Provable Q-Iteration with L infinity Guarantees and Function Approximation

Ming Yu [*] **Zhuoran Yang** [†] **Mengdi Wang** [‡] **Zhaoran Wang** [§]

Abstract

Motivated by the large state and/or action space in recent applications in reinforcement learning, much progress has been made on approximate dynamic programming which aims to approximate the policy evaluation and policy greedy step and hence improve the efficiency of the learning procedure. Most of the existing analysis relies on a constant concentrability coefficient assumption, which could easily be violated with a large state and/or action space. In this paper, we propose a novel theoretical framework with different analysis based on ℓ_∞-norm. Our analysis is based on RKHS nonparametric regression with dependent noise. We prove that our proposed algorithm converges linearly to the optimal solution in supremum norm up to a statistical error. Moreover, we show how our framework can be applied to two-player zero-sum games and regularized MDP problem.

1 Introduction

Reinforcement learning (RL) [43] with function approximation, especially with approximation by deep neural networks, has huge empirical success [24, 40, 41]. As an efficient and stable method, batch reinforcement learning (batch RL) provides an elegant theoretical framework for understanding the statistical aspects of RL algorithms with value function approximation. See [20] for an overview. In batch RL, we collect a batch of data and use this fixed dataset to learn an optimal policy. One of the most important algorithms is fitted Q-iteration (FQI) algorithm [11, 33], where we obtain a sequence of value functions by regression. Specifically, in each iteration, we collect n samples and fit a least square regression to update the value function.

Although this framework successfully provides theoretical analysis for various algorithms, it often requires assumptions on the distribution shift. Specifically, define the concentrability coefficient as

$$C_p(j) = \sup_{\pi_1,\ldots,\pi_j} \left\| \frac{d(P_{\pi_1} P_{\pi_2} \cdots P_{\pi_j} \nu)}{d\tau} \right\|_{p,\tau}, \tag{1}$$

where the τ-weighted ℓ_p norm is defined as $\|f\|_{p,\tau} = \big[\int |f(x)|^p \tau(dx)\big]^{1/p}$. Most of the existing works assume that this concentrability coefficient is an absolute constant. Here τ is the data generating distribution, and ν is some admissible distribution, which means that this distribution can be generated in an MDP by following some policy for several time stages. Intuitively, this concentrability coefficient assumption requires that any admissible distribution is not that far away from the distribution the data is generated (i.e., the distribution shift is mild). However, when the state and/or action space is large, it is hard to establish an upper bound on these coefficients without extra information of the underlying MDP. In [26], the authors provide several examples where this concentrability coefficient is well controlled. However, these are mostly carefully constructed toy problems in a tabular setting

[*]Booth School of Business, The University of Chicago, Chicago, IL.

[†]Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ.

[‡]Center for Statistics and Machine Learning, Princeton University, Princeton, NJ.

[§]Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL.

with some underlying assumptions on the MDP. In contrast, in [9], the authors show that in general, the worst case concentrability coefficient will scale with $|\mathcal{S} \times \mathcal{A}|$. This makes it intolerable as the size of state space is enormous, and it weakens the value of the analysis based on the concentrability coefficient assumption. Moreover, they show that no algorithm can achieve polynomial sample complexity without any assumptions on MDP; hence the concentrability coefficient on worst case MDP can be exponential in horizon (Theorem 4 in [9]).

As a different approach, in this paper we establish a supremum norm convergence result for these algorithms when the value functions are approximated by an RKHS. Specifically, we propose to solve the MDP problem using approximate modified policy iteration (AMPI, [37]) algorithm where the approximation policy evaluation is obtained by a nonparametric RKHS regression in Hölder space. We prove the supremum norm convergence of our algorithm where the error rate consists of statistical error and optimization error. The optimization error converges to zero in a linear rate, and the statistical error follows from the analysis on RKHS regression [52]. However, different from [52], in our case the error term is dependent on the state and action space, imposing more difficulties on quantifying this approximation error. Moreover, we show the complexity of the sampling procedure of our algorithm explicitly. Finally, we show that our proposed framework can be naturally applied to two-player zero-sum games and regularized MDP problem with both entropy and KL-regularization.

Our main contributions are as follow. Firstly, we establish a supremum norm statistical rate of convergence for batch RL with a different analysis based on ℓ_∞-norm. This contributes to the theoretical analysis of reinforcement learning. Secondly, our theoretical framework is general and can be readily modified to handle a two-player zero-sum game, regularized MDP with continuous action space, etc. Besides, our theory shed new light on the design of batch RL algorithms. It is preferred to construct value estimators with ℓ_∞-norm statistical rates. In statistics literature, various techniques such as de-biasing [46, 18, 56, 29] are proposed for controlling ℓ_∞-norm errors, which might be used in batch RL.

2 Background

In this section, we introduce the background of reinforcement learning and RKHS.

2.1 Reinforcement Learning

A Markov decision process (MDP) can be represented by $(\mathcal{S}, \mathcal{A}, P, \gamma, r)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ is the transition probability distribution where we denote $\mathcal{P}(\mathcal{X})$ as the set of probability distributions over $\mathcal{X}$ for any $\mathcal{X}$, $\gamma \in (0, 1)$ is the discounted factor, and $r \colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function. Throughout this paper we assume that the state space is given by $\mathcal{S} = [0, 1]^d$ with dimension d, the action space $\mathcal{A}$ is finite and the reward function is bounded: $|r(s, a)| \le R_{\max}$. A policy is a mapping $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ that specifies the action that an agent will take at state s. Given a fixed policy π, the state- and action-value functions of π are defined as

$$V_\pi(s) = \mathbb{E}_\pi\Bigg[\sum_{t\ge 0} \gamma^t r(s_t, a_t) \,\Bigg|\, s_0 = s\Bigg], \quad Q_\pi(s, a) = \mathbb{E}_\pi\Bigg[\sum_{t\ge 0} \gamma^t r(s_t, a_t) \,\Bigg|\, s_0 = s, a_0 = a\Bigg],$$

where $s_{t+1} \sim P(\cdot \mid s_t, a_t)$ and $a_t \sim \pi(\cdot \mid s_t)$. The optimal state- and action-value function is defined as $V^*(s) = \sup_\pi V_\pi(s)$ and $Q^*(s, a) = \sup_\pi Q_\pi(s, a)$. Define the Bellman operator with respect to policy π as

$$T_\pi Q(s, a) = \mathbb{E}\big[r(s, a) + \gamma \cdot Q(s', a') \,\big|\, s' \sim P(\cdot|s, a), a' \sim \pi(\cdot|s')\big]. \tag{2}$$

The Bellman optimality operator can be similarly defined as

$$TQ(s, a) = \mathbb{E}\Big[r(s, a) + \gamma \cdot \max_{a'\in\mathcal{A}} Q(s', a') \,\Big|\, s' \sim P(\cdot|s, a)\Big].$$

It is well known that both T_π and T are contractions under supremum norm, and Q_π is the unique fixed point of T_π.

Policy iteration, value iteration, and modified policy iteration The goal of an MDP is to maximize the cumulative discounted expected reward $R = \mathbb{E}\big[\sum_{t\le 0} \gamma^t \cdot t(s_t, a_t)\big]$. Common approaches to solve the MDP problem include policy iteration (PI) and value iteration (VI). A unified generalization of these two approaches is called modified policy iteration (MPI, [32]) with the algorithm summarized as below.

$$\pi_k = \text{Greedy}(Q_k), \qquad Q_{k+1} = (T_{\pi_k})^m Q_k. \tag{3}$$

The first step in (3) is the policy greedy step, and the second step is the policy evaluation step. Both policy iteration and value iteration are special cases of (3). When $m = 1$ it corresponds to value iteration and $m = \infty$ corresponds to policy iteration.

2.2 Reproducing Kernel Hilbert Space

A reproducing kernel Hilbert space (RKHS) is a Hilbert space of functions on some nonempty set $\mathcal{X}$ with evaluation function given by $L_x : f \mapsto f(x)$. We say that $\mathcal{H}$ is an RKHS if for any $x \in \mathcal{X}$, the evaluation function L_x is a bounded operator, i.e. there exists some constant M_x depending on x such that $|L_x(f)| := |f(x)| \leq M_x \|f\|_{\mathcal{H}}$ for any $f \in \mathcal{H}$. Here $\|f\|_{\mathcal{H}} = \sqrt{\langle f, f\rangle_{\mathcal{H}}}$ is the norm induced by the inner product. Although this definition is straightforward, a more practical definition comes from the Riesz representation theorem which says that for all $x \in \mathcal{X}$ there exists a unique element $K_x \in \mathcal{H}$ such that $f(x) = L_x(f) = \langle f, K_x\rangle_{\mathcal{H}}$ for all $f \in \mathcal{H}$. For this representer function $K_x \in \mathcal{H}$, we apply this property again and we obtain that for any $y \in \mathcal{X}$, there exists a representer function $K_y \in \mathcal{H}$ such that $K_x(y) = \langle K_x, K_y\rangle_{\mathcal{H}}$. This motivates the definition of reproducing kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with $K(x, y) = \langle K_x, K_y\rangle_{\mathcal{H}}$. From the definition it is straightforward to verify that this reproducing kernel is symmetric: $K(x, y) = K(y, x)$, and positive definite: $\sum_{i,j=1}^{n} c_i c_j K(x_i, x_j) \geq 0$. On the other hand, according to the Moore-Aronszajn theorem, with a symmetric and positive definite kernel K, we can define a unique RKHS on $\mathcal{X}$ with K as the reproducing kernel.

We then consider the square integrable function space $L^2(\mathcal{X}) = \{f : \int_{\mathcal{X}} |f|^2 d\tau < \infty\}$ where $\mathcal{X}$ is a compact space and τ is a strictly positive measure on $\mathcal{X}$. The inner product on $L^2(\mathcal{X})$ is given by $\langle f, g\rangle_{L^2(\mathcal{X})} = \int_{\mathcal{X}} f(x)g(x)d\tau(x)$. According to the Mercer's theorem, we can represent the kernel K in terms of the eigenvalues and eigenfunctions as $K(x, y) = \sum_{j=1}^{\infty} \eta_j\, \varphi_j(x)\, \varphi_j(y)$, where $\{\eta_j\}$ are at most countable decreasing and nonnegative eigenvalues and $\{\varphi_j\}$ are orthonormal basis of $L_2(\mathcal{X})$. With this eigen-representation, the RKHS defined by the kernel K is given by $\mathcal{H} = \left\{f \in L_2(\mathcal{X}) \;\middle|\; \sum_{i=1}^{\infty} \eta_i^{-1} \cdot \langle f, \varphi_i\rangle_{L_2(\mathcal{X})}^2 < \infty\right\}$, and the inner product of $\mathcal{H}$ is given by $\langle f, g\rangle_{\mathcal{H}} = \sum_{i=1}^{\infty} \eta_i^{-1} \cdot \langle f, \varphi_i\rangle_{L_2(\mathcal{X})} \langle g, \varphi_i\rangle_{L_2(\mathcal{X})}$.

3 Approximate Modified Policy Iteration

In most of the recent applications in MDP, the state and/or action space are usually large, and therefore both policy iteration, value iteration and modified policy iteration algorithms are slow since they need to loop over the whole state and/or action space. Moreover, it is usually impossible to store the whole state and/or action space in memory without function approximation. As a result, an approximate version of the algorithm has been proposed to improve the efficiency of the learning procedure, including approximate value iteration (AVI) [5, 16, 11, 26] and approximate policy iteration (API) [5, 25, 21]. Next, similar as that MPI generalize VI and PI as reviewed in Section 2.1, we can generalize AVI and API as an approximate modified policy iteration (AMPI) algorithm as proposed in [37]. A general version of AMPI is given by

$$\pi_k = \text{Greedy}_{\epsilon'_k}(Q_{k-1}), \qquad Q_k = (T_{\pi_k})^m Q_{k-1} + \epsilon_k. \tag{4}$$

The first step in (4) is approximate policy greedy step with greedy step error ϵ'_k. This means that $T_\pi Q_{k-1} \leq T_{\pi_k} Q_{k-1} + \epsilon'_k$ for any π. The second step is approximate policy evaluation step with evaluation step error ϵ_k. In this paper, we consider the AMPI-Q algorithm in [37] and work with the action-value function Q. Here $\epsilon'_k = 0$ and ϵ_k corresponds to the approximation error by using a function in RKHS to update the Q function. In the following, we first introduce the function class in Section 3.1, and then propose the algorithm for finite and infinite m in Section 3.2 and 3.3, respectively.

3.1 Function Class

Suppose $\mathcal{F}$ is an RKHS defined on the state space $\mathcal{S}$ with inner product $\langle\cdot,\cdot\rangle_{\mathcal{F}}$ and norm $\|\cdot\|_{\mathcal{F}}$. According to the definition of RKHS, for all $s \in \mathcal{S}$, we have $|f(s)| \leq M_s \cdot \|f\|_{\mathcal{F}}$ for any $f \in \mathcal{F}$ where M_s is a constant depending on s. Next, we denote the function class on state-action space as follow:

$$\mathcal{H} = \mathcal{F}^{\mathcal{A}} = \{f : \mathcal{S} \times \mathcal{A} \to \mathbb{R} : f(\cdot, a) \in \mathcal{F} \text{ for any } a \in \mathcal{A}\}.$$

Define the inner product on $\mathcal{H}$ as $\langle f, g\rangle_{\mathcal{H}} = \sum_{a\in\mathcal{A}} \big\langle f(\cdot,a), g(\cdot,a)\big\rangle_{\mathcal{F}}$, and the norm on $\mathcal{H}$ as $\|f\|_{\mathcal{H}}^2 = \sum_{a\in\mathcal{A}} \|f(\cdot,a)\|_{\mathcal{F}}^2$. We can verify that for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$ we have

$$|f(s,a)| \leq M_s \cdot \|f(\cdot,a)\|_{\mathcal{F}} \leq M_s \cdot \|f\|_{\mathcal{H}}.$$

This shows that the function class $\mathcal{H} = \mathcal{F}^{\mathcal{A}}$ is also an RKHS with inner product defined above. Note that the definition of inner product and norm on $\mathcal{H}$ may not be unique. Throughout this paper we will assume the state space is given by $\mathcal{S} = [0,1]^d$ where d is the dimension. In the following, we denote $\mathcal{X} = \mathcal{S} \times \mathcal{A}$ as the state-action space, and $x = (s,a)$ as a realization of state-action pair. Denote τ as some fixed distribution over $\mathcal{X}$. We will use this $\mathcal{H} = \mathcal{F}^{\mathcal{A}}$ as the RKHS for our proposed algorithms.

Throughout the paper we will use the following definition of α-Hölder function space

$$\Theta^\alpha(L) = \left\{ f : \left| f^{(\lfloor\alpha\rfloor)}(x) - f^{(\lfloor\alpha\rfloor)}(y) \right| \leq L \cdot \|x - y\|_\infty^{\alpha - \lfloor\alpha\rfloor} \right\}, \tag{5}$$

where $f^{(\lfloor\alpha\rfloor)}(x)$ denotes the $\lfloor\alpha\rfloor^{th}$ derivative of f. We are now ready for the algorithms.

3.2 Implementation of AMPI Algorithm for Finite m.

We start from an initial action-value function Q_0. In each iteration k we first sample n i.i.d. state action pairs $(s_i^{(0)}, a_i^{(0)})$ from a fixed distribution $\tau \in \mathcal{P}(\mathcal{S}\times\mathcal{A})$ as mentioned at the end of Section 3.1, and then for each observation i we generate a rollout of size m, according to the current action-value function Q_k. Specifically, at each $t = 0, 1, ..., m-1$, we observe the current state $s_i^{(t)}$ and take action $a_i^{(t)}$. We sample reward $r_i^{(t)} \sim r\big(\cdot\,|s_i^{(t)}, a_i^{(t)}\big)$ and the transition state $s_i^{(t+1)} \sim P\big(\cdot\,|s_i^{(t)}, a_i^{(t)}\big)$. At the new state, we sample the action according to the policy greedy rule as $a_i^{(t+1)} \sim \pi_{k+1}\big(s_i^{(t+1)}\big)$ with the policy greedy rule defined as

$$\pi_{k+1}(s) = \underset{a\in\mathcal{A}}{\operatorname{argmax}}\ Q_k(s,a). \tag{6}$$

Next, we calculate the target value as

$$y_i = \sum_{t=0}^{m-1} \gamma^t r_i^{(t)} + \gamma^m \max_{a\in\mathcal{A}} \Big[Q_k\big(s_i^{(m)}, a\big) \Big]. \tag{7}$$

Note that the superscript t in γ^t is for exponent and the superscript (t) in $s_i^{(t)}, a_i^{(t)}, r_i^{(t)}$ are for the rollout. Under this setting, we have that

$$\mathbb{E}\Big[y_i \Big| s_i^{(0)}, a_i^{(0)} \Big] = (T_{\pi_{k+1}})^m Q_k\Big(s_i^{(0)}, a_i^{(0)}\Big).$$

Therefore, an intuitive update rule is to find a state-action function Q such that the error terms $y_i - Q(s_i^{(0)}, a_i^{(0)})$ are small. Here we choose to minimize the mean square error with an ℓ_2 regularization term given by the RKHS norm. Specifically, with Q_k the current action-value function, we update Q_{k+1} by

$$Q_{k+1} = \underset{Q\in\mathcal{H}}{\operatorname{argmin}}\ \frac{1}{n}\sum_{i=1}^n \Big(y_i - Q(s_i^{(0)}, a_i^{(0)}) \Big)^2 + \lambda \cdot \|Q\|_{\mathcal{H}}^2. \tag{8}$$

In the next section, we will assume that $(T_\pi Q)^m \in \Theta^\alpha(L)$ for any Q and π, so that solving (8) is a regularized nonparametric RKHS regression problem with true value given by a function in $\Theta^\alpha(L)$. According to the representer theorem of RKHS, we can explicitly write down the solution to (8). Denote $x_i^{(0)} = (s_i^{(0)}, a_i^{(0)})$ as a state-action pair with X^0 as the stack of $\{x_i^{(0)}\}_{i=1}^n$ and Y as the stack of $\{y_i\}_{i=1}^n$, we have

$$Q_{k+1}(x) = K(x, X^{(0)})\big[K(X^{(0)}, X^{(0)}) + n\lambda I_n\big]^{-1} Y,$$

where $K(U,V)$ denotes the matrix $\big[K(u_i, v_j)\big]$ with U, V as the stack of $\{u_i\}$ and $\{v_j\}$.

The overall algorithm is summarized in Algorithm 1.

Algorithm 1 Approximate modified policy iteration with RKHS regression

```
Input: Initial estimator $Q_0$, number of iteration $K$, function class $\mathcal{H}$, number of samples $n$.
for $k = 0, 1, 2..., K-1$ do
    Sample $n$ i.i.d. states and actions $\big\{\big(s_i^{(0)}, a_i^{(0)}\big)\big\}_{i=1}^{\infty}$ from fixed distribution $\tau$.
    For each sample $i$, perform a rollout of size $m$ as the follow with $t = 0, ..., m-1$:
        $r_i^{(t)} \sim r\big(\cdot\,|s_i^{(t)}, a_i^{(t)}\big)$            // sample the reward function
        $s_i^{(t+1)} \sim P\big(\cdot\,|s_i^{(t)}, a_i^{(t)}\big)$          // sample the transition state
        $a_i^{(t+1)} \sim \pi_{k+1}\big(s_i^{(t+1)}\big)$               // sample the action where $\pi_{k+1}(\cdot)$ is defined in (6)
    For each sample $i$, calculate $y_i = \sum_{t=0}^{m-1} \gamma^t r_i^{(t)} + \gamma^m \max_{a\in\mathcal{A}} \big[Q_k\big(s_i^{(m)}, a\big)\big]$ as in (7).
    Update $Q_{k+1}$ as the solution of (8).
end for
Compute the greedy policy with respect to $Q_K$ as $\pi_K$.
Output: Policy $\pi_K$ and estimated $Q$ function $Q_K$.
```

3.3 Implementation of AMPI Algorithm with $m = \infty$.

With $m = \infty$ which corresponds to the policy iteration, Algorithm 1 fails since we cannot generate rollouts with infinite size in practice. Here we adopt the regularized policy iteration algorithm proposed in [12]. We first define the empirical Bellman operator. Suppose we have n i.i.d data $\big\{(s_i, a_i)\big\}_{i=1}^{n}$ with $(s_i, a_i) \sim \tau$ and we sample reward $r_i \sim r(\cdot|s_i, a_i)$ and transition state $s_i' \sim P(\cdot|s_i, a_i)$. For a given policy π, the empirical Bellman operator $\widehat{T}_\pi^i$ is defined as

$$\widehat{T}_\pi^i Q(s_i, a_i) = r_i + \gamma Q\big(s_i', \pi(s_i')\big).$$

Clearly the empirical Bellman operator is an unbiased estimator of the Bellman operator in that $\mathbb{E}[\widehat{T}_\pi^i Q(s_i, a_i)|s_i, a_i] = T_\pi Q(s_i, a_i)$. Following the suggestions of [12], we propose two methods: Regularized Least Squares Temporal Difference Learning (LSTD) and Regularized Bellman Residual Minimization (BRM). We detail the first one as follow and put the second on in supplementary material. We refer to [12] for the detailed discussion.

The Least Squares Temporal Difference Learning (LSTD) algorithm is proposed in [7] where we aim to find the fixed point of the equation $Q = \Pi_\tau T_\pi Q$ where Π_τ is a weighted projection on $\mathcal{H}$ defined as $\Pi_\tau Q = \operatorname{argmin}_{h\in\mathcal{H}} \|h - Q\|_\tau^2$. Here the τ-weighted ℓ_2 norm is defined as usual as $\|Q\|_\tau = \big[\int Q(x)^2 \tau(dx)\big]^{1/2}$. Follow the suggestions in [12], we modify this approach and aim to minimize $\|Q - \Pi_\tau T_\pi Q\|_\tau^2$. Once again, we choose the ℓ_2 regularization term and the regularized LSTD algorithm at iteration k is given by minimizing the sample version

$$\begin{aligned} \widehat{h}_Q &= \operatorname*{argmin}_{h\in\mathcal{H}} \Big[\frac{1}{n}\sum_{i=1}^{n}\Big(h(s_i, a_i) - \widehat{T}_{\pi_k}^i Q(s_i, a_i)\Big)^2 + \lambda\cdot\|h\|_{\mathcal{H}}^2\Big], \\ Q_k &= \operatorname*{argmin}_{Q\in\mathcal{H}} \Big[\frac{1}{n}\sum_{i=1}^{n}\Big(Q(s_i, a_i) - \widehat{h}_Q(s_i, a_i)\Big)^2 + \lambda\cdot\|Q\|_{\mathcal{H}}^2\Big]. \end{aligned} \tag{9}$$

With these two regularized approaches, we are ready for our algorithm for policy iteration. We start from some an initial Q_0 and in each iteration k we sample n i.i.d data $\big\{(s_i, a_i, r_i, s_i')\big\}_{i=1}^{n}$ as mentioned before. This step corresponds to the sampling procedure in Algorithm 1 with size $m = 1$. We then perform the approximate policy evaluation step as in (16) or (9) to update Q^{k+1}. Note that since the function class is an RKHS, we can obtain closed form solution to (16) and (9). See Theorem 10 in [12]. Finally, the policy update is obtained by policy greedy with respect to Q^{k+1} as before. The overall algorithm is summarized in Algorithm 2 in the supplementary material.

4 Theoretical Results

In this section, we provide a theoretical result for our proposed algorithm. We start by stating some mild assumptions on the function class $\mathcal{H}$ and $\Theta^\alpha(L)$ as defined in Section 3.1. Recall that the kernel K has the eigendecomposition $K(x, y) = \sum_{j=1}^{\infty} \eta_j\, \varphi_j(x)\, \varphi_j(y)$ as in Section 2.2.

Assumption 1. *The eigenvalues of the kernel K satisfy $\eta_j \asymp j^{-2\alpha/d}$.*

Assumption 2. *The eigenfunctions of the kernel K are bounded: $|\varphi_j(x)| \leq C$ for all j and $x \in \mathcal{X}$. Moreover, let $\delta(x, y)$ be a distance measure on $\mathcal{X}$, we have the Lipschitz property $|\varphi_j(x) - \varphi_j(y)| \leq Cj \cdot \delta(x, y)$ for all j and $x, y \in \mathcal{X}$.*

Assumption 3. *For any $Q \in \mathcal{H}$ and any policy π, we have that $(T_\pi Q)^m \in \Theta^\alpha(L)$ and $(T_\pi Q)^m \in \mathcal{H}$ where T_π is the Bellman operator with respect to π.*

Assumption 4. *The sampling density τ is lower bounded from 0 on the support.*

Assumption 1 and 2 are standard assumptions in RKHS function spaces. For example, these two conditions are satisfied for the Matérn kernel with smoothness index $(\alpha - d/2)$ expanded with respect to the Fourier basis where d is the dimension of $\mathcal{X}$. Since in our case $\mathcal{X}$ involves both state and action space, we can define a metric on both of them and use product metric to construct the distance measure $\delta(x, y)$. This distance measure can then be used to construct the Matérn kernel. See [6, 53] for details. For Assumption 3, the first part is a mild condition on the function class $\Theta^\alpha(L)$ so that it is rich enough. According to the definition of the Bellman operator (2), we can see that this condition is satisfied if both the transition probability distribution and the reward function of the MDP are sufficiently smooth. See [55] for more detailed discussions. The second part corresponds to Assumption A6 in [12] (no function approximation error). As argued in [12], this assumption is standard and is satisfied for RKHS with universal kernels. This additional assumption does not weaken the contribution of our proposed analysis. Assumption 4 is required so that the samples cover the whole space on the support. It is in in the same spirit as the constant concentrability condition. However, in Assumption 4 we do not take the supreme over the policies, which is required in the constant concentrability condition (1). We are now ready for our main theorem for both finite and infinite m. The following Theorem 5 quantifies the error propagation of the algorithm.

Theorem 5 (error propagation)**.** *For large enough n, after K iterations of Algorithm 1 or 2, with probability at least $1 - n^{-4}$ we have that*

$$\left\|Q^* - Q^{\pi_K}\right\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2} \cdot \max_{0 \leq k \leq K-1} \left\|Q_{k+1} - (T_{\pi_{k+1}})^m Q_k\right\|_\infty + \frac{4\gamma^K}{(1-\gamma)^2} \cdot R_{\max}. \tag{10}$$

In order to quantify the first term (one step approximation error) in (10), we adopt the theoretical analysis in RKHS regression in [52]. However, in [52] the authors assume i.i.d Gaussian noise and the noise is independent with the covariate x, whereas in our case the noise is given by $w_i = y_i - (T_{\pi_{k+1}})^m Q_k\big(s_i^{(0)}, a_i^{(0)}\big)$, and it is dependent with the state action pair $\big(s_i^{(0)}, a_i^{(0)}\big)$. This imposes additional difficulty on the proof. The following Theorem 6 quantifies one step approximation error.

Theorem 6 (one step approximation error)**.** *Suppose Assumptions 1 - 3 are satisfied. Set λ as*

$$\lambda = c_0 \cdot \left(\frac{\log n}{n}\right)^{\frac{2\alpha}{2\alpha+d}}. \tag{11}$$

For large enough n, with probability at least $1 - n^{-4}$ we have

$$\left\|Q_{k+1} - (T_{\pi_{k+1}})^m Q_k\right\|_\infty \leq C \cdot \left(\frac{\log n}{n}\right)^{\frac{\alpha}{2\alpha+d}}, \tag{12}$$

for any k and for some constant c_0 and C depends on $L, R_{\max}, \gamma$, and α.

From Theorem 5 we see that the convergence rate of the algorithm consists of two parts. The first term in (10) is the statistical error with the explicit rate given by (12) in Theorem 6. The second term in (10) is the optimization error which converges to zero in linear rate with respect to the number of iterations. Suppose the number of iteration K satisfies

$$K \geq \Big[\log C + \frac{\alpha}{2\alpha + d}\log(\log n/n)\Big] / \log(1/\gamma)$$

with some constant C, then the convergence rate in (10) is dominated by the statistical error part. According to (11) and (12), the convergence rate (up to logarithm term) under supremum norm is given by $n^{-\alpha/(2\alpha+d)}$, which matches the minimax rate of convergence in Hölder class [57, 8]. Our analysis provides an upper bound of the supremum norm of the error term, which is a stronger result compared to the literature which mostly focuses on ℓ_p norm. The proof of Theorem 5 and 6 are provided in supplementary materials.

References

[1] András Antos, Csaba Szepesvári, and Rémi Munos. Fitted q-iteration in continuous action-space mdps. In *Advances in neural information processing systems*, pages 9–16, 2008.

[2] András Antos, Csaba Szepesvári, and Rémi Munos. Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71(1):89–129, 2008.

[3] Mohammad Gheshlaghi Azar, Vicenç Gómez, and Hilbert J Kappen. Dynamic policy programming. *Journal of Machine Learning Research*, 13(Nov):3207–3245, 2012.

[4] Yannick Baraud et al. A bernstein-type inequality for suprema of random processes with applications to model selection in non-gaussian regression. *Bernoulli*, 16(4):1064–1085, 2010.

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

[6] Anirban Bhattacharya and Debdeep Pati. Posterior contraction in gaussian process regression using wasserstein approximations. *Information and Inference: A Journal of the IMA*, 6(4):416–440, 2017.

[7] Steven J Bradtke and Andrew G Barto. Linear least-squares algorithms for temporal difference learning. *Machine learning*, 22(1-3):33–57, 1996.

[8] Andrea Caponnetto and Ernesto De Vito. Optimal rates for the regularized least-squares algorithm. *Foundations of Computational Mathematics*, 7(3):331–368, 2007.

[9] Jinglin Chen and Nan Jiang. Information-theoretic considerations in batch reinforcement learning. *arXiv preprint arXiv:1905.00360*, 2019.

[10] Xiaohong Chen and Timothy M Christensen. Optimal sup-norm rates and uniform inference on nonlinear functionals of nonparametric iv regression. *Quantitative Economics*, 9(1):39–84, 2018.

[11] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

[12] Amir-massoud Farahmand, Mohammad Ghavamzadeh, Csaba Szepesvári, and Shie Mannor. Regularized policy iteration with nonparametric function spaces. *The Journal of Machine Learning Research*, 17(1):4809–4874, 2016.

[13] Amir-massoud Farahmand, Csaba Szepesvári, and Rémi Munos. Error propagation for approximate policy and value iteration. In *Advances in Neural Information Processing Systems*, pages 568–576, 2010.

[14] Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. *arXiv preprint arXiv:1512.08562*, 2015.

[15] Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized markov decision processes. *arXiv preprint arXiv:1901.11275*, 2019.

[16] Geoffrey J Gordon. Stable function approximation in dynamic programming. In *Machine Learning Proceedings 1995*, pages 261–268. Elsevier, 1995.

[17] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*, 2015.

[18] Adel Javanmard and Andrea Montanari. Confidence intervals and hypothesis testing for high-dimensional regression. *The Journal of Machine Learning Research*, 15(1):2869–2909, 2014.

[19] Michail G Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of machine learning research*, 4(Dec):1107–1149, 2003.

[20] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

[21] Alessandro Lazaric, Mohammad Ghavamzadeh, and Rémi Munos. Finite-sample analysis of least-squares policy iteration. *Journal of Machine Learning Research*, 13(Oct):3041–3074, 2012.

[22] Amir massoud Farahmand, Mohammad Ghavamzadeh, Csaba Szepesvári, and Shie Mannor. Regularized fitted q-iteration: Application to planning. In *European Workshop on Reinforcement Learning*, pages 55–68. Springer, 2008.

[23] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[25] Rémi Munos. Error bounds for approximate policy iteration. In *ICML*, volume 3, pages 560–567, 2003.

[26] Rémi Munos. Performance bounds in l_p-norm for approximate value iteration. *SIAM journal on control and optimization*, 46(2):541–561, 2007.

[27] Rémi Munos and Csaba Szepesvári. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research*, 9(May):815–857, 2008.

[28] Gerhard Neumann and Jan R Peters. Fitted q-iteration by advantage weighted regression. In *Advances in neural information processing systems*, pages 1177–1184, 2009.

[29] Yang Ning, Han Liu, et al. A general theory of hypothesis tests and confidence regions for sparse high dimensional models. *The Annals of Statistics*, 45(1):158–195, 2017.

[30] Stephen David Patek. *Stochastic and shortest path games: theory and algorithms*. PhD thesis, Massachusetts Institute of Technology, 1997.

[31] Julien Perolat, Bruno Scherrer, Bilal Piot, and Olivier Pietquin. Approximate dynamic programming for two-player zero-sum markov games. In *International Conference on Machine Learning (ICML 2015)*, 2015.

[32] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[33] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer, 2005.

[34] Mark Rudelson, Roman Vershynin, et al. Hanson-wright inequality and sub-gaussian concentration. *Electronic Communications in Probability*, 18, 2013.

[35] Bruno Scherrer. Performance bounds for λ policy iteration and application to the game of tetris. *Journal of Machine Learning Research*, 14(Apr):1181–1227, 2013.

[36] Bruno Scherrer. Approximate policy iteration schemes: a comparison. In *International Conference on Machine Learning*, pages 1314–1322, 2014.

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

[38] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[39] Paul J Schweitzer and Abraham Seidmann. Generalized polynomial approximations in markovian decision processes. *Journal of mathematical analysis and applications*, 110(2):568–582, 1985.

[40] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[41] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[42] Bernard W Silverman et al. Spline smoothing: the equivalent variable kernel method. *The Annals of Statistics*, 12(3):898–916, 1984.

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

[44] Manel Tagorti and Bruno Scherrer. On the rate of convergence and error bounds for lstd (λ). In *International Conference on Machine Learning*, pages 1521–1529, 2015.

[45] Christophe Thiery and Bruno Scherrer. Least-squares λ policy iteration: Bias-variance trade-off in control problems. In *International Conference on Machine Learning*, 2010.

[46] Sara Van de Geer, Peter Bühlmann, Ya'acov Ritov, Ruben Dezeure, et al. On asymptotically optimal confidence regions and tests for high-dimensional models. *The Annals of Statistics*, 42(3):1166–1202, 2014.

[47] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[48] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior (commemorative edition)*. Princeton university press, 2007.

[49] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

[50] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT Press Cambridge, MA, 2006.

[51] Ronald J Williams and Leemon C Baird. Tight performance bounds on greedy policies based on imperfect value functions. Technical report, Citeseer, 1993.

[52] Yun Yang, Anirban Bhattacharya, and Debdeep Pati. Frequentist coverage and sup-norm convergence rate in gaussian process regression. *arXiv preprint arXiv:1708.04753*, 2017.

[53] Yun Yang and Debdeep Pati. Bayesian model selection consistency and oracle inequality with intractable marginal likelihood. *arXiv preprint arXiv:1701.00311*, 2017.

[54] Yun Yang, Zuofeng Shang, and Guang Cheng. Non-asymptotic theory for nonparametric testing. *arXiv preprint arXiv:1702.01330*, 2017.

[55] Zhuoran Yang, Yuchen Xie, and Zhaoran Wang. A theoretical analysis of deep q-learning. *arXiv preprint arXiv:1901.00137*, 2019.

[56] Ming Yu, Mladen Kolar, and Varun Gupta. Statistical inference for pairwise graphical models using score matching. In *Advances in Neural Information Processing Systems*, pages 2829–2837, 2016.

[57] Tong Zhang. Learning bounds for kernel regression using effective data dimensionality. *Neural Computation*, 17(9):2077–2098, 2005.