in Tetris, CBMPI outperforms by a large margin existing DP approaches and compete with the current state-of-the-art methods while using fewer samples.[1]

[1] This chapter is based on these published and submitted papers [Scherrer et al., 2012, 2013, Gabillon et al., 2013]. For more details on the results in the game of Tetris, we refer

## 4.1 Introduction

Modified Policy Iteration (MPI) [Puterman and 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

$$\pi_{k+1} = \mathcal{G}V_k \qquad \text{(greedy step)} \tag{4.1}$$

$$V_{k+1} = (\mathcal{T}^{\pi_{k+1}})^m V_k \qquad \text{(evaluation step)} \tag{4.2}$$

where $\mathcal{G}V_k$ is a *greedy* policy w.r.t. (with respect to) $V_k$, $\mathcal{T}^{\pi_k}$ is the Bellman operator associated to the policy $\pi_k$, and $m \geq 1$ is a parameter. MPI generalizes the well-known dynamic programming algorithms Value Iteration (VI) and Policy Iteration (PI) for the 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 (in terms of the number of iterations) of the PI algorithm [Puterman and Shin, 1978]. In problems with large state and/or action spaces, approximate versions of VI (AVI) and PI (API) have been the focus of a rich literature (see e.g., Bertsekas and 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 [Singh and Yee, 1994, Gordon, 1995, Bertsekas and Tsitsiklis, 1996, Munos, 2007, Ernst et al., 2005, Antos et al., 2007, Munos and 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 [Bertsekas and Tsitsiklis, 1996, Munos, 2003, Lagoudakis and Parr, 2003a]. Another related algorithm is $\lambda$-policy iteration [Bertsekas and Ioffe, 1996], which is a rather complicated variation of MPI. It involves computing a fixed-point at each iteration, and thus, suffers from some of the drawbacks of the PI algorithms. This algorithm

a reader to Gabillon et al. [2013].

has been analyzed in its approximate form by Thiery and Scherrer [2010a] (see also Scherrer 2013b). The aim of this paper is to show that, similarly to its exact form, approximate MPI (AMPI) may represent an interesting alternative to AVI and API algorithms.

In this paper, we propose three implementations of AMPI (Section 4.3) that generalize the AVI implementations of Ernst et al. [2005], Antos et al. [2007], Munos and Szepesvári [2008] and the classification-based API algorithms of Lagoudakis and Parr [2003b], Fern et al. [2006], Lazaric et al. [2010a], Gabillon et al. [2011b]. We then provide an error propagation analysis of AMPI (Section 4.4), 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 is applied to the AMPI implementations presented in Section 4.3. We then detail the analysis of the three algorithms of Section 4.3 by providing their finite sample analysis in Section 4.5. Interestingly, for the classification-based implementation of MPI (CBMPI), our analysis indicates that the parameter $m$ allows us to balance the estimation error of the classifier with the overall quality of the value approximation. Finally, we evaluate the proposed algorithms and compare them with several existing methods in the Mountain Car and Tetris problems in Section 4.6. The latter is the most challenging as DP methods uniquely based on approximating the value function have performed poorly until now. We show that the classification-based approach (CBMPI) performs well in this game.

## 4.2 Background

We consider a discounted MDP $\langle \mathcal{X}, \mathcal{A}, P, r, \gamma \rangle$, where $\mathcal{X}$ is a state space, $\mathcal{A}$ is a finite action space, $P(dx'|x,a)$, for all $(x,a)$, is a probability kernel on $\mathcal{X}$, the reward function $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is bounded by $R_{\max}$, and $\gamma \in (0,1)$ is a discount factor. A deterministic policy is defined as a mapping $\pi : \mathcal{X} \to \mathcal{A}$. For a policy $\pi$, we may write $r^\pi(x) = r\big(x, \pi(x)\big)$ and $P^\pi(dx'|x) = P\big(dx'|x, \pi(x)\big)$. The value of policy $\pi$ in a state $x$ is defined as the expected discounted sum of rewards received by starting from state $x$ and following the policy $\pi$, i.e.,

$$V^\pi(x) = \mathbb{E}\big[\sum_{t=0}^{\infty} \gamma^t r^\pi(x_t) \mid x_0 = x, x_{t+1} \sim P^\pi(\cdot|x_t)\big].$$

Similarly, the action-value function of a policy $\pi$ at a state-action pair $(x,a)$, $Q^\pi(x,a)$, is the expected discounted sum of rewards received by starting from state $x$, taking action $a$, and then following the policy $\pi$:

$$Q^\pi(x,a) = \mathbb{E}\big[\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t) \mid x_0 = x, a_0 = a, x_{t+1} \sim P(\cdot|x_t, a_t), a_{t+1} = \pi(x_{t+1})\big].$$

Since the rewards are bounded by $R_{\max}$, the values and action-values are bounded by $V_{\max} = Q_{\max} = R_{\max}/(1-\gamma)$. The Bellman operator $\mathcal{T}^\pi$ of policy $\pi$ takes a function $f$ on $\mathcal{X}$ as input and returns the function $\mathcal{T}^\pi f$ defined as

$$\forall x \in \mathcal{X}, \quad [\mathcal{T}^\pi f](x) = \mathbb{E}\big[r^\pi(x) + \gamma f(x') \mid x' \sim P^\pi(\cdot|x)\big],$$

or in compact form, $\mathcal{T}^\pi f = r^\pi + \gamma P^\pi f$. It is known that $V^\pi$ is the unique fixed-point of $\mathcal{T}^\pi$. Given a function $f$ on $\mathcal{X}$, we say that a policy $\pi$ is greedy w.r.t. $f$, and write $\pi = \mathcal{G}f$, if

$$\forall x \in \mathcal{X}, \quad [\mathcal{T}^\pi f](x) = \max_a [\mathcal{T}^a f](x),$$

or equivalently $\mathcal{T}^\pi f = \max_{\pi'}[\mathcal{T}^{\pi'} f]$. We denote by $V^*$ the optimal value function. It is also known that $V^*$ is the unique fixed-point of the Bellman optimality operator $\mathcal{T} : V \to \max_\pi \mathcal{T}^\pi V = \mathcal{T}_{\mathcal{G}(V)} V$, and that a policy $\pi^*$ that is greedy w.r.t. $V^*$ is optimal and its value satisfies $V^{\pi^*} = V^*$.

## 4.3 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 $\Pi$ to represent greedy policies. In what follows, we describe the iteration $k$ of these iterative algorithms.

### 4.3.1 AMPI-V

The first and simplest AMPI algorithm presented in the paper, called AMPI-V, is described in Figure 4.1. In AMPI-V, we assume that the values $V_k$ are represented in a function space $\mathcal{F} \subseteq \mathbb{R}^{\mathcal{X}}$. In any state $x$, the action $\pi_{k+1}(x)$ that is greedy w.r.t. $V_k$ can be estimated as follows:

$$\pi_{k+1}(x) \in \arg\max_{a \in \mathcal{A}} \frac{1}{M} \sum_{j=1}^{M} \Big( r_a^{(j)} + \gamma V_k(x_a^{(j)}) \Big), \tag{4.3}$$

where $\forall a \in \mathcal{A}$ and $1 \leq j \leq M$, $r_a^{(j)}$ and $x_a^{(j)}$ are samples of rewards and next states when action $a$ is taken in state $x$. Thus, approximating the greedy action in a state $s$ requires $M|\mathcal{A}|$ samples. The algorithm works as follows. We samples $N$ states from a distribution $\mu$ on $\mathcal{X}$, and build a rollout set

$\mathcal{D}_k = \{x^{(i)}\}_{i=1}^N$, $x^{(i)} \sim \mu$. From each state $x^{(i)} \in \mathcal{D}_k$, we generate a rollout of size $m$, i.e., $\big(x^{(i)}, a_0^{(i)}, r_0^{(i)}, x_1^{(i)}, \ldots, a_{m-1}^{(i)}, r_{m-1}^{(i)}, x_m^{(i)}\big)$, where $a_t^{(i)}$ is the action suggested by $\pi_{k+1}$ in state $x_t^{(i)}$, computed using Eq. 4.3, and $r_t^{(i)}$ and $x_{t+1}^{(i)}$ are the reward and next state induced by this choice of action. For each $x^{(i)}$, we then compute a rollout estimate

$$\widehat{V}_{k+1}(x^{(i)}) = \sum_{t=0}^{m-1} \gamma^t r_t^{(i)} + \gamma^m V_k(x_m^{(i)}), \tag{4.4}$$

which is an unbiased estimate of $\big[(\mathcal{T}^{\pi_{k+1}})^m V_k\big](x^{(i)})$. Finally, $V_{k+1}$ is computed as the best fit in $\mathcal{F}$ to these estimates, i.e., it is a function $V \in \mathcal{F}$ that minimizes the empirical error

$$\widehat{\mathcal{L}}_k^{\mathcal{F}}(\widehat{\mu}; V) = \frac{1}{N}\sum_{i=1}^{N} \big(\widehat{V}_{k+1}(x^{(i)}) - V(x^{(i)})\big)^2, \tag{4.5}$$

with the goal of minimizing the true error

$$\mathcal{L}_k^{\mathcal{F}}(\mu; V) = \Big|\Big|\big[(\mathcal{T}^{\pi_{k+1}})^m V_k\big] - V\Big|\Big|_{2,\mu}^2 = \int \Big(\big[(\mathcal{T}^{\pi_{k+1}})^m V_k\big](x) - V(x)\Big)^2 \mu(dx).$$

Each iteration of AMPI-V requires $N$ rollouts of size $m$, and in each rollout, each of the $|\mathcal{A}|$ actions needs $M$ samples to compute Eq. 4.3. This gives a total of $Nm(M|\mathcal{A}|+1)$ transition samples. Note that the fitted value iteration algorithm [Munos and Szepesvári, 2008] is a special case of AMPI-V when $m = 1$.

### 4.3.2 AMPI-Q

In AMPI-Q, we replace the value function $V : \mathcal{X} \to \mathbb{R}$ with the action-value function $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$. Figure 4.2 contains the pseudocode of this algorithm. The Bellman operator for a policy $\pi$ at a state-action pair $(x, a)$

```
Input: Value function space $\mathcal{F}$, state distribution $\mu$
Initialize: Let $V_0 \in \mathcal{F}$ be an arbitrary value function
for $k = 0, 1, \dots$ do
  • Perform rollouts:
  Construct the rollout set $\mathcal{D}_k = \{x^{(i)}\}_{i=1}^N,\ x^{(i)} \overset{\text{iid}}{\sim} \mu$
  for all states $x^{(i)} \in \mathcal{D}_k$ do
    Perform a rollout (using Eq. 4.3 for each action)
    $\widehat{V}_{k+1}(x^{(i)}) = \sum_{t=0}^{m-1} \gamma^t r_t^{(i)} + \gamma^m V_k(x_m^{(i)})$
  end for
  • Approximate value function:
  $V_{k+1} \in \arg\min_{V \in \mathcal{F}} \widehat{\mathcal{L}}_k^{\mathcal{F}}(\widehat{\mu}; V)$        (regression)      (see Eq. 4.5)
end for
```

Figure 4.1: The pseudo-code of the AMPI-V algorithm.

can then be written as

$$[\mathcal{T}^\pi Q](x,a) = \mathbb{E}\big[r(x,a) + \gamma Q(x', \pi(x')) \mid x' \sim P(\cdot|x,a)\big],$$

and the greedy operator is defined as

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

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

$$\pi_{k+1}(x) \in \arg\max_{a \in \mathcal{A}} \; Q_k(x,a). \tag{4.6}$$

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 $\mu$ on $\mathcal{X} \times \mathcal{A}$ and build a rollout set $\mathcal{D}_k = \{(x^{(i)}, a^{(i)})\}_{i=1}^N,\ (x^{(i)}, a^{(i)}) \sim \mu$. For each $(x^{(i)}, a^{(i)}) \in \mathcal{D}_k$, we generate a rollout of size $m$, i.e., $\big(x^{(i)}, a^{(i)}, r_0^{(i)}, x_1^{(i)}, a_1^{(i)}, \cdots, x_m^{(i)}, a_m^{(i)}\big)$, where the first action is $a^{(i)}$, $a_t^{(i)}$ for $t \geq 1$ is the action suggested by $\pi_{k+1}$ in state $x_t^{(i)}$ computed using Eq. 4.6, and $r_t^{(i)}$ and $x_{t+1}^{(i)}$

are the reward and next state induced by this choice of action. For each $(x^{(i)}, a^{(i)}) \in \mathcal{D}_k$, we then compute the rollout estimate

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

which is an unbiased estimate of $\big[(\mathcal{T}^{\pi_{k+1}})^m Q_k\big](x^{(i)}, a^{(i)})$. Finally, $Q_{k+1}$ is the best fit to these estimates in $\mathcal{F}$, i.e., it is a function $Q \in \mathcal{F}$ that minimizes the empirical error

$$\widehat{\mathcal{L}}_k^{\mathcal{F}}(\widehat{\mu}; Q) = \frac{1}{N} \sum_{i=1}^{N} \big(\widehat{Q}_{k+1}(x^{(i)}, a^{(i)}) - Q(x^{(i)}, a^{(i)})\big)^2, \tag{4.7}$$

with the goal of minimizing the true error

$$\mathcal{L}_k^{\mathcal{F}}(\mu; Q) = \Big|\Big| \big[(\mathcal{T}^{\pi_{k+1}})^m Q_k\big] - Q \Big|\Big|_{2,\mu}^2 = \int \Big(\big[(\mathcal{T}^{\pi_{k+1}})^m Q_k\big](x,a) - Q(x,a)\Big)^2 \mu(dxda).$$

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 (a larger space than that used by AMPI-V). Note that the fitted-Q iteration algorithm [Ernst et al., 2005, Antos et al., 2007] is a special case of AMPI-Q when $m = 1$.

### 4.3.3 Classification-Based MPI

The third AMPI algorithm presented in this paper, called classification-based MPI (CBMPI), uses an explicit representation for the policies $\pi_k$, in addition to the one used for the value functions $V_k$. The idea is similar to the classification-based PI algorithms [Lagoudakis and Parr, 2003b, Fern et al., 2006, Lazaric et al., 2010a, Gabillon et al., 2011b] in which we search for the greedy policy in a policy space $\Pi$ (defined by a classifier) instead of computing it from the estimated value or action-value function (like in

```
Input: Value function space $\mathcal{F}$, state distribution $\mu$
Initialize: Let $Q_0 \in \mathcal{F}$ be an arbitrary value function
for $k = 0, 1, \dots$ do
  • Perform rollouts:
  Construct the rollout set $\mathcal{D}_k = \{(x^{(i)}, a^{(i)}\}_{i=1}^N,\ (x^{(i)}, a^{(i)}) \overset{\text{iid}}{\sim} \mu$
  for all states $(x^{(i)}, a^{(i)}) \in \mathcal{D}_k$ do
    Perform a rollout (using Eq. 4.6 for each action)
    $\widehat{Q}_{k+1}(x^{(i)}, a^{(i)}) = \sum_{t=0}^{m-1} \gamma^t r_t^{(i)} + \gamma^m Q_k(x_m^{(i)}, a_m^{(i)})$,
  end for
  • Approximate action-value function:
  $Q_{k+1} \in \arg\min_{Q \in \mathcal{F}} \widehat{\mathcal{L}}_k^{\mathcal{F}}(\widehat{\mu}; Q)$        (regression)     (see Eq. 4.7)
end for
```

Figure 4.2: The pseudo-code of the AMPI-Q algorithm.

AMPI-V and AMPI-Q).

In order to describe CBMPI, we first rewrite the MPI formulation (Eqs. 4.1 and 4.2) as

$$V_k = (\mathcal{T}^{\pi_k})^m V_{k-1} \qquad \text{(evaluation step)} \qquad (4.8)$$

$$\pi_{k+1} = \mathcal{G}\big[(\mathcal{T}^{\pi_k})^m V_{k-1}\big] \qquad \text{(greedy step)} \qquad (4.9)$$

Note that in the new formulation both $V_k$ and $\pi_{k+1}$ are functions of $(\mathcal{T}^{\pi_k})^m V_{k-1}$. CBMPI is an approximate version of this new formulation. As described in Figure 4.3, CBMPI begins with arbitrary initial policy $\pi_1 \in \Pi$ and value function $V_0 \in \mathcal{F}$.[2] At each iteration $k$, a new value function $V_k$ is built as the best approximation of the $m$-step Bellman operator $(\mathcal{T}^{\pi_k})^m V_{k-1}$ in $\mathcal{F}$ (*evaluation step*). This is done by solving a regression problem whose target function is $(\mathcal{T}^{\pi_k})^m V_{k-1}$. To set up the regression problem, we build a rollout set $\mathcal{D}_k$ by sampling $N$ states i.i.d. from a distribution $\mu$.[3] For each

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

[3] Here we used the same sampling distribution $\mu$ for both regressor and classifier, but

```
Input: Value function space $\mathcal{F}$, policy space $\Pi$, state distribution $\mu$
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 $\mathcal{D}_k = \{x^{(i)}\}_{i=1}^N,\ x^{(i)} \overset{\text{iid}}{\sim} \mu$
  for all states $x^{(i)} \in \mathcal{D}_k$ do
    Perform a rollout and return $\widehat{V}_k(x^{(i)})$                    (using Eq. 4.10)
  end for
  Construct the rollout set $\mathcal{D}'_k = \{x^{(i)}\}_{i=1}^N,\ x^{(i)} \overset{\text{iid}}{\sim} \mu$
  for all states $x^{(i)} \in \mathcal{D}'_k$ and actions $a \in \mathcal{A}$ do
    for $j = 1$ to $M$ do
      Perform a rollout and return $R_k^j(x^{(i)}, a)$                (using Eq. 4.15)
    end for
    $\widehat{Q}_k(x^{(i)}, a) = \frac{1}{M} \sum_{j=1}^M R_k^j(x^{(i)}, a)$
  end for
  • Approximate value function:
  $V_k \in \arg\min_{V \in \mathcal{F}} \widehat{\mathcal{L}}_k^{\mathcal{F}}(\widehat{\mu}; V)$          (regression)      (see Eq. 4.11)
  • Approximate greedy policy:
  $\pi_{k+1} \in \arg\min_{\pi \in \Pi} \widehat{\mathcal{L}}_k^{\Pi}(\widehat{\mu}; \pi)$       (classification)      (see Eq. 4.16)
end for
```

Figure 4.3: The pseudo-code of the CBMPI algorithm.

state $x^{(i)} \in \mathcal{D}_k$, we generate a rollout $\big(x^{(i)}, a_0^{(i)}, r_0^{(i)}, x_1^{(i)}, \ldots, a_{m-1}^{(i)}, r_{m-1}^{(i)}, x_m^{(i)}\big)$ of size $m$, where $a_t^{(i)} = \pi_k(x_t^{(i)})$, and $r_t^{(i)}$ and $x_{t+1}^{(i)}$ are the reward and next state induced by this choice of action. From this rollout, we compute an unbiased estimate $\widehat{V}_k(x^{(i)})$ of $\big[(\mathcal{T}^{\pi_k})^m V_{k-1}\big](x^{(i)})$ as in Eq. 4.4:

$$\widehat{V}_k(x^{(i)}) = \sum_{t=0}^{m-1} \gamma^t r_t^{(i)} + \gamma^m V_{k-1}(x_m^{(i)}), \tag{4.10}$$

and use it to build a training set $\big\{\big(s^{(i)}, \widehat{V}_k(x^{(i)})\big)\big\}_{i=1}^N$. This training set is then used by the regressor to compute $V_k$ as an estimate of $(\mathcal{T}^{\pi_k})^m V_{k-1}$. Similar

in general different distributions may be used for these two components of the algorithm.

to the AMPI-V algorithm, the regressor here finds a function $V \in \mathcal{F}$ that minimizes the empirical error

$$\widehat{\mathcal{L}}_k^{\mathcal{F}}(\widehat{\mu}; V) = \frac{1}{N}\sum_{i=1}^{N}\big(\widehat{V}_k(x^{(i)}) - V(x^{(i)})\big)^2, \tag{4.11}$$

with the goal of minimizing the true error

$$\mathcal{L}_k^{\mathcal{F}}(\mu; V) = \Big|\Big|\big[(\mathcal{T}^{\pi_k})^m V_{k-1}\big] - V\Big|\Big|_{2,\mu}^2 = \int \Big(\big[(\mathcal{T}^{\pi_k})^m V_{k-1}\big](x) - V(x)\Big)^2 \mu(dx).$$

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

$$\big[\mathcal{T}^{\pi}(\mathcal{T}^{\pi_k})^m V_{k-1}\big](x) = \max_{a\in\mathcal{A}} \big[\mathcal{T}^{a}(\mathcal{T}^{\pi_k})^m V_{k-1}\big](x). \tag{4.12}$$

By defining $Q_k(x,a) = \big[\mathcal{T}^{a}(\mathcal{T}^{\pi_k})^m V_{k-1}\big](x)$, we may rewrite Eq. 4.12 as

$$Q_k\big(x, \pi(x)\big) = \max_{a\in\mathcal{A}} Q_k(x,a). \tag{4.13}$$

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

$$\mathcal{L}_{\pi_k, V_{k-1}}^{\Pi}(\mu; \pi) = \int \Big[\max_{a\in\mathcal{A}} Q_k(x,a) - Q_k\big(x, \pi(x)\big)\Big]\mu(dx). \tag{4.14}$$

To simplify the notation we use $\mathcal{L}_k^{\Pi}$ instead of $\mathcal{L}_{\pi_k, V_{k-1}}^{\Pi}$. 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 $\mu$. For each state $x^{(i)} \in \mathcal{D}'_k$ and each action

$a \in \mathcal{A}$, we build $M$ independent rollouts of size $m+1$, i.e.,[4]

$$\Big(x^{(i)}, a, r_0^{(i,j)}, x_1^{(i,j)}, a_1^{(i,j)}, \ldots, a_m^{(i,j)}, r_m^{(i,j)}, x_{m+1}^{(i,j)}\Big)_{j=1}^{M},$$

where for $t \geq 1$, $a_t^{(i,j)} = \pi_k(x_t^{(i,j)})$, and $r_t^{(i,j)}$ and $x_{t+1}^{(i,j)}$ are the reward and next state induced by this choice of action. From these rollouts, we compute an unbiased estimate of $Q_k(x^{(i)}, a)$ as $\widehat{Q}_k(x^{(i)}, a) = \frac{1}{M}\sum_{j=1}^{M} R_k^j(x^{(i)}, a)$ where

$$R_k^j(x^{(i)}, a) = \sum_{t=0}^{m} \gamma^t r_t^{(i,j)} + \gamma^{m+1} V_{k-1}(x_{m+1}^{(i,j)}). \tag{4.15}$$

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

$$\widehat{\mathcal{L}}_k^{\Pi}(\widehat{\mu}; \pi) = \frac{1}{N'} \sum_{i=1}^{N'} \Big[ \max_{a \in \mathcal{A}} \widehat{Q}_k(x^{(i)}, a) - \widehat{Q}_k\big(x^{(i)}, \pi(s^{(i)})\big) \Big], \tag{4.16}$$

with the goal of minimizing the true error $\mathcal{L}_k^{\Pi}(\mu; \pi)$ defined by Eq. 4.14.

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 $\infty$, we recover the DPI algorithm proposed and analyzed by Lazaric et al. [2010a].

### 4.3.4 Possible Approaches to Reuse the Samples

In all the proposed AMPI algorithms, we generate fresh samples for the rollouts, and even for the starting states, at each iteration. This results in high sample complexity for these algorithms. In this section, we propose two possible approaches to circumvent this problem and to keep the number of

[4] We may implement CBMPI more sample efficient by reusing the rollouts generated for the greedy step in the evaluation step, but this makes the analysis of the algorithm more complicated.

samples independent of the number of iterations.

One approach would be to use a fixed set of starting samples $(x^{(i)})$ or $(x^{(i)}, a^{(i)})$ for all iterations, and think of a tree of depth $m$ that contains all the possible outcomes of $m$-steps choices of actions (this tree contains $|A|^m$ leaves). This is reminiscent of the work by Kearns et al. [2000]. Using this tree, all the trajectories with the same actions share the same samples. In practice, it is not necessarily to build the entire depth $m$ tree, it is only needed to add a branch when the desired action does not belong to the tree. Using this approach, the sample complexity of the algorithm no longer depends on the number of iterations. For example, we may only need $NM|A|^m$ transitions for the CBMPI algorithm.

We may also consider the case where we do not have access to a generative model of the system, and all we have is a set of trajectories of size $m$ generated by a behavior policy $\pi_b$ that is assumed to choose all actions $a$ in each state $x$ with a positive probability (i.e., $\pi_b(a|x) > 0,\ \forall x, \forall a$) [Precup et al., 2000, 2001]. In this case, one may still compute an unbiased estimate of the application of $(\mathcal{T}^\pi)^m$ operator to value and action-value functions. For instance, given a $m$-step sample trajectory $(x, a_0, r_0, x_1, \ldots, x_m, a_m)$ generated by $\pi_b$, an unbiased estimate of $[(\mathcal{T}^\pi)^m V](x)$ may be computed as (assuming that the distribution $\mu$ has the following factored form $p(x, a_0|\mu) = p(x)\pi_b(a_0|x)$ at state $x$)

$$y = \sum_{t=0}^{m-1} \alpha_t \gamma^t r_t + \alpha_m \gamma^m V(x_m), \qquad \text{where} \qquad \alpha_t = \prod_{j=1}^{t} \frac{1_{a_j = \pi(x_j)}}{\pi_b(a_j|x_j)}$$

is an importance sampling correction factor that can be computed along the trajectory. However, this process may significantly increase the variance of such an estimate, and thus, requires many more samples.

## 4.4 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 $\gamma$-contraction in max-norm [Bertsekas and 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., for $m > 1$).

**Proposition 42** *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** We consider the MDP used to prove a similar result for $\lambda$-policy iteration [Scherrer, 2013b]. It is a deterministic model with two states $\{x_1, x_2\}$, two actions $\{change, stay\}$, rewards $r(x_1) = 0,\ r(x_2) = 1$, and transitions $P_{ch}(x_2|x_1) = P_{ch}(x_1|x_2) = P_{st}(x_1|x_1) = P_{st}(x_2|x_2) = 1$. Consider 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 $(\mathcal{T}^{\pi})^m V = \begin{pmatrix} \gamma^m \epsilon \\ 1 + \gamma^m \epsilon \end{pmatrix}$ and $(\mathcal{T}^{\pi'})^m V' = \begin{pmatrix} \frac{\gamma - \gamma^m}{1-\gamma} + \gamma^m \epsilon \\ \frac{1-\gamma^m}{1-\gamma} + \gamma^m \epsilon \end{pmatrix}$. Thus, $(\mathcal{T}^{\pi'})^m V' - (\mathcal{T}^{\pi})^m V = \begin{pmatrix} \frac{\gamma-\gamma^m}{1-\gamma} \\ \frac{\gamma-\gamma^m}{1-\gamma} \end{pmatrix}$, while $V' - V = \begin{pmatrix} -\epsilon \\ \epsilon \end{pmatrix}$. Since $\epsilon$ can be arbitrarily small, the norm of $(\mathcal{T}^{\pi'})^m V' - (\mathcal{T}^{\pi})^m V$ can be arbitrarily larger than the norm of $V - V'$ as long 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 and 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 42 that for $m \neq 1$ and $\infty$, MPI is neither contracting nor non-decreasing, and thus, a new proof is needed for the propagation of errors in this algorithm.

To study error propagation in AMPI, we introduce an abstract algorithmic model that accounts for potential errors. 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 $\epsilon'_k$, called the *greedy step error*. Thus, we write the new policy $\pi_k$ as

$$\pi_k = \widehat{\mathcal{G}}_{\epsilon'_k} V_{k-1}. \tag{4.17}$$

Eq. 4.17 means that for any policy $\pi'$, we have $\mathcal{T}^{\pi'} V_{k-1} \leq \mathcal{T}^{\pi_k} V_{k-1} + \epsilon'_k$. AMPI then generates the new value function $V_k$ with some error $\epsilon_k$, called the *evaluation step error*

$$V_k = (\mathcal{T}^{\pi_k})^m V_{k-1} + \epsilon_k. \tag{4.18}$$

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 4.3 separately.

**AMPI-V:** The term $\epsilon_k$ is the error when 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. The term $\epsilon'_k$ is the error due to using a finite number of samples $M$ for estimating the greedy actions.

**AMPI-Q:** In this case $\epsilon'_k = 0$ and $\epsilon_k$ is the error in fitting the state-action value function $Q_k$.

**CBMPI:** This algorithm iterates as follows:

$$V_k = (\mathcal{T}^{\pi_k})^m V_{k-1} + \epsilon_k$$
$$\pi_{k+1} = \widehat{\mathcal{G}}_{\epsilon'_{k+1}} \left[ (\mathcal{T}^{\pi_k})^m V_{k-1} \right].$$

Unfortunately, this does not exactly match the model described in Eqs. 4.17 and 4.18. By introducing the auxiliary variable $W_k \overset{\Delta}{=} (\mathcal{T}^{\pi_k})^m V_{k-1}$, we have $V_k = W_k + \epsilon_k$, and thus, we may write

$$\pi_{k+1} = \widehat{\mathcal{G}}_{\epsilon'_{k+1}} \left[ W_k \right]. \tag{4.19}$$

Using $V_{k-1} = W_{k-1} + \epsilon_{k-1}$, we have

$$W_k = (\mathcal{T}^{\pi_k})^m V_{k-1} = (\mathcal{T}^{\pi_k})^m (W_{k-1} + \epsilon_{k-1}) = (\mathcal{T}^{\pi_k})^m W_{k-1} + (\gamma P^{\pi_k})^m \epsilon_{k-1}. \tag{4.20}$$

Now, Eqs. 4.19 and 4.20 exactly match Eqs. 4.17 and 4.18 by replacing $V_k$ with $W_k$ and $\epsilon_k$ with $(\gamma P^{\pi_k})^m \epsilon_{k-1}$.

The rest of this section is devoted to show how the errors $\epsilon_k$ and $\epsilon'_k$ propagate through the iterations of an AMPI algorithm. We only outline the main arguments that will lead to the performance bound of Theorem 48 and report most proofs in Appendices 4.8.1 to 4.8.4. Here we follow the line of analysis developed by Scherrer and Thiéry [2010]. The results are obtained using the following three quantities:

**1)** The distance between the optimal value function and the value before approximation at the $k^{\text{th}}$ iteration: $d_k \overset{\Delta}{=} V^* - (\mathcal{T}^{\pi_k})^m V_{k-1} = V^* - (V_k - \epsilon_k)$.

**2)** The shift between the value before approximation and the value of the policy at the $k^{\text{th}}$ iteration: $s_k \overset{\Delta}{=} (\mathcal{T}^{\pi_k})^m V_{k-1} - V^{\pi_k} = (V_k - \epsilon_k) - V^{\pi_k}$.

**3)** The (approximate) Bellman residual at the $k^{\text{th}}$ iteration: $b_k \overset{\Delta}{=} V_k - \mathcal{T}^{\pi_{k+1}} V_k$.

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

**Lemma 43** *Let $k \geq 1$, $x_k \stackrel{\Delta}{=} (I - \gamma P^{\pi_k})\epsilon_k + \epsilon'_{k+1}$ and $y_k \stackrel{\Delta}{=} -\gamma P^{\pi^*}\epsilon_k + \epsilon'_{k+1}$. We have:*

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

**Proof** See Appendix 4.8.1. ■

Since the stochastic kernels are non-negative, the bounds in Lemma 43 indicate that the loss $l_k$ will be bounded if the errors $\epsilon_k$ and $\epsilon'_k$ are controlled. In fact, if we define $\epsilon$ as a uniform upper-bound on the pointwise absolute value of the errors, $|\epsilon_k|$ and $|\epsilon'_k|$, the first inequality in Lemma 43 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 43. Before we state this result, we introduce some notations that will ease our formulation.

**Definition 44** *For a positive integer $n$, we define $\mathbb{P}_n$ as the set of discounted 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 $\Gamma^n$ 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 in Definition 44, we now derive a point-wise bound on the loss.

**Lemma 45** *After k iterations, the losses of AMPI-V and AMPI-Q satisfy*

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

*while the loss of CBMPI satisfies*

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

*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 Appendix 4.8.2. ■

**Remark 46** *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:*

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

*This indicates that the bound in Lemma 45 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 44.*[5]

The next step is to show how the point-wise bound of Lemma 45 can turn to a bound in weighted $L_p$-norm, which for any function $f : \mathcal{S} \to \mathbb{R}$ and any distribution $\mu$ on $\mathcal{S}$ is defined as $\|f\|_{p,\mu} \overset{\Delta}{=} \left[\int |f(x)|^p \mu(dx)\right]^{1/p}$. Munos [2003, 2007], Munos and 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 51. We will also show through the proofs of Theorems 48 and 59, how the result of Lemma 47 provides us with a flexible tool for turning point-wise bounds into $L_p$-norm bounds. Theorem 59 in Appendix 4.8.4 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).

**Lemma 47** *Let $\mathcal{I}$ and $(\mathcal{J}_i)_{i\in\mathcal{I}}$ be sets of positive integers, $\{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$ 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 $\rho$ and $\mu$, we have*

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

[5] Note however that the dependence on $m$ will reappear if we make explicit what is hidden in $\Gamma^j$ terms.

*with the following concentrability coefficients*

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

*with the Radon-Nikodym derivative based quantity*

$$c_q(j) \triangleq \max_{\pi_1,\cdots,\pi_j}\left\|\frac{d(\rho P^{\pi_1}P^{\pi_2}\cdots P^{\pi_j})}{d\mu}\right\|_{q,\mu}. \tag{4.21}$$

**Proof** See Appendix 4.8.3. ■

We now derive a $L_p$-norm bound for the loss of the AMPI algorithm by applying Lemma 47 to the point-wise bound of Lemma 45.

**Theorem 48** *Let $\rho$ and $\mu$ be distributions over states. Let $p$, $q$, and $q'$ be such that $\frac{1}{q}+\frac{1}{q'}=1$. After $k$ iterations, the loss of AMPI satisfies*

$$\|l_k\|_{p,\rho} \le \frac{2(\gamma-\gamma^k)\left(\mathcal{C}_q^{1,k,0}\right)^{\frac{1}{p}}}{(1-\gamma)^2}\sup_{1\le j\le k-1}\|\epsilon_j\|_{pq',\mu}+\frac{(1-\gamma^k)\left(\mathcal{C}_q^{0,k,0}\right)^{\frac{1}{p}}}{(1-\gamma)^2}\sup_{1\le j\le k}\|\epsilon'_j\|_{pq',\mu}+g(k), \tag{4.22}$$

*while the loss of CBMPI satisfies*

$$\|l_k\|_{p,\rho} \le \frac{2\gamma^m(\gamma-\gamma^{k-1})\left(\mathcal{C}_q^{2,k,m}\right)^{\frac{1}{p}}}{(1-\gamma)^2}\sup_{1\le j\le k-2}\|\epsilon_j\|_{pq',\mu}+\frac{(1-\gamma^k)\left(\mathcal{C}_q^{1,k,0}\right)^{\frac{1}{p}}}{(1-\gamma)^2}\sup_{1\le j\le k}\|\epsilon'_j\|_{pq',\mu}+g(k), \tag{4.23}$$

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

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

*with $c_q(j)$ given by Eq. 4.21, and $g(k)$ is defined as*

$$g(k) \triangleq \frac{2\gamma^k}{1-\gamma} \left(\mathcal{C}_q^{k,k+1,0}\right)^{\frac{1}{p}} \min\left(\|d_0\|_{pq',\mu}, \|b_0\|_{pq',\mu}\right).$$

**Proof** See Appendix 4.8.4. ■

**Remark 49** *When $p$ tends to infinity, the first bound of Theorem 48 reduces to*

$$\|l_k\|_\infty \leq \frac{2(\gamma-\gamma^k)}{(1-\gamma)^2} \sup_{1\leq j\leq k-1} \|\epsilon_j\|_\infty + \frac{1-\gamma^k}{(1-\gamma)^2} \sup_{1\leq j\leq k} \|\epsilon'_j\|_\infty + \frac{2\gamma^k}{1-\gamma} \min(\|d_0\|_\infty, \|b_0\|_\infty). \tag{4.24}$$

*When $k$ goes to infinity, Eq. 4.24 gives us a generalization of the API ($m = \infty$) bound of Bertsekas and Tsitsiklis [1996, Proposition 6.2], i.e.,*

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

*Moreover, since our point-wise analysis generalizes those of API and AVI (as noted in Remark 46), the $L_p$-bound of Eq. 4.22 unifies and generalizes those for API [Munos, 2003] and AVI [Munos, 2007].*

**Remark 50** *The arguments we developed globally follow those originally developed for $\lambda$-policy iteration [Scherrer, 2013b]. With respect to that work, our proof is significantly simpler thanks to the use of the notation $\Gamma^n$ (Definition 44) and the fact that the AMPI scheme is itself much simpler that $\lambda$-policy iteration. Moreover, the results are deeper since we consider a possible error in the greedy step and more general concentration coefficients. Canbolat and Rothblum [2012] recently (and independently) developed an analysis of an approximate form of MPI. While Canbolat and Rothblum [2012] only consider the error in the greedy step, our work is more general since it takes into account both this error and the error in the value update. Note that* *it is required to consider both sources of error for the analysis of CBMPI. Moreover, Canbolat and Rothblum [2012] provide bounds when the errors are controlled in max-norm, while we consider the more general $L_p$-norm. At a more technical level, Theorem 2 in Canbolat and Rothblum [2012] bounds the norm of the distance $V^* - V_k$ while we bound the loss $V^* - V^{\pi_k}$. Finally, if we derive a bound on the loss (using e.g., Theorem 1 in Canbolat and Rothblum 2012), this leads to a bound on the loss that is looser than ours. In particular, this does not allow to recover the standard bounds for AVI/API, as we managed to obtain here (c.f., Remark 49).*

**Remark 51** *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] and Munos and Szepesvári [2008], and $q = q' = 2$ in Farahmand et al. [2010].*

**Remark 52** *Interestingly, our loss bound for AMPI does not "directly" depend on $m$ (although as we will discuss in the next section, it actually does depend "indirectly" through $\epsilon_k$). For CBMPI, the parameter $m$ controls the influence of the value function approximator, cancelling it out in the limit when $m$ tends to infinity (see Eq. 4.23). Assuming a fixed budget of sample transitions, increasing $m$ reduces the number of rollouts used by the classifier and thus worsens its quality. In such a situation, $m$ allows making a trade-off between the estimation errors of the classifier and the overall value function approximation.*

**Remark 53** *The result that we have just stated means the following: if one can control the errors $\epsilon_k$ and $\epsilon'_k$, then the performance loss is also controlled. The main limitation of this result is that in general, even if we consider* *that there is no sampling noise ($N = \infty$ for all algorithms and $M = \infty$ for AMPI-V), the error $\epsilon_k$ of the* evaluation step *may grow arbitrarily and make the algorithm diverge. The fundamental reason is that the composition of the approximation and the Bellman operator $\mathcal{T}^\pi$ is not necessarily contracting. A simple well-known pathological example is due to Tsitsiklis and Van Roy [1997] and involves a two-state uncontrolled MDP and a linear projection on a 1-dimensional space (that contains the real value function). Increasing the parameter $m$ of the algorithm makes the operator $\mathcal{T}^\pi$ more contracting and in principle can address this issue. For instance, if we consider that we have a state space of finite size $|\mathcal{X}|$, and take the uniform distribution $\mu$, it can be easily seen that for any $V$ and $V'$, we have*

$$\begin{aligned} \|(\mathcal{T}^\pi)^m V - (\mathcal{T}^\pi)^m V'\|_{2,\mu} &= \gamma^m \|(P^\pi)^m (V - V')\|_{2,\mu} \\ &\leq \gamma^m \|(P^\pi)^m\|_{2,\mu} \|V - V'\|_{2,\mu} \\ &\leq \gamma^m \sqrt{|\mathcal{X}|} \|V - V'\|_{2,\mu}. \end{aligned}$$

*In other words, $\mathcal{T}^\pi$ is contracting w.r.t. the $\mu$-weighted norm as soon as $m > \frac{\log|\mathcal{X}|}{2\log\frac{1}{\gamma}}$. In particular it is sufficient for $m$ to be exponentially smaller than the state space size to solve this potential divergence problem.*

## 4.5 Finite-Sample Analysis of the Algorithms

In this section, we first show how the error terms $\epsilon_k$ and $\epsilon'_k$ appeared in Theorem 48 (Eqs. 4.22 and 4.23) are bounded in each of the three proposed algorithms, and then use the obtained results and derive finite-sample performance bounds for these algorithms. We first bound the *evaluation step error $\epsilon_k$*. In AMPI-V and CBMPI, the evaluation step at each iteration $k$ is a regression problem with the target $(\mathcal{T}^{\pi_k})^m V_{k-1}$ and a training set of the form $\big\{\big(x^{(i)}, \widehat{V}_k(x^{(i)})\big)\big\}_{i=1}^N$ in which the states $x^{(i)}$ are i.i.d. samples from the distribution $\mu$ and $\widehat{V}_k(x^{(i)})$'s are unbiased estimates of the target computed using Eq. 4.4. The situation is the same for AMPI-Q, except everything is in terms of action-value function $Q_k$ instead of value function $V_k$. Therefore, in the following we only show how to bound $\epsilon_k$ in AMPI-V and CBMPI, the extension to AMPI-Q is straightforward.

We may use different function spaces $\mathcal{F}$ (linear or non-linear) to approximate $(\mathcal{T}^{\pi_k})^m V_{k-1}$. Here we consider a linear architecture with parameters $\alpha \in \mathbb{R}^d$ and bounded (by $L$) basis functions $\{\varphi_j\}_{j=1}^d$, $\|\varphi_j\|_\infty \leq L$. We denote by $\phi : \mathcal{X} \to \mathbb{R}^d$, $\phi(\cdot) = \big(\varphi_1(\cdot), \ldots, \varphi_d(\cdot)\big)^\top$ the feature vector, and by $\mathcal{F}$ the linear function space spanned by the features $\varphi_j$, i.e., $\mathcal{F} = \{f_\alpha(\cdot) = \phi(\cdot)^\top \alpha : \alpha \in \mathbb{R}^d\}$. Now if we define $V_k$ as the truncation (by $V_{\max}$) of the solution of the above linear regression problem, we may bound the *evaluation step error* $\epsilon_k$ using the following lemma.

**Lemma 54 (Evaluation step error)** *Consider the linear regression setting described above, then we have*

$$\|\epsilon_k\|_{2,\mu} \leq 4 \inf_{f \in \mathcal{F}} \|(\mathcal{T}^{\pi_k})^m V_{k-1} - f\|_{2,\mu} + e_1(N,\delta) + e_2(N,\delta),$$

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

$$e_1(N,\delta) = 32 V_{\max} \sqrt{\frac{2}{N} \log\Big(\frac{27(12e^2N)^{2(d+1)}}{\delta}\Big)}\,,$$
$$e_2(N,\delta) = 24\Big(V_{\max} + \|\alpha^*\|_2 \cdot \sup_x \|\phi(x)\|_2\Big)\sqrt{\frac{2}{N}\log\frac{9}{\delta}}\,,$$

*and* $\alpha^*$ *is such that* $f_{\alpha^*}$ *is the best approximation (w.r.t.* $\mu$*) of the target function* $(\mathcal{T}^{\pi_k})^m V_{k-1}$ *in* $\mathcal{F}$*.*

**Proof** See Appendix 4.8.5. ■

After we showed how to bound the *evaluation step error* $\epsilon_k$ for the proposed algorithms, we now turn our attention to bounding the *greedy step error* $\epsilon'_k$, that contrary to the evaluation step error, varies more significantly across the algorithms. While the greedy step error equals to zero in AMPI-Q, it is based on sampling in AMPI-V, and depends on a classifier in CBMPI. To bound the *greedy step error* in AMPI-V and CBMPI, we assume that the action space $\mathcal{A}$ contains only two actions, i.e., $|\mathcal{A}| = 2$. The extension to more than two actions is straightforward along the same line of analysis as in Section 6 of Lazaric et al. [2010b]. The main difference w.r.t. the two action case is that the VC-dimension of the policy space is replaced with its Natarajan dimension. We begin with AMPI-V.

**Lemma 55 (Greedy step error of AMPI-V)** *Let $\mu$ be a distribution over the state space $\mathcal{X}$ and $N$ be the number of states in the rollout set $\mathcal{D}_k$ drawn i.i.d. from $\mu$. For each state $x \in \mathcal{D}_k$ and each action $a \in \mathcal{A}$, we sample $M$ states resulted from taking action $a$ in state $s$. For any $\delta > 0$, the greedy step error $\epsilon'_k$ in the AMPI-V algorithm is bounded as*

$$||\epsilon'_k||_{1,\mu} \leq e'_3(N,\delta) + 2e'_4(N,M,\delta),$$

*with probability at least $1-\delta$, where $h$ is the VC-dimension of the policy space obtained by Eq. 4.3 from the truncation (by $V_{\max}$) of the function space $\mathcal{F}$, and*

$$e'_3(N,\delta) = 16V_{\max}\sqrt{\frac{2}{N}(h\log\frac{eN}{h} + \log\frac{24}{\delta})}\ ,$$
$$e'_4(N,M,\delta) = 8V_{\max}\sqrt{\frac{2}{MN}\big(h\log\frac{eMN}{h} + \log\frac{24}{\delta}\big)}\ .$$

**Proof** See Appendix 4.8.6. ■

We now show how to bound $\epsilon'_k$ in CBMPI. From the definitions of $\epsilon'_k$ (Eq. 4.19) and $\mathcal{L}^{\Pi}_k(\mu;\pi)$ (Eq. 4.14), it is easy to see that $\|\epsilon'_k\|_{1,\mu} = \mathcal{L}^{\Pi}_{k-1}(\mu;\pi_k)$.

This is because

$$\begin{aligned}\epsilon'_k(s) &= \max_{a\in\mathcal{A}} \big[\mathcal{T}^a(\mathcal{T}^{\pi_{k-1}})^m V_{k-2}\big](x) - \big[\mathcal{T}^{\pi_k}(\mathcal{T}^{\pi_{k-1}})^m V_{k-2}\big](x) \qquad \text{(see Eq. 4.12)} \\ &= \max_{a\in\mathcal{A}} Q_{k-1}(x,a) - Q_{k-1}\big(x,\pi_k(x)\big). \qquad \text{(see Eqs. 4.13 and 4.14)}\end{aligned}$$

**Lemma 56 (Greedy step error of CBMPI)** *Let the policy space $\Pi$ defined by the classifier have finite VC-dimension $h = VC(\Pi) < \infty$, and $\mu$ be a distribution over the state space $\mathcal{X}$. Let $N$ be the number of states in $\mathcal{D}'_{k-1}$ drawn i.i.d. from $\mu$, $M$ be the number of rollouts per state-action pair used in the estimation of $\widehat{Q}_{k-1}$, and $\pi_k = \arg\min_{\pi\in\Pi} \widehat{\mathcal{L}}^{\Pi}_{k-1}(\widehat{\mu},\pi)$ be the policy computed at iteration $k-1$ of CBMPI. Then, for any $\delta > 0$, we have*

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

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

$$\begin{aligned} e'_1(N',\delta) &= 16Q_{\max}\sqrt{\frac{2}{N'}\big(h\log\frac{eN'}{h} + \log\frac{32}{\delta}\big)}\ , \\ e'_2(N',M,\delta) &= 8Q_{\max}\sqrt{\frac{2}{MN'}\big(h\log\frac{eMN'}{h} + \log\frac{32}{\delta}\big)}\ . \end{aligned}$$

**Proof** See Appendix 4.8.7. ■

From Lemma 54, we have a bound on $\|\epsilon_k\|_{2,\mu}$ for all the three algorithms. Since $\|\epsilon_k\|_{1,\mu} \le \|\epsilon_k\|_{2,\mu}$, we also have a bound on $\|\epsilon_k\|_{1,\mu}$ for all the algorithms. On the other hand, from Lemmas 55 and 56, we have a bound on $\|\epsilon'_k\|_{1,\mu}$ for the AMPI-V and CMBPI algorithms. This means that for AMPI-V, AMPI-Q ($\epsilon'_k = 0$ for this algorithm), and CBMPI, we can control the RHS of Eqs. 4.22 and 4.23 in $L_1$-norm, which in the context of Theorem 48 means $p = 1$, $q' = 1$, and $q = \infty$. This leads to the main result of this section, finite sample performance bounds for the three proposed algorithms.

**Theorem 57** *Let*

$$d' = \sup_{g\in\mathcal{F},\pi'\in\Pi} \inf_{\pi\in\Pi} \mathcal{L}^{\Pi}_{\pi',g}(\mu;\pi) \qquad \textit{and} \qquad d_m = \sup_{g\in\mathcal{F},\pi\in\Pi} \inf_{f\in\mathcal{F}} \|(\mathcal{T}^{\pi})^m g - f\|_{2,\mu}$$

*where $\mathcal{F}$ is the function space used by the algorithms and $\Pi$ is the policy space used by CBMPI. With the notations of Theorem 48 and Lemmas 54-56, after $k$ iterations, and with probability $1-\delta$, the expected losses $\mathbb{E}_\rho[l_k] = \|l_k\|_{1,\rho}$ of the proposed AMPI algorithms satisfy:*[6]

$$\textit{AMPI-V:} \qquad \|l_k\|_{1,\rho} \le \frac{2(\gamma-\gamma^k)\mathcal{C}_\infty^{1,k,0}}{(1-\gamma)^2}\left(d_m + e_1(N,\frac{\delta}{k}) + e_2(N,\frac{\delta}{k})\right) + \frac{(1-\gamma^k)\mathcal{C}_\infty^{0,k,0}}{(1-\gamma)^2}\left(e'_3(N,\frac{\delta}{k}) + e'_4(N,M,\frac{\delta}{k})\right) + g(k),$$

$$\textit{AMPI-Q:} \qquad \|l_k\|_{1,\rho} \le \frac{2(\gamma-\gamma^k)\mathcal{C}_\infty^{1,k,0}}{(1-\gamma)^2}\left(d_m + e_1(N,\frac{\delta}{k}) + e_2(N,\frac{\delta}{k})\right) + g(k),$$

$$\textit{CBMPI:} \qquad \|l_k\|_{1,\rho} \le \frac{2\gamma^m(\gamma-\gamma^{k-1})\mathcal{C}_\infty^{2,k,m}}{(1-\gamma)^2}\left(d_m + e_1(N,\frac{\delta}{2k}) + e_2(N,\frac{\delta}{2k})\right) + \frac{(1-\gamma^k)\mathcal{C}_\infty^{1,k,0}}{(1-\gamma)^2}\left(d' + e'_1(N',\frac{\delta}{2k}) + e'_2(N',M,\frac{\delta}{2k})\right) + g(k).$$

**Remark 58** *The CBMPI bound in Theorem 57 allows us to restate Remark 52. Assume that we have a fixed budget $B = Nm + N'M|\mathcal{A}|(m+1)$ that we equally divide over the classifier and regressor. Note that the budget is measured in terms of the number of calls to the generative model. Then up to constants and logarithmic factors, the bound has the form*

$$\|l_k\|_{1,\mu} \le O\left(\gamma^m\left(d_m + \sqrt{\frac{m}{B}}\right) + d' + \sqrt{\frac{M|A|m}{B}}\right).$$

[6] As mentioned above, the bounds of AMPI-V and AMPI-Q may also be written with $(p=2, q'=1, q=\infty)$, and $(p=1, q'=2, q=2)$.

*This shows a trade-off in tuning the parameter $m$: a large value of $m$ makes the influence of the regressor's error (both approximation and estimation errors) smaller, but at the same time makes the influence of the estimation error of the classifier bigger, in the final error.*

## 4.6 Experimental Results

For our experiments, we evaluate CBMPI in two different domains: **1)** the *mountain car* problem and **2)** the more challenging game of Tetris. In several experiments, we compare the performance of CBMPI with the DPI algorithm [Lazaric et al., 2010a], which is basically CBMPI without value function approximation[7]. Hence, comparing DPI and CBMPI allows us to highlight the role of the value function approximation.

As discussed in Remark 52, the parameter $m$ in CBMPI balances between the errors in evaluating the value function and the policy. The value function approximation error tends to zero for large values of $m$. Although this would suggest to have large values for $m$, as mentioned in Remark 58, the size of the rollout sets $\mathcal{D}$ and $\mathcal{D}'$ would correspondingly decreases as $N = O(B/m)$ and $N' = O(B/m)$, thus decreasing the accuracy of both the regressor and classifier. 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 case of DPI, when there is no value function, $m$ should be selected in a way to guarantee large enough rollout sets (parameters $N$ and $N'$), and at the same time, a sufficient number of rollouts (parameter $M$).

[7] DPI, as it is presented by Lazaric et al. [2010a], uses infinitely long rollouts and is thus equivalent to CBMPI with $m = \infty$. In practice, implementations of DPI use rollouts that are truncated after some horizon $H$, and is then equivalent to CBMPI with $m = H$ and $v_k = 0$ for all iterations $k$.

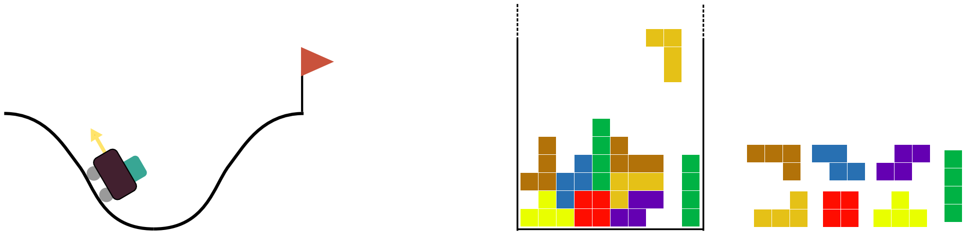


Figure 4.4: ***(Left)*** The Mountain Car (MC) problem in which the car needs to learn to oscillate back and forth in order to build up enough inertia to reach the top of the one-dimensional hill. ***(Right)*** A screen-shot of the game of Tetris and the seven pieces (shapes) used in the game.

One of the objectives of our experiments is to show the role of these parameters in the performance of CBMPI. However, since we almost always obtained our best results with $M = 1$, we only focus on the parameters $m$ and $N$ in our experiments. Moreover, as mentioned in Footnote 3, we implement a more sample efficient version of CBMPI by reusing the rollouts generated for the classifier in the regressor. More precisely, at each iteration $k$, for each state $s^{(i)} \in \mathcal{D}'_k$ and each action $a \in \mathcal{A}$, we generate one rollout of length $m+1$, i.e., $\big(x^{(i)}, a, r_0^{(i)}, x_1^{(i)}, a_1^{(i)}, \dots, a_m^{(i)}, r_m^{(i)}, x_{m+1}^{(i)}\big)$. We then take the rollout of action $\pi_k(x^{(i)})$, select its last $m$ steps, i.e., $\big(x_1^{(i)}, a_1^{(i)}, \dots, a_m^{(i)}, r_m^{(i)}, x_{m+1}^{(i)}\big)$ (note that all the actions here have been taken according to the current policy $\pi_k$), use it to estimate the value function $\widehat{V}_k(x_1^{(i)})$, and add it to the training set of the regressor. This process guarantees to have $N = N'$.

In each experiment, we run the algorithms with the same budget $B$ per iteration. The budget $B$ is the number of next state samples generated by the generative model of the system at each iteration. In DPI and CBMPI, we generate a rollout of length $m+1$ for each state in $\mathcal{D}'$ and each action in $\mathcal{A}$, so, $B = (m+1)N|\mathcal{A}|$. In AMPI-Q, we generate one rollout of length $m$ for each state-action pair in $\mathcal{D}$, and thus, $B = mN$.

### 4.6.1 Mountain Car

*Mountain Car* (MC) is the problem of driving a car up to the top of a one-dimensional hill (see Figure 4.4). The car is not powerful enough to accelerate directly up the hill, and thus, it must learn to oscillate back and forth to build up enough inertia. There are three possible actions: forward (+1), reverse (−1), and stay (0). The reward is −1 for all the states but the goal state at the top of the hill, where the episode ends with a reward 0. The discount factor is set to $\gamma = 0.99$. Each state $x$ consists of the pair $(x_s, \dot{x}_s)$ where $x_s$ is the position of the car and $\dot{x}_s$ is its velocity. We use the formulation described in Dimitrakakis and Lagoudakis [2008b] with uniform noise in $[-0.2, 0.2]$ added to the actions.

In this section, we report the empirical evaluation of CBMPI and AMPI-Q and compare it to DPI and LSPI [Lagoudakis and Parr, 2003a] in the MC problem. In our experiments, we show that CBMPI, by combining policy and value function approximation, can improve over AMPI-Q, DPI, and LSPI.

**Problem Setting**

The value function is approximated using a linear space spanned by a set of radial basis functions (RBFs) evenly distributed over the state space. More precisely, we uniformly divide the 2-dimensional state space into a number of regions and place a Gaussian function at the center of each of them. We set the standard deviation of the Gaussian functions to the width of a region.

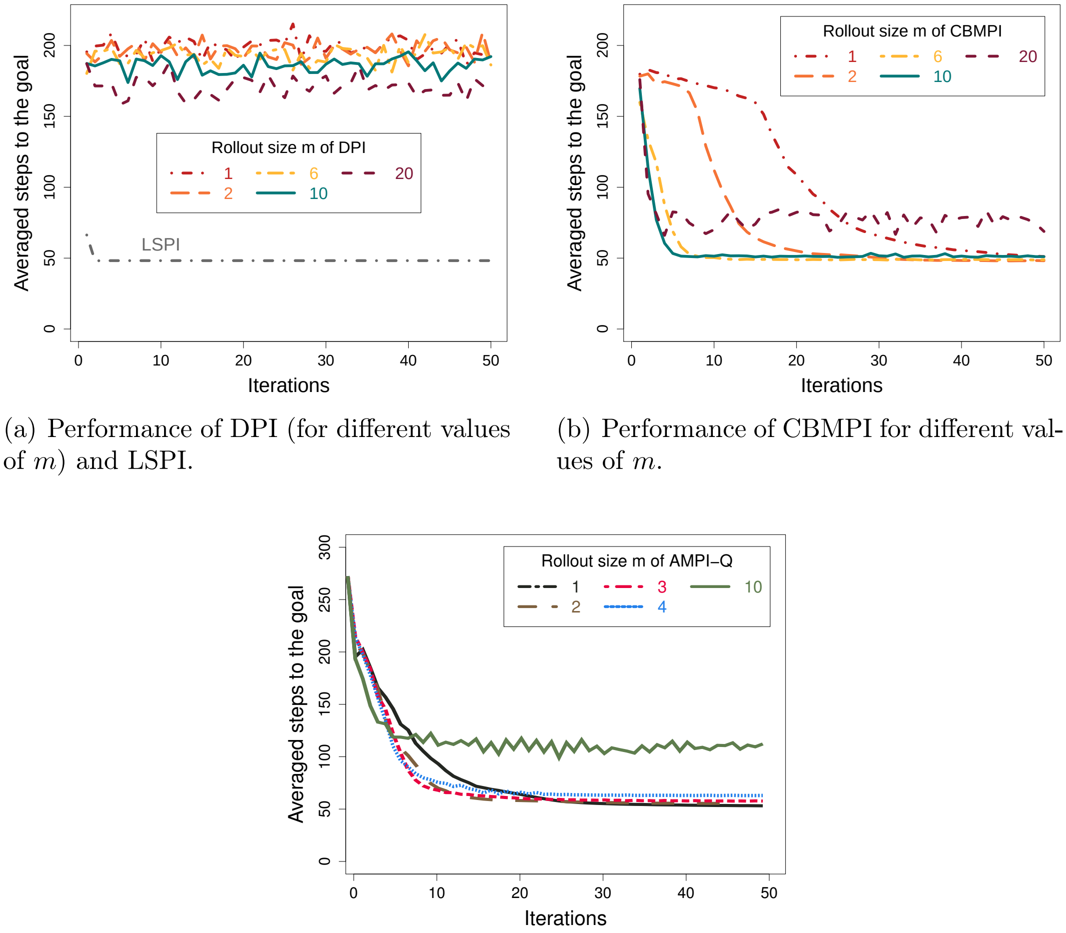
The function space to approximate the action-value function in LSPI is obtained by replicating the state-features for each action. We run LSPI off-policy (i.e., samples are collected once and reused through the iterations of the algorithm). The policy space $\Pi$ (classifier) is defined by a regularized support vector classifier (C-SVC) using the LIBSVM implementation by Chang and Lin [2011]. We use the RBF kernel $\exp(-|u-v|^2)$ and set the cost parameter $C = 1000$. We minimize the classification error instead of directly solving the cost-sensitive multi-class classification step as in Figure 4.3. In fact, the classification error is an upper-bound on the empirical error defined by Eq. 4.16. Finally, the rollout set is sampled uniformly over the state space.

In our MC experiments, the policies learned by the algorithms are evaluated by the number of steps-to-go (average number of steps to reach the goal with a maximum of 300) averaged over $4,000$ independent trials. More precisely, we define the possible starting configurations (positions and velocities) of the car by placing a $20 \times 20$ uniform grid over the state space, and run the policy 10 times from each possible initial configuration. The performance of each algorithm is represented by a learning curve whose value at each iteration is the average number of steps-to-go of the policies learned by the algorithm at that iteration in 1000 separate runs of the algorithm.

We tested the performance of DPI, CBMPI, and AMPI-Q on a wide range of parameters $(m, M, N)$, but only report their performance for the best choice of $M$ (as mentioned earlier, $M = 1$ was the best choice in all the experiments) and different values of $m$.

### Experimental Results

Figure 4.5 shows the learning curves of DPI, CBMPI, AMPI-Q, and LSPI algorithms with budget $B = 4,000$ per iteration and the function space $\mathcal{F}$ composed of a $3 \times 3$ RBF grid. We notice from the results that this space is rich enough to provide a good approximation for the value function components (e.g., in CBMPI, for $(\mathcal{T}^\pi)^m V_{k-1}$ defined by Eq. 4.18). Therefore, LSPI and DPI obtain the best and worst results with about 50 and 160 steps to reach the goal, respectively. The best DPI results are obtained with the large value of $m = 20$. DPI performs better for large values of $m$ because the reward function is constant everywhere except at the goal, and thus, a DPI rollout is only *informative* if it reaches there. We also report the performance of CBMPI and AMPI-Q for different values of $m$. The value function approximation is so accurate that CBMPI and AMPI-Q achieve performance similar to LSPI for $m < 20$. However when $m$ is large ($m = 20$), the performance of these algorithms is worse, because in this case, the rollout set does not have enough elements ($N$ small) to learn the greedy policy and value function well. Note that as we increase $m$ (up to $m = 10$), CBMPI and AMPI-Q converge faster to a good policy.



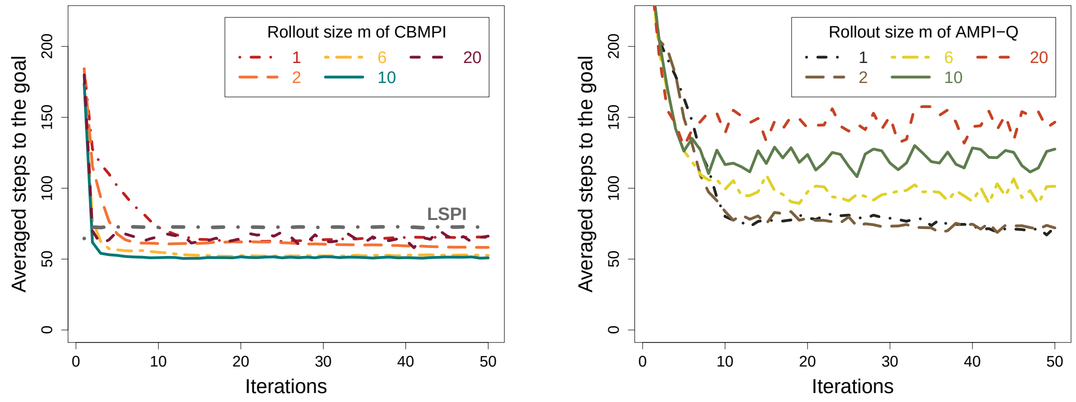
(a) Performance of DPI (for different values of $m$) and LSPI.

(b) Performance of CBMPI for different values of $m$.

(c) Performance of AMPI-Q for different values of $m$.

Figure 4.5: Performance of the policies learned by **(a)** DPI and LSPI, **(b)** CBMPI, and **(c)** AMPI-Q algorithms in the Mountain Car (MC) problem, when we use a $3 \times 3$ RBF grid to approximate the value function. The results are averaged over $1,000$ runs. The total budget $B$ is set to $4,000$ per iteration.

Although this experiment shows that the use of a critic in CBMPI compensates for the truncation of the rollouts (CBMPI performs better than DPI), most of this advantage is due to the richness of the function space $\mathcal{F}$ (LSPI and AMPI-Q perform as well as CBMPI – LSPI even converges faster). Therefore, it seems that it would be more efficient to use LSPI instead of CBMPI in this case. In the next experiment, we study the performance of these algorithms when the function space $\mathcal{F}$ is less rich, composed of a $2 \times 2$ RBF grid. The results are reported in Figure 4.6. Now, the performance of LSPI and AMPI-Q (for the best value of $m = 1$) degrades to 75 and 70 steps, respectively. Although $\mathcal{F}$ is not rich, it still helps CBMPI to outperform DPI. We notice the effect of (weaker) $\mathcal{F}$ in CBMPI when we observe that it no longer converges to its best performance (about 50 steps) for small values of $m = 1$ and $m = 2$. Note that CMBPI outperforms all the other algorithms for $m = 10$ (and even for $m = 6$), while still has a sub-optimal performance for $m = 20$, mainly due to the fact that the rollout set would be too small in this case.



(a) Performance of CBMPI (for different values of $m$) and LSPI.

(b) Performance of AMPI-Q for different values of $m$.

Figure 4.6: Performance of the policies learned by **(a)** CBMPI and LSPI and **(b)** AMPI-Q algorithms in the Mountain Car (MC) problem, when we use a $2 \times 2$ RBF grid to approximate the value function. The results are averaged over $1,000$ runs. The total budget $B$ is set to $4,000$ per iteration.

### 4.6.2 Tetris

Tetris is a popular video game created by Alexey Pajitnov in 1985. The game is played on a grid originally composed of 20 rows and 10 columns, where pieces of 7 different shapes fall from the top (see Figure 4.4). The player has to choose where to place each falling piece by moving it horizontally and rotating it. When a row is filled, it is removed and all the cells above it move one line down. The goal is to remove as many rows as possible before the game is over, i.e., when there is no space available at the top of the grid for the new piece. Here, we consider the variation of the game in which the player knows only the current falling piece, and not the next several coming pieces. This game constitutes an interesting optimization benchmark in which the goal is to find a controller (policy) that maximizes the average (over multiple games) number of lines removed in a game (score).[8] This optimization problem is known to be computationally hard. It contains a huge number of board configurations (about $2^{200} \simeq 1.6 \times 10^{60}$), and even in the case that the sequence of pieces is known in advance, finding the strategy to maximize the score is an NP hard problem [Demaine et al., 2003].

Approximate dynamic programming (ADP) and reinforcement learning (RL) algorithms including approximate value iteration [Tsitsiklis and Van Roy, 1996], $\lambda$-policy iteration ($\lambda$-PI) [Bertsekas and Ioffe, 1996, Scherrer, 2013b], linear programming [Farias and Roy, 2006], and natural policy gradient [Kakade, 2002, Furmston and Barber, 2012] have a long history in the game of Tetris. These algorithms formulate Tetris as an MDP in which the state is defined by the current board configuration plus the falling piece, the actions are the possible orientations of the piece and the possible locations that it can be placed on the board,[9] and the reward is defined such that maximizing the expected sum of rewards from each state coincides with maximizing the score

[8] Note that this number is finite because it was shown that Tetris is a game that ends with probability one [Burgiel, 1997].

[9] The total number of actions at a state depends on the shape of the falling piece, with the maximum of 32 actions in a state, i.e., $|\mathcal{A}| \leq 32$.

from that state. Since the state space is large in Tetris, these methods use value function approximation schemes (often linear approximation) and try to tune the value function parameters (weights) from game simulations. Despite the long history, ADP/RL algorithms, that have been (almost) entirely based on approximating the value function, have not been successful in finding good policies in Tetris. On the other hand, methods that search directly in the space of policies by learning the policy parameters using black-box optimization, such as the cross entropy (CE) method [Rubinstein and Kroese, 2004], have achieved the best reported results in this game (see e.g., Szita and LHorincz 2006, Thiery and Scherrer 2009b). This makes us conjecture that Tetris is a game in which good policies are easier to represent, and thus, learn than their corresponding value functions. So, in order to obtain a good performance with ADP in Tetris, we should use those ADP algorithms that search in a policy space, like CBMPI and DPI, instead of the more traditional ones that search in a value function space.

In this section, we evaluate the performance of CBMPI in Tetris and compare it with DPI, $\lambda$-PI, and CE. In these experiments, we show that CBMPI improves over all the previously reported ADP results. Moreover, it obtains the best results reported in the literature for Tetris in both small $10 \times 10$ and large $10 \times 20$ boards. Although the CBMPI's results are similar to those achieved by the CE method in the large board, it uses considerably fewer (almost 1/10) samples (call to the generative model of the game) than CE.

### Algorithms and Experimental Setup

In this section, we briefly describe the algorithms used in our experiments: the cross entropy (CE) method, our particular implementation of CBMPI, and its slight variation DPI. We refer the readers to Scherrer [2013b] for $\lambda$-PI. We begin by defining some terms and notations. A state $x$ in Tetris consists of two components: the description of the board $b$ and the type of the falling piece $p$. All controllers rely on an evaluation function that gives a value to each possible action at a given state. Then, the controller chooses the action with the highest value. In ADP, algorithms aim at tuning the weights such that the evaluation function approximates well the optimal expected future score from each state. Since the total number of states is large in Tetris, the evaluation function $f$ is usually defined as a linear combination of a set of features $\phi$, i.e., $f(\cdot) = \phi(\cdot)^\top \theta$. We can think of the parameter vector $\theta$ as a policy (controller) whose performance is specified by the corresponding evaluation function $f(\cdot) = \phi(\cdot)^\top \theta$. The features used in Tetris for a state-action pair $(x, a)$ may depend on the description of the board $b'$ resulted from taking action $a$ in state $x$, e.g., the maximum height of $b'$. Computing such features requires the knowledge of the game's dynamics, which is known in Tetris. We consider the following sets of features, plus a constant offset feature:[10]

**(i) Bertsekas Features:** First introduced by Bertsekas and Tsitsiklis [1996], this set of 22 features has been mainly used in the ADP/RL community and consists of: *the number of holes in the board*, *the height of each column*, *the difference in height between two consecutive columns*, and *the maximum height of the board.*

**(ii) Dellacherie-Thiery (D-T) Features:** This set consists of the six features of Dellacherie [Fahey, 2003], i.e., *the landing height of the falling piece*, *the number of eroded piece cells*, *the row transitions*, *the column transitions*, *the number of holes*, and *the number of board wells*; plus 3 additional features proposed in Thiery and Scherrer [2009b], i.e., *the hole depth*, *the number of rows with holes*, and *the pattern diversity feature.* Note that the best policies reported in the literature have been

[10] For a precise definition of the features, see Thiery and Scherrer [2009a] or the documentation of their code [Thiery and Scherrer, 2010b]. Note that the constant offset feature has no incidence when modelling policies while it plays a role to approximate the value functions.

learned using this set of features.

**(iii) RBF Height Features:** These new 5 features are defined as $\exp(\frac{-|c-ih/4|^2}{2(h/5)^2})$, $i = 0, \ldots, 4$, where $c$ is the average height of the columns and $h = 10$ or 20 is the total number of rows in the board.

**The Cross Entropy (CE) Method:** CE [Rubinstein and Kroese, 2004] is an iterative method whose goal is to optimize a function $f$ parameterized by a vector $\theta \in \Theta$ by direct search in the parameter space $\Theta$. Figure 4.7 contains the pseudo-code of the CE algorithm used in our experiments [Szita and LHorincz, 2006, Thiery and Scherrer, 2009b]. At each iteration $k$, we sample $n$ parameter vectors $\{\theta_i\}_{i=1}^{n}$ from a multivariate Gaussian distribution with diagonal covariance matrix $\mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2))$. At the beginning, the parameters of this Gaussian have been set to cover a wide region of $\Theta$. For each parameter $\theta_i$, we play $G$ games and calculate the average number of rows removed by this controller (an estimate of the expected score). We then select $\lfloor \zeta n \rfloor$ of these parameters with the highest score, $\theta'_1, \ldots, \theta'_{\lfloor \zeta n \rfloor}$, and use them to update the mean $\boldsymbol{\mu}$ and variance $\text{diag}(\boldsymbol{\sigma}^2)$ of the Gaussian distribution, as shown in Figure 4.7. This updated Gaussian is used to sample the $n$ parameters at the next iteration. The goal of this update is to sample more parameters from the promising parts of $\Theta$ at the next iteration, and hopefully converge to a global maximum of $f$. In our experiments, in the pseudo-code of Figure 4.7, we set $\zeta = 0.1$ and $\eta = 4$, the best parameters reported in Thiery and Scherrer [2009b]. We also set $n = 1,000$ and $G = 10$ in the small board and $n = 100$ and $G = 1$ in the large board.

**Our Implementation of CBMPI (DPI):** We use the algorithm whose pseudo-code is shown in Figure 4.3. We sampled states from the trajectories generated by a good policy for Tetris, namely the DU controller obtained by Thiery and Scherrer [2009b]. Since this policy is good, the resulted roll-out set is biased towards boards with small height. We noticed from our

```
Input: parameter space $\Theta$, number of parameter vectors $n$, proportion $\zeta \leq 1$, noise $\eta$
Initialize: Set the mean and variance parameters $\boldsymbol{\mu} = (0, 0, \dots, 0)$ and $\boldsymbol{\sigma}^2 = (100, 100, \dots, 100)$
for $k = 1, 2, \dots$ do
    Generate a random sample of $n$ parameter vectors $\{\theta_i\}_{i=1}^{n} \sim \mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$
    For each $\theta_i$, play $G$ games and calculate the average number of rows removed (score) by the controller
    Select $\lfloor \zeta n \rfloor$ parameters with the highest score $\theta'_1, \dots, \theta'_{\lfloor \zeta n \rfloor}$
    Update $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$:  $\mu(j) = \frac{1}{\lfloor \zeta n \rfloor} \sum_{i=1}^{\lfloor \zeta n \rfloor} \theta'_i(j)$  and  $\sigma^2(j) = \frac{1}{\lfloor \zeta n \rfloor} \sum_{i=1}^{\lfloor \zeta n \rfloor} [\theta'_i(j) - \mu(j)]^2 + \eta$
end for
```

Figure 4.7: The pseudo-code of the cross-entropy (CE) method used in our experiments.

experiments that the performance can be significantly improved if we use boards with different heights, and thus, we generated a rollout set in which the board height distribution is more uniform by subsampling the set resulted from the DU policy. This means that better performance can be achieved with more uniform sampling distribution, which is consistent with what we can learn from the CBMPI and DPI performance bounds. We set the initial value function parameter to $\alpha = (0, 0, \dots, 0)$ and select the initial policy $\pi_1$ (policy parameter $\beta$) randomly. We also set the CMA-ES parameters (classifier parameters) to $\zeta = 0.5$, $\eta = 0$, and $n$ equal to 15 times the number of features. Finally, we set the discount factor $\gamma = 1$.

- **Regressor:** We use linear function approximation for the value function, i.e., $\widehat{V}_k(x^{(i)}) = \phi(x^{(i)})\alpha$, where $\phi(\cdot)$ and $\alpha$ are the feature and weight vectors, and minimize the empirical error $\widehat{\mathcal{L}}_k^{\mathcal{F}}(\widehat{\mu}; V)$ using the standard least-squares method.

- **Classifier:** The training set of the classifier is of size $N$ with $x^{(i)} \in \mathcal{D}'_k$ as input and $\big(\max_a \widehat{Q}_k(x^{(i)}, a) - \widehat{Q}_k(x^{(i)}, a_1), \dots, \max_a \widehat{Q}_k(x^{(i)}, a) -$

$\widehat{Q}_k(x^{(i)}, a_{|\mathcal{A}|})\big)$ as output. We use the policies of the form $\pi_\beta(x) = \arg\max_a \psi(x,a)^\top \beta$, where $\psi$ is the policy feature vector (possibly different from the value function feature vector $\phi$) and $\beta \in \mathcal{B}$ is the policy parameter vector. We compute the next policy $\pi_{k+1}$ by minimizing the empirical error $\widehat{\mathcal{L}}_k^\Pi(\widehat{\mu}; \pi_\beta)$, defined by (4.16), using the covariance matrix adaptation evolution strategy (CMA-ES) algorithm [Hansen and Ostermeier, 2001]. In order to evaluate a policy $\beta \in \mathcal{B}$ in CMA-ES, we only need to compute $\widehat{\mathcal{L}}_k^\Pi(\widehat{\mu}; \pi_\beta)$, and given the training set, this procedure does not require any simulation of the game. This is in contrary with policy evaluation in CE that requires playing several games.

### Experiments

In our Tetris experiments, the policies learned by the algorithms are evaluated by their score (average number of rows removed in a game started with an empty board) averaged over 200 games in the small 10 × 10 board and over 20 games in the large 10 × 20 board (since the game takes much more time to complete in the large board). The performance of each algorithm is represented by a learning curve whose value at each iteration is the average score of the policies learned by the algorithm at that iteration in 100 separate runs of the algorithm. In addition to their score, we also evaluate the algorithms by the number of samples they use. In particular, we show that CBMPI/DPI use 10 times fewer samples than CE in the large board. As discussed in Section 4.6.2, this is due the fact that although the classifier in CBMPI/DPI uses a direct search in the space of policies (for the greedy policy), it evaluates each candidate policy using the empirical error of Eq. 4.16, and thus, does not require any simulation of the game (other than those used to estimate the $\widehat{Q}_k$'s in its training set). In fact, the budget $B$ of CBMPI/DPI is fixed in advance by the number of rollouts $NM$ and the rollout's length $m$ as $B = (m+1)NM|\mathcal{A}|$. In contrary, CE evaluates a candidate policy by playing several games, a process that can be extremely costly (sample-wise), especially for good policies in the large board.

We first run the algorithms on the small board to study the role of their parameters and to select the best features and parameters, and then use the selected features and parameters and apply the algorithms to the large board. Finally, we compare the best policies found in our experiments with the best controllers reported in the literature (Tables 4.1 and 4.2).

### Small ($10 \times 10$) Board

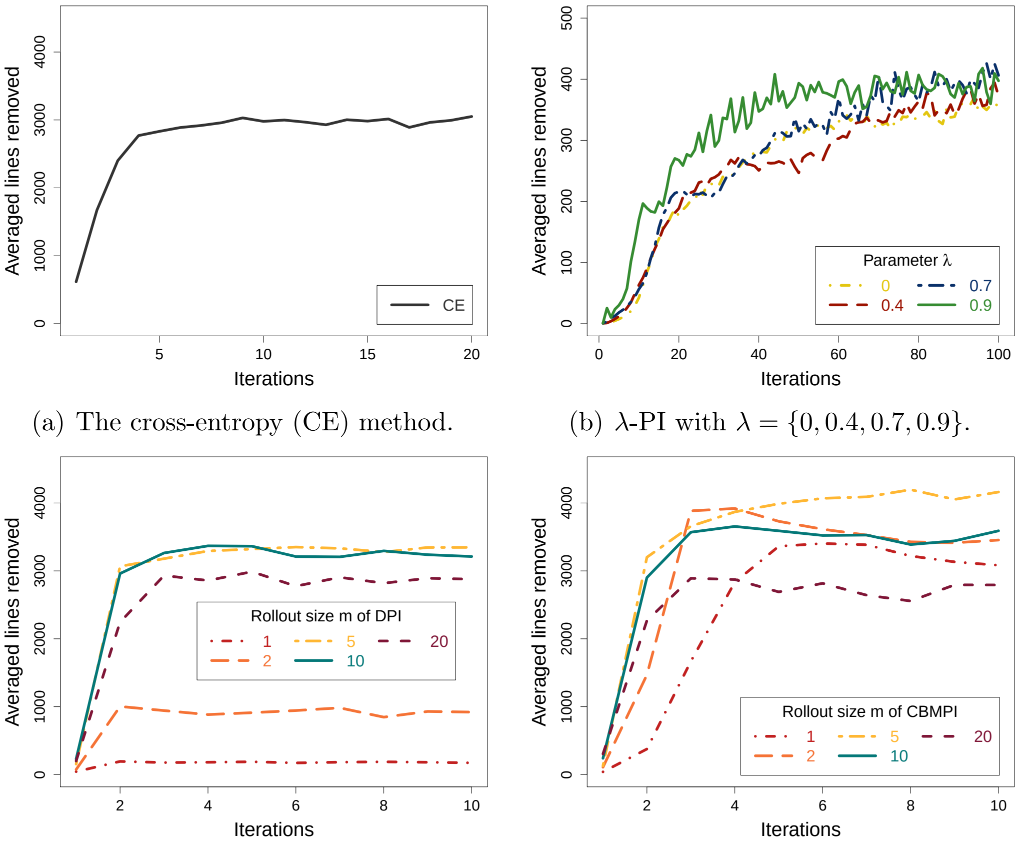
Here we run the algorithms with two different feature sets: *Dellacherie-Thiery (D-T)* and *Bertsekas*, and report their results.

*D-T Features:* Figure 4.8 shows the learning curves of CE, $\lambda$-PI, DPI, and CBMPI algorithms. Here we use D-T features plus constant offset for the evaluation function in CE, the value function in $\lambda$-PI, and the policy in DPI and CBMPI. We ran CBMPI with different choices of features for the value function and “D-T plus the 5 RBF features and constant offset" achieved the best performance (Figure 4.8(d)). The budget of CBMPI and DPI is set to $B = 8,000,000$ per iteration. The CE method reaches the score 3000 after 10 iterations using an average budget $B = 65,000,000$. $\lambda$-PI with the best value of $\lambda$ only manages to score 400. In Figure 4.8(c), we report the performance of DPI for different values of $m$. DPI achieves its best performance for $m = 5$ and $m = 10$ by removing $3,400$ lines on average. As explained in Section 4.6.1, having short rollouts ($m = 1$) in DPI leads to poor action-value estimates $\widehat{Q}$, while having too long rollouts ($m = 20$) decreases the size of the training set of the classifier $N$. CBMPI outperforms the other algorithms, including CE, by reaching the score of $4,200$ for $m = 5$. This value of $m = 5$ corresponds to $N = \frac{8000000}{(5+1)\times 32} \approx 42,000$. Note that unlike DPI, CBMPI achieves good performance with very short rollouts $m = 1$. This indicates that CBMPI is able to approximate the value function well, and as a result, build a more accurate training set for its classifier than

DPI. Despite this improvement, the good results obtained by DPI in Tetris indicate that with small rollout horizons like $m = 5$, one has already fairly accurate action value estimates in order to detect greedy actions accurately (at each iteration).

The results of Figure 4.8 show that an ADP algorithm, namely CBMPI, outperforms the CE method using a similar budget (80 vs. 65 millions after 10 iterations). Note that CBMPI takes less iterations to converge than CE. More generally Figure 4.8 confirms the superiority of the policy search and classification-based PI methods to value-function based ADP algorithms ($\lambda$-PI). This suggests that the D-T features are more suitable to represent the policies than the value functions in Tetris.

*Bertsekas Features:* Figure 4.9(a)-(c) show the performance of CE, $\lambda$-PI, DPI, and CBMPI algorithms. Here all the approximations in the algorithms are with the Bertsekas features plus constant offset. CE achieves the score 500 after about 60 iterations and outperforms $\lambda$-PI with score 350. It is clear that the Bertsekas features lead to much weaker results than those obtained by the D-T features (Figure 4.8) for all the algorithms. We may conclude then that the D-T features are more suitable than the Bertsekas features to represent both value functions and policies in Tetris. In DPI and CBMPI, we managed to obtain results similar to CE, only after multiplying the per iteration budget $B$ used in the D-T experiments by 10. Indeed, CBMPI and DPI need more samples to solve the classification and regression problems in this 22-dimensions weight vector space than with the 9 D-T features. Moreover, in the classifier, the minimization of the empirical error through the CMA-ES method (see Eq. 4.11) was converging most of the times to a local minimum. To solve this issue, we run multiple times the minimization problem with different starting points and small initial covariance matrices for the Gaussian distribution in order to force local exploration of different parts of the weight vector areas. However, CBMPI and CE use the same num-



(a) The cross-entropy (CE) method.

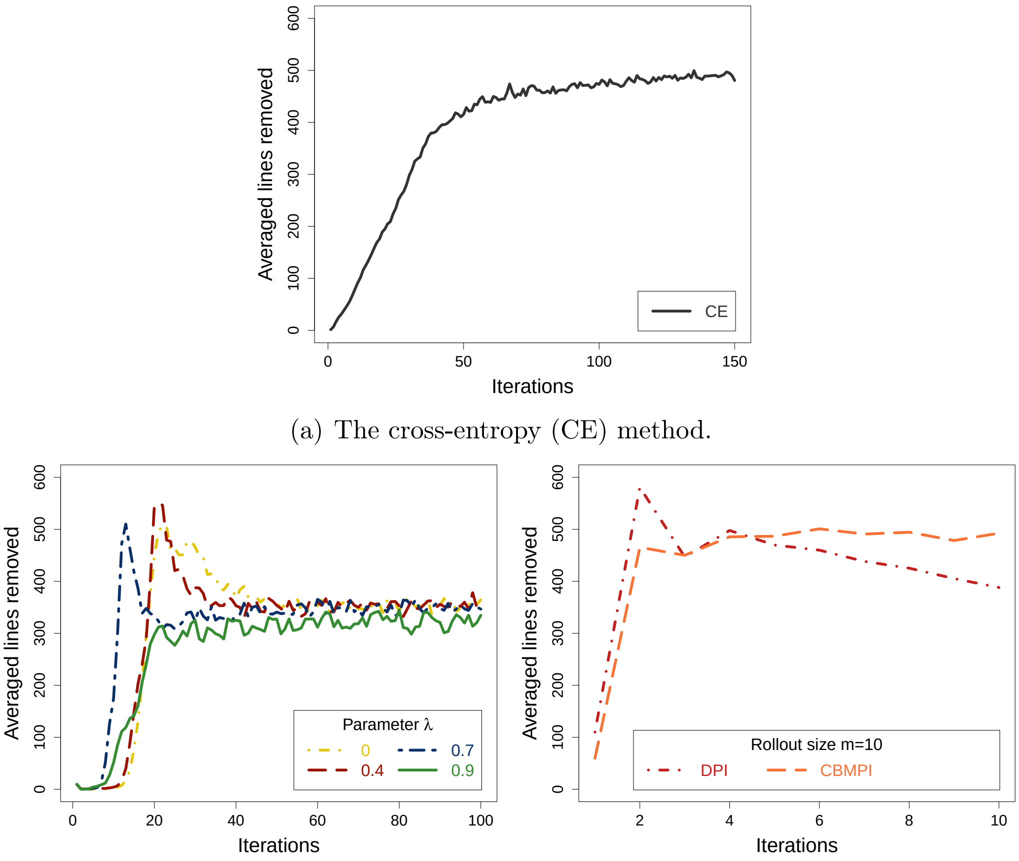
(b) $\lambda$-PI with $\lambda = \{0, 0.4, 0.7, 0.9\}$.

(c) DPI with budget $B = 8,000,000$ per iteration and $m = \{1, 2, 5, 10, 20\}$.

(d) CBMPI with budget $B = 8,000,000$ per iteration and $m = \{1, 2, 5, 10, 20\}$.

Figure 4.8: Learning curves of CE, $\lambda$-PI, DPI, and CBMPI algorithms using the 9 Dellacherie-Thiery (D-T) features on the small $10 \times 10$ board. The results are averaged over 100 runs of the algorithms.

ber of samples, 150, 000, 000, when they converge after 2 and 60 iterations, respectively (see Figure 4.9). Note that DPI and CBMPI obtain the same performance, which means that the use of a value function approximation by CBMPI does not lead to a significant performance improvement over DPI. At the end, we tried several values of $m$ in this setting among which $m = 10$ achieved the best performance for both DPI and CBMPI.



(a) The cross-entropy (CE) method.

(b) $\lambda$-PI with $\lambda = \{0, 0.4, 0.7, 0.9\}$.

(c) DPI (dash-dotted line) & CBMPI (dash line) with budget $B = 80,000,000$ per iteration and $m = 10$.

Figure 4.9: (a)-(c) Learning curves of CE, $\lambda$-PI, DPI, and CBMPI algorithms using the 22 Bertsekas features on the small $10 \times 10$ board.

### Large ($10 \times 20$) Board

We now use the best parameters and features in the small board experiments, run CE, DPI, and CBMPI algorithms in the large board, and report their results in Figure 4.10 (left). We also report the results of $\lambda$-PI in the large board in Figure 4.10 (right). The per iteration budget of DPI and CBMPI is set to $B = 16,000,000$. While $\lambda$-PI with per iteration budget $620,000$, at its best, achieves the score of $2,500$, DPI and CBMPI, with $m = 10$, reach the scores of $12,000,000$ and $16,000,000$ after 3 and 6 iterations, respectively. Although CE outperforms CBMPI with the score of $20,000,000$ after 6 iterations, this is achieved with almost 13 times more samples: after 8 iterations, CBMPI and CE use $128,000,000$ and $1,700,000,000$ samples, respectively.

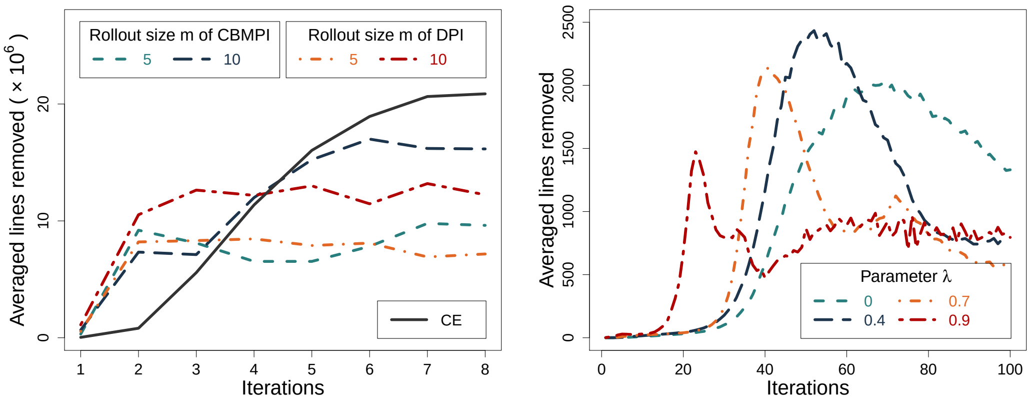


Figure 4.10: Learning curves of CBMPI, DPI and CE (left), and $\lambda$-PI (right) using the 9 features listed in Table 4.2 on the large $10 \times 20$ board. The total budget $B$ of CBMPI and DPI is set to 16,000,000 per iteration.

### Comparison of the Best Policies

So far the reported scores for each algorithm was averaged over the policies learned in 100 separate runs. Here we select the best policies observed in our all experiments and compute their scores more accurately by averaging over $10,000$ games. We then compare these results with the best policies reported in the literature, i.e., DU and BDU [Thiery and Scherrer, 2009b] in both small and large boards in Table 4.1. The DT-10 and DT-20 policies, whose weights and features are given in Table 4.2, are policies learned by CBMPI with D-T features in the small and large boards, respectively. As shown in Table 4.1, DT-10 removes $5,000$ lines and outperforms DU, BDU, and DT-20 in the small board. Note that DT-10 is the only policy among these four that has been learned in the small board. In the large board, DT-20 obtains the score of $51,000,000$ and not only outperforms the other three policies, but also achieves the best reported result in the literature (to the best of our knowledge). We observed in our experiments that the policies learned by CBMPI have more variance in their performance than those learned by CE. This is why in the large board, although the policies learned by CE have better average performance than CBMPI (see Figure 4.10 (left)), the best policy learned by CBMPI outperforms that learned by CE (see Table 4.1).

| **Boards \ Policies** | **DU** | **BDU** | **DT-10** | **DT-20** |
|---|---|---|---|---|
| Small $(10 \times 10)$ board | 3800 | 4200 | 5000 | 4300 |
| Large $(10 \times 20)$ board | $31,000,000$ | $36,000,000$ | $29,000,000$ | $51,000,000$ |

Table 4.1: Average (over $10,000$ games) score of DU, BDU, DT-10, and DT-20 policies.

| **feature** | **weight** | | **feature** | **weight** | | **feature** | **weight** | |
|---|---|---|---|---|---|---|---|---|
| landing height | -2.18 | -2.68 | column transitions | -3.31 | -6.32 | hole depth | -0.81 | -0.43 |
| eroded piece cells | 2.42 | 1.38 | holes | 0.95 | 2.03 | rows w/ holes | -9.65 | -9.48 |
| row transitions | -2.17 | -2.41 | board wells | -2.22 | -2.71 | diversity | 1.27 | 0.89 |

Table 4.2: The weights of the 9 Dellacherie-Thiery features in DT-10 (left) and DT-20 (right) policies.

## 4.7 Conclusion

In this paper, we considered a dynamic programming (DP) scheme for Markov decision processes, known as modified policy iteration (MPI). We proposed three original approximate MPI (AMPI) algorithms that are extensions of existing approximate DP (ADP) algorithms: fitted-value iteration, fitted-Q iteration, and classification-based policy iteration. We reported a general error propagation analysis for AMPI that unifies those for approximate policy and value iteration. We instantiated this analysis for the three algorithms that we introduced, which led to a finite-sample analysis of their guaranteed performance. For the last introduced algorithm, CBMPI, our analysis indicated that the main parameter of MPI controls the balance of errors (between value function approximation and estimation of the greedy policy). The role of this parameter was illustrated for all algorithms on two benchmark problems: Mountain Car and Tetris. Remarkably, in the game of Tetris, CBMPI showed advantages over all previous approaches: it significantly outperforms previous ADP approaches, and is competitive with black-box optimization techniques—the current state of the art for this domain—while using fewer samples. In particular, CBMPI led to what is to our knowledge the currently best Tetris controller, removing $51,000,000$ lines on average. Interesting future work includes 1) the adaptation and precise analysis of our three algorithms to the computation of non-stationary policies[11] and 2) considering problems with large action spaces, for which the methods we have proposed here are likely to be limited.

## 4.8 Appendix

### 4.8.1 Proof of Lemma 43

Before we start, we recall the following definitions:

$$\begin{aligned}
b_k &= V_k - \mathcal{T}^{\pi_{k+1}} V_k, \\
d_k &= V^* - (\mathcal{T}^{\pi_k})^m V_{k-1} = V^* - (V_k - \epsilon_k), \\
s_k &= (\mathcal{T}^{\pi_k})^m V_{k-1} - V^{\pi_k} = (V_k - \epsilon_k) - V^{\pi_k}.
\end{aligned}$$

[11] We recently showed that considering a variation of AMPI for computing non-stationary policies allows improving the $\frac{1}{(1-\gamma)^2}$ constant [Lesner and Scherrer, 2013].

**Bounding $b_k$**

$$\begin{aligned}
b_k &= V_k - \mathcal{T}^{\pi_{k+1}} V_k \\
&= V_k - \mathcal{T}^{\pi_k} V_k + \mathcal{T}^{\pi_k} V_k - \mathcal{T}^{\pi_{k+1}} V_k \\
&\overset{(a)}{\leq} V_k - \mathcal{T}^{\pi_k} V_k + \epsilon'_{k+1} \\
&= V_k - \epsilon_k - \mathcal{T}^{\pi_k} V_k + \gamma P^{\pi_k} \epsilon_k + \epsilon_k - \gamma P^{\pi_k} \epsilon_k + \epsilon'_{k+1} \\
&\overset{(b)}{=} V_k - \epsilon_k - \mathcal{T}^{\pi_k}(V_k - \epsilon_k) + (I - \gamma P^{\pi_k})\epsilon_k + \epsilon'_{k+1}. \qquad (4.25)
\end{aligned}$$

Using the definition of $x_k$, i.e.,

$$x_k \triangleq (I - \gamma P^{\pi_k})\epsilon_k + \epsilon'_{k+1}, \qquad (4.26)$$

we may write Eq. 4.25 as

$$\begin{aligned}
b_k &\leq V_k - \epsilon_k - \mathcal{T}^{\pi_k}(V_k - \epsilon_k) + x_k \\
&\overset{(c)}{=} (\mathcal{T}^{\pi_k})^m V_{k-1} - \mathcal{T}^{\pi_k}(\mathcal{T}^{\pi_k})^m V_{k-1} + x_k \\
&= (\mathcal{T}^{\pi_k})^m V_{k-1} - (\mathcal{T}^{\pi_k})^m(\mathcal{T}^{\pi_k} V_{k-1}) + x_k \\
&\overset{(d)}{=} (\gamma P^{\pi_k})^m (V_{k-1} - \mathcal{T}^{\pi_k} V_{k-1}) + x_k \\
&= (\gamma P^{\pi_k})^m b_{k-1} + x_k. \qquad (4.27)
\end{aligned}$$

**(a)** From the definition of $\epsilon'_{k+1}$, we have $\forall \pi' \;\; \mathcal{T}^{\pi'} V_k \leq \mathcal{T}^{\pi_{k+1}} V_k + \epsilon'_{k+1}$, 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 $\mathcal{T}^{\pi_k}(V + V') = \mathcal{T}^{\pi_k} V + \gamma P^{\pi_k} V'$.
**(c)** This is from the definition of $\epsilon_k$, i.e., $V_k = (\mathcal{T}^{\pi_k})^m V_{k-1} + \epsilon_k$.
**(d)** This step is due to the fact that for every $V$ and $V'$, any $m$, we have $(\mathcal{T}^{\pi_k})^m V - (\mathcal{T}^{\pi_k})^m V' = (\gamma P^{\pi_k})^m (V - V')$.

**Bounding** $d_k$

$$\begin{aligned}
d_{k+1} &= V^* - (\mathcal{T}^{\pi_{k+1}})^m V_k \\
&= \mathcal{T}^{\pi^*} V^* - \mathcal{T}^{\pi^*} V_k + \mathcal{T}^{\pi^*} V_k - \mathcal{T}^{\pi_{k+1}} V_k + \mathcal{T}^{\pi_{k+1}} V_k - (\mathcal{T}^{\pi_{k+1}})^m V_k \\
&\overset{(a)}{\leq} \gamma P^{\pi^*}(V^* - V_k) + \epsilon'_{k+1} + g_{k+1} \\
&= \gamma P^{\pi^*}(V^* - V_k) + \gamma P^{\pi^*}\epsilon_k - \gamma P^{\pi^*}\epsilon_k + \epsilon'_{k+1} + g_{k+1} \\
&\overset{(b)}{=} \gamma P^{\pi^*}\big(V^* - (V_k - \epsilon_k)\big) + y_k + g_{k+1} \\
&= \gamma P^{\pi^*} d_k + y_k + g_{k+1} \\
&\overset{(c)}{=} \gamma P^{\pi^*} d_k + y_k + \sum_{j=1}^{m-1} (\gamma P^{\pi_{k+1}})^j b_k. \qquad (4.28)
\end{aligned}$$

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

$$g_{k+1} \triangleq \mathcal{T}^{\pi_{k+1}} V_k - (\mathcal{T}^{\pi_{k+1}})^m V_k. \qquad (4.29)$$

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

$$y_k \triangleq -\gamma P^{\pi^*}\epsilon_k + \epsilon'_{k+1}. \qquad (4.30)$$

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

$$\begin{aligned} g_{k+1} &= \mathcal{T}^{\pi_{k+1}} V_k - (\mathcal{T}^{\pi_{k+1}})^m V_k \\ &= \sum_{j=1}^{m-1} \left[ (\mathcal{T}^{\pi_{k+1}})^j V_k - (\mathcal{T}^{\pi_{k+1}})^{j+1} V_k \right] \\ &= \sum_{j=1}^{m-1} \left[ (\mathcal{T}^{\pi_{k+1}})^j V_k - (\mathcal{T}^{\pi_{k+1}})^j (\mathcal{T}^{\pi_{k+1}} V_k) \right] \\ &= \sum_{j=1}^{m-1} (\gamma P^{\pi_{k+1}})^j (V_k - \mathcal{T}^{\pi_{k+1}} V_k) \\ &= \sum_{j=1}^{m-1} (\gamma P^{\pi_{k+1}})^j b_k. \end{aligned} \tag{4.31}$$

**Bounding** $s_k$ With some slight abuse of notation, we have

$$V^{\pi_k} = (\mathcal{T}^{\pi_k})^\infty V_k$$

and thus:

$$
\begin{aligned}
s_k &= (\mathcal{T}^{\pi_k})^m V_{k-1} - V_{\pi_k} \\
&\overset{(a)}{=} (\mathcal{T}^{\pi_k})^m V_{k-1} - (\mathcal{T}^{\pi_k})^\infty V_{k-1} \\
&= (\mathcal{T}^{\pi_k})^m V_{k-1} - (\mathcal{T}^{\pi_k})^m (\mathcal{T}^{\pi_k})^\infty V_{k-1} \\
&= (\gamma P^{\pi_k})^m \big( V_{k-1} - (\mathcal{T}^{\pi_k})^\infty V_{k-1} \big) \\
&= (\gamma P^{\pi_k})^m \sum_{j=0}^{\infty} \big[ (\mathcal{T}^{\pi_k})^j V_{k-1} - (\mathcal{T}^{\pi_k})^{j+1} V_{k-1} \big] \\
&= (\gamma P^{\pi_k})^m \big( \sum_{j=0}^{\infty} \big[ (\mathcal{T}^{\pi_k})^j V_{k-1} - (\mathcal{T}^{\pi_k})^j \mathcal{T}^{\pi_k} V_{k-1} \big] \\
&= (\gamma P^{\pi_k})^m \Big( \sum_{j=0}^{\infty} (\gamma P^{\pi_k})^j \Big) (V_{k-1} - \mathcal{T}^{\pi_k} V_{k-1}) \\
&= (\gamma P^{\pi_k})^m (I - \gamma P^{\pi_k})^{-1} (V_{k-1} - \mathcal{T}^{\pi_k} V_{k-1}) \\
&= (\gamma P^{\pi_k})^m (I - \gamma P^{\pi_k})^{-1} b_k.
\end{aligned}
\tag{4.32}
$$

**(a)** For any $V$, we have $V^{\pi_k} = (\mathcal{T}^{\pi_k})^\infty V$. This step follows by setting $V = V_{k-1}$, i.e., $V^{\pi_k} = (\mathcal{T}^{\pi_k})^\infty V_{k-1}$.

### 4.8.2 Proof of Lemma 45

We begin by focusing our analysis on AMPI. Here we are interested in bounding the loss $l_k = V^* - V^{\pi_k} = d_k + s_k$.

By induction, from Eqs. 4.27 and 4.28, we obtain

$$
b_k \leq \sum_{i=1}^{k} \Gamma^{m(k-i)} x_i + \Gamma^{mk} b_0, \tag{4.33}
$$

$$
d_k \leq \sum_{j=0}^{k-1} \Gamma^{k-1-j} \Big( y_j + \sum_{l=1}^{m-1} \Gamma^l b_j \Big) + \Gamma^k d_0. \tag{4.34}
$$

in which we have used the notation introduced in Definition 44. In Eq. 4.34, we also used the fact that from Eq. 4.31, we may write $g_{k+1} = \sum_{j=1}^{m-1} \Gamma^j b_k$. Moreover, we may rewrite Eq. 4.32 as

$$s_k = \Gamma^m \sum_{j=0}^{\infty} \Gamma^j b_{k-1} = \sum_{j=0}^{\infty} \Gamma^{m+j} b_{k-1}. \tag{4.35}$$

**Bounding $l_k$** From Eqs. 4.33 and 4.34, we may write

$$\begin{aligned} d_k &\leq \sum_{j=0}^{k-1} \Gamma^{k-1-j} \left( y_j + \sum_{l=1}^{m-1} \Gamma^l \Big( \sum_{i=1}^{j} \Gamma^{m(j-i)} x_i + \Gamma^{mj} b_0 \Big) \right) + \Gamma^k d_0 \\ &= \sum_{i=1}^{k} \Gamma^{i-1} y_{k-i} + \sum_{j=0}^{k-1} \sum_{l=1}^{m-1} \sum_{i=1}^{j} \Gamma^{k-1-j+l+m(j-i)} x_i + z_k, \end{aligned} \tag{4.36}$$

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. 4.36 may be written as

$$\begin{aligned} \sum_{j=0}^{k-1} \sum_{l=1}^{m-1} \sum_{i=1}^{j} \Gamma^{k-1-j+l+m(j-i)} x_i &= \sum_{i=1}^{k-1} \sum_{j=i}^{k-1} \sum_{l=1}^{m-1} \Gamma^{k-1+l+j(m-1)-mi} x_i \\ &= \sum_{i=1}^{k-1} \sum_{j=mi+k-i}^{mk-1} \Gamma^{j-mi} x_i \\ &= \sum_{i=1}^{k-1} \sum_{j=k-i}^{m(k-i)-1} \Gamma^j x_i \\ &= \sum_{i=1}^{k-1} \sum_{j=i}^{mi-1} \Gamma^j x_{k-i}. \end{aligned} \tag{4.37}$$

Using Eq. 4.37, we may write Eq. 4.36 as

$$d_k \leq \sum_{i=1}^{k} \Gamma^{i-1} y_{k-i} + \sum_{i=1}^{k-1} \sum_{j=i}^{mi-1} \Gamma^j x_{k-i} + z_k. \tag{4.38}$$

Similarly, from Eqs. 4.35 and 4.33, we have

$$\begin{aligned}
s_k &\leq \sum_{j=0}^{\infty} \Gamma^{m+j} \Big( \sum_{i=1}^{k-1} \Gamma^{m(k-1-i)} x_i + \Gamma^{m(k-1)} b_0 \Big) \\
&= \sum_{j=0}^{\infty} \Big( \sum_{i=1}^{k-1} \Gamma^{m+j+m(k-1-i)} x_i + \Gamma^{m+j+m(k-1)} b_0 \Big) \\
&= \sum_{i=1}^{k-1} \sum_{j=0}^{\infty} \Gamma^{j+m(k-i)} x_i + \sum_{j=0}^{\infty} \Gamma^{j+mk} b_0 = \sum_{i=1}^{k-1} \sum_{j=0}^{\infty} \Gamma^{j+mi} x_{k-i} + \sum_{j=mk}^{\infty} \Gamma^j b_0 \\
&= \sum_{i=1}^{k-1} \sum_{j=mi}^{\infty} \Gamma^j x_{k-i} + z'_k, \qquad (4.39)
\end{aligned}$$

where we used the following definition

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

Finally, using the bounds in Eqs. 4.38 and 4.39, we obtain the following bound on the loss

$$\begin{aligned}
l_k &\leq d_k + s_k \\
&\leq \sum_{i=1}^{k} \Gamma^{i-1} y_{k-i} + \sum_{i=1}^{k-1} \Big( \sum_{j=i}^{mi-1} \Gamma^j + \sum_{j=mi}^{\infty} \Gamma^j \Big) x_{k-i} + z_k + z'_k \\
&= \sum_{i=1}^{k} \Gamma^{i-1} y_{k-i} + \sum_{i=1}^{k-1} \sum_{j=i}^{\infty} \Gamma^j x_{k-i} + \eta_k, \qquad (4.40)
\end{aligned}$$

where we used the following definition

$$\eta_k \stackrel{\Delta}{=} z_k + z'_k = \sum_{j=k}^{\infty} \Gamma^j b_0 + \Gamma^k d_0. \tag{4.41}$$

Note that we have the following relation between $b_0$ and $d_0$

$$\begin{aligned} b_0 &= V_0 - \mathcal{T}^{\pi_1} V_0 \\ &= V_0 - V^* + \mathcal{T}^{\pi^*} V^* - \mathcal{T}^{\pi^*} V_0 + \mathcal{T}^{\pi^*} V_0 - \mathcal{T}^{\pi_1} V_0 \\ &\leq (I - \gamma P^{\pi^*})(-d_0) + \epsilon'_1, \end{aligned} \tag{4.42}$$

In Eq. 4.42, we used the fact that $V^* = \mathcal{T}^{\pi^*} V^*$, $\epsilon_0 = 0$, and $\mathcal{T}^{\pi^*} V_0 - \mathcal{T}^{\pi_1} V_0 \leq \epsilon'_1$ (this is because the policy $\pi_1$ is $\epsilon'_1$-greedy w.r.t. $V_0$). As a result, we may write $|\eta_k|$ either as

$$\begin{aligned} |\eta_k| &\leq \sum_{j=k}^{\infty} \Gamma^j \big[ (I - \gamma P^{\pi^*}) |d_0| + |\epsilon'_1| \big] + \Gamma^k |d_0| \\ &\leq \sum_{j=k}^{\infty} \Gamma^j \big[ (I + \Gamma^1) |d_0| + |\epsilon'_1| \big] + \Gamma^k |d_0| \\ &= 2 \sum_{j=k}^{\infty} \Gamma^j |d_0| + \sum_{j=k}^{\infty} \Gamma^j |\epsilon'_1|, \end{aligned} \tag{4.43}$$

or using the fact that from Eq. 4.42, we have $d_0 \leq (I - \gamma P^{\pi^*})^{-1}(-b_0 + \epsilon'_1)$, as

$$\begin{aligned}
|\eta_k| &\leq \sum_{j=k}^{\infty} \Gamma^j |b_0| + \Gamma^k \sum_{j=0}^{\infty} (\gamma P^{\pi^*})^j \big(|b_0| + |\epsilon'_1|\big) \\
&= \sum_{j=k}^{\infty} \Gamma^j |b_0| + \Gamma^k \sum_{j=0}^{\infty} \Gamma^j \big(|b_0| + |\epsilon'_1|\big) \\
&= 2 \sum_{j=k}^{\infty} \Gamma^j |b_0| + \sum_{j=k}^{\infty} \Gamma^j |\epsilon'_1|. \qquad (4.44)
\end{aligned}$$

Now, using the definitions of $x_k$ and $y_k$ in Eqs. 4.26 and 4.30, the bound on $|\eta_k|$ in Eq. 4.43 or 4.44, and the fact that $\epsilon_0 = 0$, we obtain

$$\begin{aligned}
|l_k| &\leq \sum_{i=1}^{k} \Gamma^{i-1} \big[\Gamma^1 |\epsilon_{k-i}| + |\epsilon'_{k-i+1}|\big] + \sum_{i=1}^{k-1} \sum_{j=i}^{\infty} \Gamma^j \big[(I + \Gamma^1)|\epsilon_{k-i}| + |\epsilon'_{k-i+1}|\big] + |\eta_k| \\
&= \sum_{i=1}^{k-1} \Big(\Gamma^i + \sum_{j=i}^{\infty} (\Gamma^j + \Gamma^{j+1})\Big) |\epsilon_{k-i}| + \Gamma^k |\epsilon_0| \qquad (4.45) \\
&\quad + \sum_{i=1}^{k-1} \Big(\Gamma^{i-1} + \sum_{j=i}^{\infty} \Gamma^j\Big) |\epsilon'_{k-i+1}| + \Gamma^{k-1} |\epsilon'_1| + \sum_{j=k}^{\infty} \Gamma^j |\epsilon'_1| + h(k) \\
&= 2 \sum_{i=1}^{k-1} \sum_{j=i}^{\infty} \Gamma^j |\epsilon_{k-i}| + \sum_{i=1}^{k-1} \sum_{j=i-1}^{\infty} \Gamma^j |\epsilon'_{k-i+1}| + \sum_{j=k-1}^{\infty} \Gamma^j |\epsilon'_1| + h(k) \\
&= 2 \sum_{i=1}^{k-1} \sum_{j=i}^{\infty} \Gamma^j |\epsilon_{k-i}| + \sum_{i=0}^{k-1} \sum_{j=i}^{\infty} \Gamma^j |\epsilon'_{k-i}| + h(k), \qquad (4.46)
\end{aligned}$$

where we used the following definition

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

depending on whether one uses Eq. 4.43 or Eq. 4.44.

We end this proof by adapting the error propagation to CBMPI. As expressed by Eqs. 4.19 and 4.20 in Section 4.4, an analysis of CBMPI can be deduced from that we have just done by replacing $v_k$ with the auxiliary variable $w_k = (\mathcal{T}^{\pi_k})^m V_{k-1}$ and $\epsilon_k$ with $(\gamma P^{\pi_k})^m \epsilon_{k-1} = \Gamma^m \epsilon_{k-1}$. Therefore, using the fact that $\epsilon_0 = 0$, we can rewrite the bound of Eq. 4.46 for CBMPI as follows:

$$\begin{aligned} l_k &\leq 2\sum_{i=1}^{k-1}\sum_{j=i}^{\infty}\Gamma^{j+m}|\epsilon_{k-i-1}| + \sum_{i=0}^{k-1}\sum_{j=i}^{\infty}\Gamma^{j}|\epsilon'_{k-i}| + h(k) \\ &= 2\sum_{i=1}^{k-2}\sum_{j=m+i}^{\infty}\Gamma^{j}|\epsilon_{k-i-1}| + \sum_{i=0}^{k-1}\sum_{j=i}^{\infty}\Gamma^{j}|\epsilon'_{k-i}| + h(k). \end{aligned} \tag{4.47}$$

### 4.8.3 Proof of Lemma 47

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

$$\rho\Gamma^t|z| = \left\|\Gamma^t|z|\right\|_{1,\rho} \leq \gamma^t c_q(t)\|z\|_{q',\mu} = \gamma^t c_q(t)\left(\mu|z|^{q'}\right)^{\frac{1}{q'}}. \tag{4.48}$$

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. 4.48, 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 $K$.

### 4.8.4 Proof of Theorem 48

**Proof** We only detail the proof for AMPI (the proof being similar for CBMPI). 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 $i \in \mathcal{I}$

$$g_i = \begin{cases} 2\epsilon_{k-i} & \text{if} \quad 1 \leq i \leq k-1, \\ \epsilon'_{k-(i-k)} & \text{if} \quad k \leq i \leq 2k-1, \\ 2d_0 \text{ (or } 2b_0) & \text{if} \quad i = 2k, \end{cases} \qquad \text{and}$$

$$\mathcal{J}_i = \begin{cases} \{i, i+1, \cdots\} & \text{if} \quad 1 \leq i \leq k-1, \\ \{i-k, i-k+1, \cdots\} & \text{if} \quad k \leq i \leq 2k-1, \\ \{k, k+1, \cdots\} & \text{if} \quad i = 2k. \end{cases}$$

Note that here we have divided the terms in the point-wise bound of Lemma 45 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 45 may be rewritten as

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

The result follows by applying Lemma 47 and noticing that $\sum_{i=i_0}^{k-1} \sum_{j=i}^{\infty} \gamma^j = \frac{\gamma^{i_0} - \gamma^k}{(1-\gamma)^2}$. ■

Here in order to show the flexibility of Lemma 47, we group the terms differently and derive an alternative $L_p$-bound for the loss of AMPI and CBMPI. 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 59** *With the notations of Theorem 48, after k iterations, the loss of AMPI satisfies*

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

*while the loss of CBMPI satisfies*

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

**Proof** Again, we only detail the proof for AMPI (the proof being similar for CBMPI). We define $\mathcal{I}$, $(g_i)$ and $(\mathcal{J}_i)$ as in the proof of Theorem 48. 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 47. ■

## 4.8.5 Proof of Lemma 54

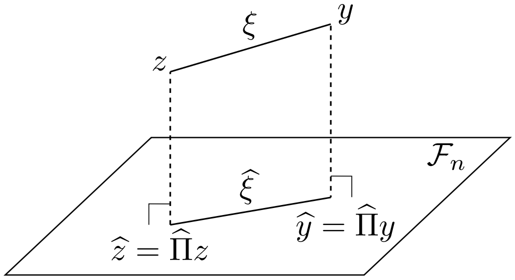

$\xi$
$y$
$z$
$\mathcal{F}_n$
$\widehat{\xi}$
$\widehat{y} = \widehat{\Pi} y$
$\widehat{z} = \widehat{\Pi} z$

Figure 4.11: The vectors used in the proof.

Let $\widehat{\mu}$ be the empirical distribution corresponding to states $x^{(1)}, \dots, x^{(n)}$. Let us define two $N$-dimensional vectors $z = \Big(\big[(\mathcal{T}^{\pi_k})^m V_{k-1}\big](x^{(1)}), \dots, \big[(\mathcal{T}^{\pi_k})^m V_{k-1}\big](x^{(N)})\Big)^\top$ and $y = \big(\widehat{V}_k(x^{(1)}), \dots, \widehat{V}_k(x^{(N)})\big)^\top$ and their orthogonal projections onto the vector space $\mathcal{F}_N$ as $\widehat{z} = \widehat{\Pi} z$ and $\widehat{y} = \widehat{\Pi} y = \big(\widetilde{V}_k(x^{(1)}), \dots, \widetilde{V}_k(x^{(N)})\big)^\top$, where $\widetilde{V}_k$ is the result of linear regression and its truncation (by $V_{\max}$) is $V_k$, i.e., $V_k = \mathbb{T}(\widetilde{V}_k)$ (see Figure 4.11). What we are interested in 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

$$\|z-\widehat{y}\|_{2,\widehat{\mu}} \leq \|\widehat{z}-\widehat{y}\|_{2,\widehat{\mu}} + \|z-\widehat{z}\|_{2,\widehat{\mu}} = \|\widehat{\xi}\|_{2,\widehat{\mu}} + \|z-\widehat{z}\|_{2,\widehat{\mu}}, \tag{4.49}$$

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 $\xi_i = \big[(\mathcal{T}^{\pi_k})^m V_{k-1}\big](x^{(i)}) - \widehat{V}_k(x^{(i)})$. 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

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

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

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

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

$$\|\widehat{z}-\widehat{y}\|_{2,\widehat{\mu}} = \|\widehat{\xi}\|_{2,\widehat{\mu}} \leq 4V_{\max}\sqrt{\frac{2}{N}\log\left(\frac{3(9e^2N)^{d+1}}{\delta'}\right)}. \tag{4.50}$$

From Eqs. 4.49 and 4.50, we have

$$\|(\mathcal{T}^{\pi_k})^m V_{k-1} - \widetilde{V}_k\|_{2,\widehat{\mu}} \leq \|(\mathcal{T}^{\pi_k})^m V_{k-1} - \widehat{\Pi}(\mathcal{T}^{\pi_k})^m V_{k-1}\|_{2,\widehat{\mu}} + 4V_{\max}\sqrt{\frac{2}{N}\log\left(\frac{3(9e^2N)^{d+1}}{\delta'}\right)}. \tag{4.51}$$

Now in order to obtain a random design bound, we first define $f_{\widehat{\alpha}^*} \in \mathcal{F}$ as $f_{\widehat{\alpha}^*}(x^{(i)}) = \big[\widehat{\Pi}(\mathcal{T}^{\pi_k})^m V_{k-1}\big](x^{(i)})$, and then define $f_{\alpha^*} = \Pi(\mathcal{T}^{\pi_k})^m V_{k-1}$ that is the best approximation (w.r.t. $\mu$) of the target function $(\mathcal{T}^{\pi_k})^m V_{k-1}$ 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

$$\begin{aligned}\|f_{\widehat{\alpha}^*} - (\mathcal{T}^{\pi_k})^m V_{k-1}\|_{2,\widehat{\mu}} &\leq \|f_{\alpha^*} - (\mathcal{T}^{\pi_k})^m V_{k-1}\|_{2,\widehat{\mu}} \\ &\leq 2\|f_{\alpha^*} - (\mathcal{T}^{\pi_k})^m V_{k-1}\|_{2,\mu} \\ &\quad + 12\Big(V_{\max} + \|\alpha^*\|_2 \ \sup_x \|\phi(x)\|_2\Big)\sqrt{\frac{2}{N}\log\frac{3}{\delta'}}\end{aligned} \tag{4.52}$$

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 $\|f_{\alpha^*} - (\mathcal{T}^{\pi_k})^m V_{k-1}\|_\infty \leq V_{\max} + \|\alpha^*\|_2 \ \sup_x \|\phi(x)\|_2$. Similarly, we can write the left-hand-side of Eq. 4.51 as

$$\begin{aligned}2\|(\mathcal{T}^{\pi_k})^m V_{k-1} - \widetilde{V}_k\|_{2,\widehat{\mu}} &\geq 2\|(\mathcal{T}^{\pi_k})^m V_{k-1} - \mathbb{T}(\widetilde{V}_k)\|_{2,\widehat{\mu}} \qquad (4.53) \\ &\geq \|(\mathcal{T}^{\pi_k})^m V_{k-1} - \mathbb{T}(\widetilde{V}_k)\|_{2,\mu} - 24V_{\max}\sqrt{\frac{2}{N}\Lambda(N,d,\delta')}\end{aligned}$$

with probability at least $1-\delta'$, where $\Lambda(N,d,\delta') = 2(d+1)\log N + \log\frac{e}{\delta'} + \log\big(9(12e)^{2(d+1)}\big)$. Putting together Eqs. 4.51, 4.52, and 4.53 and using the fact that $\mathbb{T}(\widetilde{V}_k) = V_k$, we obtain

$$\begin{aligned}\|\eta_k\|_{2,\mu} &= \|(\mathcal{T}^{\pi_k})^m V_{k-1} - V_k\|_{2,\mu} \\ &\leq 2\Bigg(2\|(\mathcal{T}^{\pi_k})^m V_{k-1} - f_{\alpha^*}\|_{2,\mu} \\ &\quad + 12\Big(V_{\max} + \|\alpha^*\|_2 \ \sup_x \|\phi(x)\|_2\Big)\sqrt{\frac{2}{N}\log\frac{3}{\delta'}} + 4V_{\max}\sqrt{\frac{2}{N}\log\left(\frac{3(9e^2N)^{d+1}}{\delta'}\right)}\Bigg) \\ &\quad + 24V_{\max}\sqrt{\frac{2}{N}\Lambda(N,d,\delta')}.\end{aligned}$$

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

### 4.8.6 Proof of Lemma 55

**Proof** We prove the following series of inequalities:

$$\begin{aligned}
||\epsilon'_k||_{1,\mu} &\overset{(a)}{\leq} ||\epsilon'_k||_{1,\widehat{\mu}} + e'_3(N,\delta') && \text{w.p. } 1-\delta' \\
&\overset{(b)}{=} \frac{1}{N}\sum_{i=1}^{N}\Big[\max_{a\in\mathcal{A}}\big(\mathcal{T}^a V_{k-1}\big)(x^{(i)}) - \big(\mathcal{T}^{\pi_k} V_{k-1}\big)(x^{(i)})\Big] + e'_3(N,\delta') \\
&\overset{(c)}{\leq} \frac{1}{N}\sum_{i=1}^{N}\Big[\max_{a\in\mathcal{A}}\big(\mathcal{T}^a V_{k-1}\big)(x^{(i)}) - \big(\widehat{\mathcal{T}}^{\pi_k} V_{k-1}\big)(x^{(i)})\Big] + e'_3(N,\delta') + e'_4(N,M,\delta') \\
& && \text{w.p. } 1-2\delta' \\
&\overset{(d)}{=} \frac{1}{N}\sum_{i=1}^{N}\Big[\max_{a\in\mathcal{A}}\big(\mathcal{T}^a V_{k-1}\big)(x^{(i)}) - \max_{a'\in\mathcal{A}}\big(\widehat{\mathcal{T}}^{a'} V_{k-1}\big)(x^{(i)})\Big] + e'_3(N,\delta') + e'_4(N,M,\delta') \\
&\overset{(e)}{\leq} \frac{1}{N}\sum_{i=1}^{N}\Big\{\max_{a\in\mathcal{A}}\Big[\big(\mathcal{T}^a V_{k-1}\big)(x^{(i)}) - \big(\widehat{\mathcal{T}}^{a} V_{k-1}\big)(x^{(i)})\Big]\Big\} + e'_3(N,\delta') + e'_4(N,M,\delta') \\
&\overset{(f)}{\leq} e'_3(N,\delta') + 2e'_4(N,M,\delta') && \text{w.p. } 1-3\delta'
\end{aligned}$$

**(a)** This step is the result of the following lemma.

**Lemma 60** *Let $\Pi$ be the policy space of the policies obtained by Eq. 4.3 from the truncation (by $V_{\max}$) of the function space $\mathcal{F}$, with finite VC-dimension $h = VC(\Pi) < \infty$. Let $N > 0$ be the number of states in the rollout set $\mathcal{D}_k$, drawn i.i.d. from the state distribution $\mu$. Then, we have*

$$\mathbb{P}_{\mathcal{D}_k}\Big[\sup_{\pi\in\Pi} \big|||\epsilon'_k(\pi)||_{1,\widehat{\mu}} - ||\epsilon'_k(\pi)||_{1,\mu}\big| > e'_3(N,\delta)\Big] \leq \delta \ ,$$

*with $e'_3(N,\delta) = 16V_{\max}\sqrt{\frac{2}{N}(h\log\frac{eN}{h} + \log\frac{8}{\delta})}$.*

**Proof** The proof is similar to the proof of Lemma 1 in Lazaric et al. [2010a].

■

**(b)** This is from the definition of $||\epsilon'_k||_{1,\widehat{\mu}}$.

**(c)** This step is the result of bounding

$$\sup_{\pi\in\Pi}\left[\frac{1}{N}\sum_{i=1}^{N}\big(\widehat{\mathcal{T}}^{\pi}V_{k-1}\big)(s^{(i)}) - \frac{1}{N}\sum_{i=1}^{N}\big(\mathcal{T}^{\pi}V_{k-1}\big)(s^{(i)})\right]$$

by $e'_4(N,M,\delta)$. The supremum over all the policies in the policy space $\Pi$ is due to the fact that $\pi_k$ is a random object whose randomness comes from all the randomly generated samples at the $k$'th iteration (i.e., the states in the rollout set and all the generated rollouts). We bound this term using the following lemma.

**Lemma 61** *Let $\Pi$ be the policy space of the policies obtained by Eq. 4.3 from the truncation (by $V_{\max}$) of the function space $\mathcal{F}$, with finite VC-dimension $h = VC(\Pi) < \infty$. Let $\{x^{(i)}\}_{i=1}^{N}$ be $N$ states sampled i.i.d. from the distribution $\mu$. For each sampled state $x^{(i)}$, we take the action suggested by policy $\pi$, $M$ times, and observe the next states $\{x^{(i,j)}\}_{j=1}^{M}$. Then, we have*

$$\mathbb{P}\left[\sup_{\pi\in\Pi}\left|\frac{1}{N}\sum_{i=1}^{N}\frac{1}{M}\sum_{j=1}^{M}\big[r\big(x^{(i)},\pi(x^{(i)}\big)+\gamma V_{k-1}\big(x^{(i,j)}\big)\big] - \frac{1}{N}\sum_{i=1}^{N}\big(\mathcal{T}^{\pi}V_{k-1}\big)(x^{(i)})\right| > e'_4(N,M,\delta)\right] \leq \delta\ ,$$

*with* $e'_4(N,M,\delta) = 8V_{\max}\sqrt{\frac{2}{MN}\big(h\log\frac{eMN}{h}+\log\frac{8}{\delta}\big)}$.

**Proof** The proof is similar to the proof of Lemma 4 in Lazaric et al. [2010b].
■

**(d)** This step is from the definition of $\pi_k$ in the AMPI-V algorithm (Eq. 4.3).

**(e)** This step is algebra, replacing two maximums with one.

**(f)** This step is similar to Step (c).

The proof follows by setting $\delta = 3\delta'$. ■

### 4.8.7 Proof of Lemma 56

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

**Lemma 62** *Let $\Pi$ 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 $\mu$. Then we have*

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

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

**Proof** This is a restatement of Lemma 1 in Lazaric et al. [2010a]. ■

**Lemma 63** *Let $\Pi$ be a policy space with finite VC-dimension $h = VC(\Pi) < \infty$ and $s^{(1)}, \ldots, s^{(N')}$ be an arbitrary sequence of states. Assume that at each state, we simulate $M$ independent rollouts. We have*

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

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

**Proof** The proof is similar to the one for Lemma 62. ■

**Proof (Lemma 56)** Let $a^*(\cdot) \in \arg\max_{a \in \mathcal{A}} Q_{k-1}(\cdot, a)$ be a 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:

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

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

**(a)** This follows from Lemma 62.

**(b)** Here we introduce the estimated action-value function $\widehat{Q}_{k-1}$ by bounding

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

using Lemma 63.

**(c)** From the definition of $\pi_k$ in CBMPI, we have

$$\pi_k=\operatorname*{arg\,min}_{\pi\in\Pi}\widehat{\mathcal{L}}^{\Pi}_{k-1}(\widehat{\mu};\pi)=\operatorname*{arg\,max}_{\pi\in\Pi}\frac{1}{N'}\sum_{i=1}^{N'}\widehat{Q}_{k-1}\big(x^{(i)},\pi(x^{(i)})\big),$$

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

$$\widetilde{\pi}=\operatorname*{arg\,min}_{\pi\in\Pi}\int_{\mathcal{X}}\left(\max_{a\in\mathcal{A}}Q_{k-1}(x,a)-Q_{k-1}\big(x,\pi(x)\big)\right)\mu(ds).$$

■

# Chapter 5

# Analysis of Regularized Approximate Dynamic Programming Algorithms [MGH4, MGH9, MGH12, MGH15, MGH19, MGH20]

Another area that I have worked on is the application of reinforcement leaning (RL) to problems with high-dimensional state and/or action spaces. Since this line of work is not directly related to the topic of this thesis, I only summarize it in this chapter. With the explosive growth and ever increasing complexity of data, developing theory and algorithms for learning with big and high-dimensional data has become an important challenge in statistical machine learning and control. There have been recent advances in handling high-dimensional data in the field of statistical machine learning, namely the new developments in compressive sensing and regularization with $\ell_1$ and $\ell_2$ norms. Although many learning techniques with promising performance have been developed, there still remain significant gaps in the theoretical foundations. Moreover, most of the research has been focused on supervised learning problems (regression and classification), and only a few preliminary results have been reported in RL and control. However, the recent results, especially those in regression, can help us in developing new theory and algorithms for RL with high-dimensional state and action spaces. The main objective here is to devise and analyze RL algorithms whose sample and computational complexities do not grow rapidly with the dimension of the state space. I have tackled this problem from two different angles that will be briefly discussed in the next two sections.

## 5.1 Exploiting the Regularities of the Problem

In order to solve RL in high dimensions, we should exploit all the regularities of the problem in hand. *Smoothness* is the most common regularity. We have done theoretical and algorithmic work on controlling the smoothness of the value function approximation in RL by adding $\ell_2$-regularization to a number of widely-used approximate dynamic programming (ADP) and RL algorithms. These algorithms include both approximate policy iteration, least-square temporal-difference learning (LSTD) [Bradtke and Barto, 1996] and its control version least-squares policy iteration (LSPI) [Lagoudakis and Parr, 2003a] and modified Bellman residual minimization (BRM) [Antos et al., 2008], and approximate value iteration, fitted Q-iteration [Ernst et al., 2005], methods. Here is a summary of my work on $\ell_2$-regularized ADP and RL algorithms:

- **Regularized Policy Iteration [Farahmand et al., 2008, 2013a]:** We studied two $\ell_2$-regularization-based approximate policy iteration algorithms, namely REG-LSPI and REG-BRM, to solve RL in discounted Markov Decision Processes (MDPs) with large state and finite action spaces. The core of these algorithms are the $\ell_2$-regularized extensions of LSTD and modified BRM, which are used in the algorithms' policy evaluation steps. Regularization provides a convenient way to control the complexity of the function space to which the estimated value function belongs and as a result enables us to work with rich function spaces. We derived efficient implementations of our methods when the function space is a reproducing kernel Hilbert space. We analyzed the statistical properties of REG-LSPI and provide an upper bound on the policy evaluation error and the performance loss of the policy returned by this method. Our bound shows the dependence of the loss on the number of samples, the capacity of the function space, and some intrinsic properties of the underlying Markov Decision Process. The dependence of the policy evaluation bound on the number of samples is minimax optimal.

- **Regularized Fitted Q-Iteration [Farahmand et al., 2009]:** We considered the $\ell_2$-regularized fitted Q-iteration algorithm and provided generalization bounds that account for small sample sizes, and used a realistic visual-servoing problem to illustrate the benefits of using this regularization procedure.

*Sparsity* is another form of regularity that clearly plays a central role in the emerging theory of learning in high dimensions. We have worked on using $\ell_1$-regularization in ADP and RL, which may also serve as a method for feature selection in value function approximation. We have worked on adding $\ell_1$-penalty to the LSTD algorithm and on integrating LSTD with the Dantzig Selector. Here is a brief description of these two works:

- **Finite- Sample Analysis of Lasso-TD [Ghavamzadeh et al., 2011]:** We analyzed the performance of Lasso-TD, a modification of LSTD in which the projection operator is defined as a *Lasso* problem [Hastie et al., 2001]. We first showed that Lasso-TD is guaranteed to have a unique fixed point and its algorithmic implementation coincides with the recently presented LARS-TD [Kolter and Ng, 2009] and LC-TD [Johns et al., 2010] methods. We then derived two bounds on the prediction error of Lasso-TD in the Markov design setting, i.e., when the performance is evaluated on the same states used by the method. The first bound makes no assumption, but has a slow rate with respect to the number of samples. The second bound is under an assumption on the empirical Gram matrix, called the compatibility condition, but has an improved rate and directly relates the prediction error to the sparsity of the value function in the feature space at hand.

- **Temporal-Difference Learning with a Dantzig Selector [Geist et al., 2012]:** Since LSTD is not a simple regression algorithm, but rather solves a fixed-point problem, its integration with $\ell_1$-regularization is not straightforward and might come with some drawbacks (e.g., the P-matrix assumption for Lasso-TD; see Kolter and Ng 2009, Johns et al. 2010). In this work, we introduced a novel algorithm obtained by integrating LSTD with the Dantzig selector. We investigated the performance of the proposed algorithm and its relationship with the existing regularized approaches, and showed how it addresses some of their drawbacks.

## 5.2 Random Projections

We have also looked into recent directions popularized in compressive sensing [Candès and Wakin, 2008] concerning the preservation of properties, such as norm or inner-product, of high dimensional objects when projected on possibly much lower dimensional random subspaces (the so-called Johnson-Lindenstrauss Lemma; see e.g., Achlioptas 2003). Those techniques (based on concentration of measure phenomena) turn high dimensionality of certain problems to a blessing rather than a curse; they have started to appear in the statistical learning community [Ailon and Chazelle, 2006, Rahimi and Recht, 2008, Zhou et al., 2008], but have not been used much in RL. On this topic, we studied the popular LSTD algorithm when a space of low dimension is generated with a random projection from the high-dimensional space, and derived performance bounds for the resulting algorithm:

- **LSTD with Random Projections [Ghavamzadeh et al., 2010]:** We considered the problem of RL in high-dimensional spaces when the number of features is bigger than the number of samples. In particular, we studied the LSTD learning algorithm when a space of low dimension is generated with a random projection from a high-dimensional space. We provided a thorough theoretical analysis of the LSTD with random projections and derived performance bounds for the resulting algorithm. We also showed how the error of LSTD with random projections is propagated through the iterations of a policy iteration algorithm and provided a performance bound for the resulting LSPI algorithm.

# Bibliography

Y. Abbasi-Yadkori, D. Pal, and Cs. Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems 25*, 2011.

D. Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 4:671–687, 2003.

N. Ailon and B. Chazelle. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *Proceedings of the Thirty Eighth Annual ACM Symposium on Theory of Computing*, pages 557–563, 2006.

A. Antos, R. Munos, and Cs. Szepesvári. Fitted Q-iteration in continuous action-space MDPs. In *Proceedings of the Advances in Neural Information Processing Systems 19*, pages 9–16, 2007.

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

J.-Y. Audibert, S. Bubeck, and R. Munos. Best Arm Identification in Multi-Armed Bandits. In *Proceedings of the Twenty-Third Annual Conference on Learning Theory*, pages 41–53, 2010.

B. Ávila Pires, M. Ghavamzadeh, and Cs. Szepesvári. Cost-sensitive multiclass classification risk bounds. In *Proceedings of the Thirtieth International Conference on Machine Learning*, pages 1391–1399, 2013.

B. Ávila Pires and Cs. Szepesvári. Statistical linear estimation with penalized estimators: an application to reinforcement learning. In *Proceedings of the Twenty-Ninth International Conference on Machine Learning*, pages 1535–1542, 2012.

J. Bagnell, S. Kakade, A. Ng, and J. Schneider. Policy search by dynamic programming. In *Proceedings of Advances in Neural Information Processing Systems 16*, 2003.

L. Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 30–37, 1995.

A. Barto, R. Sutton, and C. Anderson. Neuron-like elements that can solve difficult learning control problems. *IEEE Transaction on Systems, Man and Cybernetics*, 13:835–846, 1983.

S. Ben-David, N. Cesa-Bianchi, D. Haussler, and P. Long. Characterizations of learnability for classes of {0...n}-valued functions. *Journal of Computer and System Sciences*, 50:74–86, 1995.

D. Bertsekas. *Dynamic Programming and Optimal Control, volume II.* Athena Scientific, 2007.

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

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

A. Beygelzimer, V. Dani, T. Hayes, J. Langford, and B. Zadrozny. Error limiting reductions between classification tasks. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, pages 49–56, 2005.

A. Beygelzimer, J. Langford, and P. Ravikumar. Error-correcting tournaments. *CoRR*, 2009.

J. Boyan. Least-squares temporal difference learning. *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 49–56, 1999.

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

S. Bubeck, R. Munos, and G. Stoltz. Pure exploration in multi-armed bandit problems. In *Proceedings of the Twentieth International Conference on Algorithmic Learning Theory*, pages 23–37, 2009.

H. Burgiel. How to Lose at Tetris. *Mathematical Gazette*, 81:194–200, 1997.

R. Busa-Fekete and B. Kégl. Fast boosting using adversarial bandits. In *Proceedings of the Twenty-Seventh International Conference on Machine Learning*, pages 49–56, 2010.

P. Canbolat and U. Rothblum. (approximate) iterated successive approximations algorithm for sequential decision processes. *Annals of Operations Research*, pages 1–12, 2012.

E. Candès and M. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21–30, 2008.

C. Chang and C. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27:1–27:27, 2011.

V. de la Peña and G. Pang. Exponential inequalities for self-normlized processes with applications. *Electronic Communications in Probability*, 14: 372–381, 2009.

V. de la Peña, M. Klass, and T. Leung Lai. Pseudo-maximization and self-normalized processes. *Propability Surveys*, 4:172–192, 2007.

S. Delattre and S. Gaïffas. Nonparametric regression with martingale increment errors. *Stochastic Processes and their Applications*, 121(12):2899–2924, 2011.

E. Demaine, S. Hohenberger, and D. Liben-Nowell. Tetris is hard, even to approximate. In *Proceedings of the Ninth International Computing and Combinatorics Conference*, pages 351–363, 2003.

L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer-Verlag, 1996.

C. Dimitrakakis and M. Lagoudakis. Algorithms and bounds for sampling-based approximate policy iteration. In *Recent Advances in Reinforcement Learning (EWRL)*. Springer, 2008a.

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

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

C. Fahey. Tetris AI, Computer plays Tetris, 2003.

A. Farahmand, M. Ghavamzadeh, Cs. Szepesvári, and S. Mannor. Regularized policy iteration. In *Advances in Neural Information Processing Systems 21*, pages 441–448, 2008.

A. Farahmand, R. Munos, and Cs. Szepesvári. Error propagation for approximate policy and value iteration. In *Advances in Neural Information Processing Systems*, pages 568–576, 2010.

A. Farahmand, M. Ghavamzadeh, Cs. Szepesvári, and S. Mannor. Regularized policy iteration. *Submitted to the Journal of Machine Learning Research*, 2013a.

A. Farahmand, D. Precup, M. Ghavamzadeh, and A. Barreto. On classification-based approximate policy iteration. *Submitted to the IEEE Transactions on Automatic Control*, 2013b.

A. M. Farahmand, M. Ghavamzadeh, Cs. Szepesvári, and S. Mannor. Regularized fitted Q-iteration for planning in continuous-space Markovian decision problems. In *Proceedings of the American Control Conference*, pages 725–730, 2009.

V. Farias and B. Van Roy. *Tetris: A study of randomized constraint sampling.* Springer-Verlag, 2006.

A. Fern, S. Yoon, and R. Givan. Approximate policy iteration with a policy language bias. In *Proceedings of Advances in Neural Information Processing Systems 16*, 2004.

A. Fern, S. Yoon, and R. Givan. Approximate policy iteration with a policy language bias: Solving relational Markov decision processes. *Journal of Artificial Intelligence Research*, 25:85–118, 2006.

T. Furmston and D. Barber. A unifying perspective of parametric policy search methods for Markov decision processes. In *Proceedings of the Advances in Neural Information Processing Systems 24*, pages 2726–2734, 2012.

V. Gabillon, A. Lazaric, and M. Ghavamzadeh. Rollout allocation strategies for classification-based policy iteration. In *ICML-2010 Workshop on Reinforcement Learning and Search in Very Large Spaces*, 2010.

V. Gabillon, M. Ghavamzadeh, A. Lazaric, and S. Bubeck. Multi-bandit best arm identification. In *Proceedings of Advances in Neural Information Processing Systems 24*, pages 2222–2230, 2011a.

V. Gabillon, A. Lazaric, M. Ghavamzadeh, and B. Scherrer. Classification-based policy iteration with a critic. In *Proceedings of the Twenty-Eighth International Conference on Machine Learning*, pages 1049–1056, 2011b.

V. Gabillon, M. Ghavamzadeh, and A. Lazaric. Best arm identification: A unified approach to fixed budget and fixed confidence. In *Proceedings of Advances in Neural Information Processing Systems 25*, pages 3221–3229, 2012.

V. Gabillon, M. Ghavamzadeh, and B. Scherrer. Approximate dynamic programming finally performs well in the game of tetris. In *Proceedings of Advances in Neural Information Processing Systems 26*, 2013.

M. Geist, B. Scherrer, A. Lazaric, and M. Ghavamzadeh. A Dantzig selector approach to temporal difference learning. In *Proceedings of the Twenty-Ninth International Conference on Machine Learning*, pages 1399–1406, 2012.

M. Ghavamzadeh and A. Lazaric. Conservative and greedy approaches to classification-based policy iteration. In *Proceedings of the Twenty-Sixth Conference on Artificial Intelligence*, pages 914–920, 2012.

M. Ghavamzadeh, A. Lazaric, O. Maillard, and R. Munos. Lstd with random projections. In *Advances in Neural Information Processing Systems 24*, pages 721–729, 2010.

M. Ghavamzadeh, A. Lazaric, R. Munos, and M. Hoffman. Finite-sample analysis of lasso-td. In *Proceedings of the Twenty-Eighth International Conference on Machine Learning*, pages 1177–1184, 2011.

G. Gordon. Stable function approximation in dynamic programming. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 261–268, 1995.

L. Györfi, M. Kohler, A. Krzyżak, and H. Walk. *A distribution-free theory of nonparametric regression.* Springer-Verlag, 2002.

N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9:159–195, 2001.

T. Hansen, P. Miltersen, and U. Zwick. Strategy iteration is strongly polynomial for two-player turn-based stochastic games with a constant discount factor. *Journal of ACM*, 60(1):1–16, 2013.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, 2001.

R. A. Howard. *Dynamic Programming and Markov Processes.* The MIT Press, 1960.

D. Hsu, S. Kakade, and T. Zhang. Random design analysis of ridge regression. In *Proceedings of the Twenty-Fifth Conference on Learning Theory*, 2012.

J. Johns, C. Painter-Wakefield, and R. Parr. Linear complementarity for regularized policy evaluation and improvement. In *Proceedings of Advances in Neural Information Processing Systems 23*, pages 1009–1017, 2010.

S. Kakade. A natural policy gradient. In *Proceedings of the Advances in Neural Information Processing Systems 14*, pages 1531–1538, 2002.

S. Kakade. *On the Sample Complexity of Reinforcement Learning.* PhD thesis, Gatsby Computational Neuroscience Unit., University College London, 2003.

S. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 267–274, 2002.

M. Kearns, Y. Mansour, and A. Ng. Approximate planning in large pomdps via reusable trajectories. In *Proceedings of the Advances in Neural Information Processing Systems 12*, pages 1001–1007, 2000.

Z. Kolter and A. Ng. Regularization and feature selection in least-squares temporal difference learning. In *Proceedings of the Twenty-Sixth International Conference on Machine Learning*, pages 521–528, 2009.

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

M. Lagoudakis and R. Parr. Reinforcement learning as classification: Leveraging modern classifiers. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 424–431, 2003b.

J. Langford and B. Zadrozny. Relating reinforcement learning performance to classification performance. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, pages 473–480, 2005.

A. Lazaric, M. Ghavamzadeh, and R. Munos. Analysis of a classification-based policy iteration algorithm. In *Proceedings of the Twenty-Seventh International Conference on Machine Learning*, pages 607–614, 2010a.

A. Lazaric, M. Ghavamzadeh, and R. Munos. Analysis of a classification-based policy iteration algorithm. Technical Report inria-00482065, INRIA, 2010b.

A. Lazaric, M. Ghavamzadeh, and R. Munos. Finite-sample analysis of LSTD. In *Proceedings of the Twenty-Seventh International Conference on Machine Learning*, pages 615–622, 2010c.

Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99(465): 67–81, 2004.

B. Lesner and B. Scherrer. Tight performance bounds for approximate modified policy iteration with non-stationary policies. *CoRR*, abs/1304.5610, 2013.

L. Li, V. Bulitko, and R. Greiner. Focus of attention in reinforcement learning. *Journal of Universal Computer Science*, 13(9):1246–1269, 2007.

A. Lozano and N. Abe. Multi-class cost-sensitive boosting with p-norm loss functions. In *Proceeding of the Fourteenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 506–514, 2008.

O. Maron and A. Moore. Hoeffding races: Accelerating model selection search for classification and function approximation. In *Proceedings of Advances in Neural Information Processing Systems 7*, 1993.

R. Meir. Nonparametric time series prediction through adaptive model selection. *Machine Learning*, 39(1):5–34, 2000.

S. Meyn and Tweedie R. *Markov chains and stochastic stability.* Springer-Verlag, 1993.

R. Munos. Error bounds for approximate policy iteration. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 560–567, 2003.

R. Munos. Performance bounds in $l_p$-norm for approximate value iteration. *SIAM Journal of Control and Optimization*, 46(2):541–561, 2007.

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

D. Pollard. *Convergence of Stochastic Processes*. Springer-Verlag, 1984.

D. Precup, R. Sutton, and S. Singh. Eligibility traces for off-policy policy evaluation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 759–766, 2000.

D. Precup, R. Sutton, and S. Dasgupta. Off-policy temporal difference learning with function approximation. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 417–424, 2001.

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

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Proceedings of Advances in Neural Information Processing Systems 20*, pages 1177–1184, 2008.

R. Rubinstein and D. Kroese. *The cross-entropy method: A unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*. Springer-Verlag, 2004.

B. Scherrer. Should one compute the temporal difference fix point or minimize the Bellman residual? the unified oblique projection view. In *Proceedings of the Twenty-Seventh International Conference on Machine Learning*, pages 959–966, 2010.

B. Scherrer. Improved and generalized upper bounds on the complexity of policy iteration. In *Proceedings of Advances in Neural Information Processing Systems 26*, 2013a.

B. Scherrer. Performance bounds for $\lambda$-policy iteration and application to the game of tetris. *Journal of Machine Learning Research*, 14:1175–1221, 2013b.

B. Scherrer and C. Thiéry. Performance bound for approximate optimistic policy iteration. Technical report, INRIA, 2010.

B. Scherrer, M. Ghavamzadeh, V. Gabillon, and M. Geist. Approximate modified policy iteration. In *Proceedings of the Twenty-Ninth International Conference on Machine Learning*, pages 1207–1214, 2012.

B. Scherrer, M. Ghavamzadeh, V. Gabillon, B. Lesner, and M. Geist. Approximate modified policy iteration. *Submitted to the Journal of Machine Learning Research*, 2013.

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

S. Singh and R. Yee. An Upper Bound on the Loss from Approximate Optimal-Value Functions. *Machine Learning*, 16(3):227–233, 1994.

R. Sutton. *Temporal credit assignment in reinforcement learning.* PhD thesis, University of Massachusetts Amherst, 1984.

R. Sutton and A. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

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

I. Szita and A. LHorincz. Learning Tetris Using the Noisy Cross-Entropy Method. *Neural Computation*, 18(12):2936–2941, 2006.

M. Talagrand. *The Generic Chaining: Upper and Lower Bounds of Stochastic Processes*. Springer-Verlag, 2005.

C. Thiery and B. Scherrer. Building controllers for tetris. *International Computer Games Association Journal*, 32:3–11, 2009a.

C. Thiery and B. Scherrer. Improvements on Learning Tetris with Cross Entropy. *International Computer Games Association Journal*, 32, 2009b.

C. Thiery and B. Scherrer. Least-squares $\lambda$-policy iteration: Bias-variance trade-off in control problems. In *Proceedings of the Twenty-Seventh International Conference on Machine Learning*, pages 1071–1078, 2010a.

C. Thiery and B. Scherrer. MDPTetris features documentation, 2010b.

J. Tsitsiklis and B Van Roy. Feature-based methods for large scale dynamic programming. *Machine Learning*, 22:59–94, 1996.

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

H. Tu and H. Lin. One-sided support vector regression for multiclass cost-sensitive classification. In *Proceedings of the Twenty-Seventh International Conference on Machine Learning*, pages 49–56, 2010.

Y. Ye. The simplex and policy iteration methods are strongly polynomial for the Markov decision problem with a fixed discount rate. *Mathematics of Operations Research*, 36(4):593–603, 2011.

B. Yu. Rates of convergence for empirical processes of stationary mixing sequences. *The Annals of Probability*, 22(1):94–116, 1994.

H. Yu. Convergence of least squares temporal difference methods under general conditions. In *Proceedings of the Twenty-Seventh International Conference on Machine Learning*, pages 1207–1214, 2010.

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

B. Zadrozny, J. Langford, and N. Abe. Cost-sensitive learning by cost-proportionate example weighting. In *Proceedings of the Third IEEE International Conference on Data Mining*, page 435, 2003.

S. Zhou, J. Lafferty, and L. Wasserman. Compressed regression. In *Proceedings of Advances in Neural Information Processing Systems 20*, pages 1713–1720, 2008.