2020 59th IEEE Conference on Decision and Control (CDC)
Jeju Island, Republic of Korea, December 14-18, 2020

Finite Time Guarantees for Continuous State MDPs with Generative Model

Hiteshi Sharma and Rahul Jain

Abstract— **In this paper, we present Online Empirical Value Learning (ONEVaL), an 'online' reinforcement learning algorithm for continuous MDPs that is 'quasi-model-free' (needs a generative/simulation model but not the model per se) that can compute nearly-optimal policies and comes with non-asymptotic performance guarantees including prescriptions on required sample complexity for specified performance bounds. The algorithm relies on use of a 'fully' randomized policy that will generate a β-mixing sample trajectory. It also relies on randomized function approximation in an RKHS for arbitrarily small function approximation error, and an 'empirical' estimate of value from the next state by several samples of the next state from the generative model. We demonstrate its' good numerical performance on some benchmark problems. We note that the algorithm requires no hyper-parameter tuning, and is also robust to other concerns that seem to plague Deep RL algorithms.**

I. Introduction

Recently, reinforcement learning (RL) algorithms have been shown to have remarkably good performance on simulated control problems ranging from video games to locomotion tasks [13], [20], [9]. Unfortunately, these methods are rather complicated, require a lot of data to perform well [7],and do not yield much insight. Furthermore, it has been shown [8] that many of these RL algorithms are not robust to changes in hyperparameters, network architectures, random seeds, or even different implementations of the same algorithm. In fact, a large amount of time is typically spent on hyper-parameter tuning than acually training of the algorithm. Furthermore, [10] showed that a simple linear policy-based method with weights updated by a random search method can outperform some of these state-of-the-art results. A key question is whether the remarkable performance of these algorithms is more than just random search, and whether algorithms based on more principled approaches could be designed that could potentially match the remarkable performance (eventually).

There is a large class of model-based algorithms based on dynamic programming (DP) ideas [17] including for continuous state space MDPs. Unfortunately, in many problems (e.g., robotics), the system model is unknown, or simply too complicated to be succintly stated and used in DP algorithms. Usually, latter is the more likely case. Thus, model-free algorithms such as Q-Learning have been in popular usage [21]. The problem with such stochastic approximation algorithms is that they are too slow to converge as shown in [4], [3], [2]. An alternative has been 'empirical' algorithms such as [15], [4] which replace the expectation in the Bellman operator with a sample average approximation obtained by getting multiple samples of the next state for each state-action pair. This requires access to a generative model. At first glance, this seems restrictive, and somewhat short of the goal of an 'online' algorithm for reinforcement learning. But in a large variety of applications from video games [13] to robotics, we *do* have access to a generative model of the system (even if it is too complicated for theoretical analysis). In fact, in robotics, for example, training of RL algorithms is first done offline in high-fidelity simulators as the training takes too long to converge for it to be done direcly on the physical robot.

In this paper, we propose an 'online' RL algorithm for continuous state space MDPs that is quasi-model-free in the sense of only needing access to a generative model (e.g., in a simulator), and unlike other approaches [13], [20], [9] provably converges to the optimal policy with finite time (or non-asymptotic) guarantees. The idea is inspired by the 'empirical' algorithms proposed in [15], [4]. While [4] only considered finite state MDPs, [15] considered continuous MDPs but with user-specified deterministic function class which may result in a large approximation error. Recently, in [5], the authors seem to have gotten around this problem by leveraging [19] and doing randomized function approximation in an RKHS. Though, this algorithm for continuous MDPs is 'quasi-model-free' (essentially model-free but needs a generative model), computes nearly-optimal policies and comes with theoretical performance guarantees, it is still an 'offline' algorithm. An 'online' version of [15] was presented in [1] but that algorithm can have a large optimality gap for continuous MDPs.

The main contribution of this paper is an 'online' quasi-model-free reinforcement learning algorithm for continuous MDPs that computes nearly-optimal policies and also comes with non-asymptotic theoretical performance guarantees. We assume that we can use a randomized stationary policy that yields a β-mixing sample trajectory. At each instant, we do a value function update by doing a randomized function approximation in an RKHS. This involves getting multiple next state samples by use of the generative model, and then use them for an 'empirical' estimate of the expectation in the Bellman operator. The function approximation is performed on the base-points in some finite window of the sample trajectory. We prove convergence and yield finite time sample complexity bounds by combining the β-mixing analysis with the random operator theoretic analysis and randomized function approximation analysis in [4], [5].

Rahul Jain and Hiteshi Sharma are with the EE Department at the University of Southern California. They were supported by NSF Awards CCF-1817212 and ECCS-1810447. The authors would also like to thank Yifan Sun for help with experiments. Email address: (rahul.jain,hiteshis)@usc.edu

978-1-7281-7447-1/20/$31.00 ©2020 IEEE

II. Problem Formulation

Consider a discounted MDP $(\mathcal{X}, \mathcal{A}, P, r, \gamma)$ where $\mathcal{X}$ is the state space and $\mathcal{A}$ is the action space. The transition probability kernel is given by $P(\cdot|x, a)$, i.e., if action a is executed in state x, the probability that the next state is in a Borel-measurable set B is $P(x_{t+1} \in B | x_t = x, a_t = a)$ where x_t and a_t are the state and action at time t. The reward function is $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$. We are interested in maximizing the infinite horizon expected discounted reward where the discount parameter is γ. Let Π denote the class of stationary deterministic Markov policies mappings $\pi : \mathcal{X} \to \mathcal{A}$ which only depend on history through the current state. We only consider such policies since it is well known that there is an optimal MDP policy in this class. When the initial state is given, any policy π determines a probability measure P^{π}. Let the expectation with respect to this measure be $\mathbb{E}^{\pi}$. We focus on infinite horizon discounted reward criterion. The expected infinite horizon discounted reward or the value function for a policy π and initial state x is given as

$$v^{\pi}(x) = \mathbb{E}^{\pi} \left[\sum_{t=0}^{\infty} \gamma^t \, r(x_t, a_t) \middle| x_0 = x \right].$$

The optimal value function is given as $v^*(x) = \sup_{\pi \in \Pi} v^{\pi}(x)$ and the policy which maximizes the value function is the optimal policy, π^*. We make the following regularity assumption on the MDP.

Assumption 1: (**Regularity of MDP**) The state space $\mathcal{X}$ is a compact subset of d-dimensional Euclidean space. The rewards are uniformly bounded by $r_{\max}$, i.e, $|r(x, a)| \leq r_{\max}$ for all $(x, a) \in \mathcal{X} \times \mathcal{A}$. Furthermore, there exists $L_P, L_r > 0$ such that for all $(x, x', a) \in \mathcal{X} \times \mathcal{X} \times \mathcal{A}$, we have

$$\|P(\cdot|x, a) - P(\cdot|x', a)\|_{TV} \leq L_P \, \|x - x'\|_2$$
$$|r(x, a) - r(x', a)| \leq L_r \, \|x - x'\|_2$$

where TV denotes the total variation norm. The assumption above implies that for any policy π, $v^{\pi} \leq v_{\max} = r_{\max}/(1-\gamma)$. Let $B(\mathcal{X})$ be the set of functions $f : \mathcal{X} \to \mathbb{R}$ such that $\|f\|_{\infty} \leq v_{\max}$. Let us now define the Bellman operator $T : B(\mathcal{X}) \to B(\mathcal{X})$ as follows:

$$T \, v(x) = \max_a \left[r(x, a) + \gamma \mathbb{E}_{x' \sim P(\cdot|x,a)} v(x') \right].$$

It is well known that the operator T is a contraction with respect to $\| \cdot \|_{\infty}$ norm and the contraction parameter is the discount factor, γ. Hence, the sequence of iterates $v_k = T \, v_{k-1}$ converge to v^* geometrically. It is easy to check that T maps bounded functions to Lipschitz functions with Lipschitz constant $L_r + \gamma \, L_P \, v_{\max}$.

We can pick a function space in which to do function approximation which can provide an arbitrarily close approximation to any continuous function. As in [19], [5], we pick a parametrized reproducible kernel Hilbert space (RKHS) and do randomized function approximation via random features which can be called Random parametric basis functions (RPBF). Let Θ be a set of parameters and let $\phi : \mathcal{X} \times \Theta \to \mathbb{R}$ be a feature function. The feature functions need to satisfy $\sup_{(x,\, \theta) \in \mathcal{X} \times \Theta} |\phi\,(x;\, \theta)\,| \leq 1$. One example of such basis functions are Fourier features. Let $\mathcal{F}\,(\Theta) \triangleq \left\{ f\,(\cdot) := \int_{\Theta} \phi\,(\cdot;\, \theta)\, \alpha\,(\theta)\, d\theta \,|\, : \alpha\,(\theta)\,| \leq C\, \nu\,(\theta)\,,\, \forall \theta \in \Theta \right\}$. We are interested in finding the best fit within finite sums of the form $\sum_{j=1}^{J} \alpha_j \phi\,(s;\, \theta_j)$. Doing classical function fitting with $\sum_{j=1}^{J} \alpha_j \phi\,(s;\, \theta_j)$ leads to non-convex optimization problems because of the joint dependence in α and θ. Instead, we fix a density ν on Θ and draw a random sample θ_j from Θ for $j = 1, 2, \ldots J$. Once these $(\theta_j)_{j=1}^{J}$ are fixed, consider the function space $\widehat{\mathcal{F}}\left(\theta^{1:J}\right) \triangleq \left\{ f\,(\cdot) = \sum_{j=1}^{J} \alpha_j \phi\,(\cdot;\, \theta_j) \,:\, \|\,(\alpha_1, \ldots, \alpha_J)\,\|_{\infty} \leq C/J \right\}$.

Now, it remains to calculate weights α by minimizing a convex loss. Furthermore, let us define the $L_{2,\mu}$ norm of a function for a given a probability distribution μ on $\mathcal{X}$ as $\|f\|_{2,\,\mu}^2 = \left(\int_{\mathcal{X}} |f\,(x)\,|^2 \mu\,(dx)\right)$. The empirical norm at given samples $(x_1, x_2, \ldots x_N)$ is defined as $\|f\|_{2,\,\hat{\mu}}^2 = \frac{1}{N} \sum_{i=1}^{N} |f(x_i)|^2$. Recall two distance measures for function spaces: $d_{2,\,\mu}\,(T\,f,\, \mathcal{F}) := \inf_{f' \in \mathcal{F}} \|f' - T\,f\|_{2,\,\mu}$ is the approximation error for a specific f and $d_{2,\,\mu}\,(T\,\mathcal{F},\, \mathcal{F}) := \sup_{f \in \mathcal{F}} d_{2,\,\mu}\,(T\,f,\, \mathcal{F})$ is the inherent Bellman error for the entire class $\mathcal{F}$. Now, because of Lipschitz continuity assumption on the MDP (Assumption 1), we need to choose a function space that is dense in the space of Lipschitz functions. In fact, $\mathcal{F}\,(\Theta)$ is shown to be dense in the space of continuous functions [18]. Hence, for such a function class $d_{2,\,\mu}\,(T\,\mathcal{F},\, \mathcal{F}) = 0$.

III. The Algorithm and the Main Result

We now present our ONline Empirical Value Learning (`ONEVaL`) algorithm. We assume that we have interactions $(x_1, x_2, \ldots x_N, x_{N+1}, \ldots.)$ generated by a randomized policy π_g i.e., $x_{t+1} \sim P(\cdot|x_t, a_t)$ where $a_t \sim \pi_g(\cdot|x_t)$. We assume that for any a and x, $\pi_g(a|x)$ is strictly positive. We keep a window of size N which moves forward one step in every iteration. Now these N samples serve as the states for which we will compute our approximate value function and then use function approximation to generalize. It is clear that these samples are not i.i.d. Hence, we need some mixing conditions on this stochastic process. Mixing conditions quantify the decrease in dependence as the future moves farther from the past. There are various types of mixing conditions, of which the following are popular in statistical learning theory: α-mixing, β-mixing and ϕ- mixing. As mentioned in [22, Chapter 2], ϕ-mixing is too strong an assumption while α-mixing is too weak but β-mixing is "just right". Let us now revisit the definition of β-mixing:

Definition 1 (β-mixing): Let $\{X_t\}_{t=1}^{\infty}$ be the sequence of random variables from a probability space $(\Omega, \mathfrak{F}, \mathbb{P})$. Let $\mathbb{P}_i^j$ define the joint distribution of random variables $(X_i, X_{i+1} \ldots X_j)$, $j \geq i$. Then, the β-mixing coefficient is defined as

$$\beta_m := \sup_{t \geq 1} \|\mathbb{P}_1^t \times \mathbb{P}_{t+m}^{\infty} - \mathbb{P}_{t,m}\|_{TV}$$

where $\|\cdot\|_{TV}$ is the total variation norm and $\mathbb{P}_{t,m}$ is the joint distribution of $(X_1^t, X_{t+m}^{\infty})$. A stochastic process is said to be β-mixing if $\beta_m \to 0$ as $m \to \infty$.

β-mixing coefficient measures the distance between the joint distribution of random variables separated by m time units and the distribution under which they are independent. β-mixing processes could be either algebraically mixing or exponentially mixing (we focus on the latter)[12]. A β-mixing process is said to mix at an exponential rate with parameters $b, c > 0$ if $\beta_m = O(\exp(-bm^c))$. Now, we state the mixing assumption on our samples:

Assumption 2 (β-mixing): The sequence of samples $\{x_t\}_{t\geq 1}$ is β-mixing at an exponential rate, i.e., there exists positive constants b, c, c_1 such that $\beta_m = c_1 \exp(-bm^c)$. Furthermore, this is a strictly stationary process and $x_t \sim \mu$.

As mentioned earlier, we focus on *fully randomized* policy (i.e., each action is taken with positive probability in each state) for generating these interactions which is known to induce a Markov process satisfying the β-mixing properties [16]. But one could modify by sampling from these interactions, for instance, sampling from experience replay buffer [14]. As long as the mixing properties are satisfied, we are guaranteed to generalize well. In fact, in the experiments section, we show that sampling from previous samples satisfy the mixing conditions. Since we will be analyzing the convergence in L_2 norm, we no longer have the contraction property. Hence, we need bounded Radon-Nikodym derivatives of transitions which we illustrate in the next assumption. Such an assumption has been used earlier in [15], [5]:

Assumption 3: (**Stochastic Transitions**) For all $(x,\, a) \in \mathcal{X} \times \mathcal{A}$, $P\,(\cdot\,|\,x,\, a)$ is absolutely continuous with respect to μ and $C_\mu \triangleq \sup_{(x,\,a)\in\mathcal{X}\times\mathcal{A}} \left\| \frac{dP(\cdot\,|\,x,\,a)}{d\mu} \right\|_\infty < \infty$.
When there is an uncertainty in the underlying environment, computing expectation in the Bellman operator is expensive. If we have a generative model of the environment, we can replace the expectation by empirical mean leading to definition of empirical Bellman operator:

$$\widehat{T}_M\, v(x) := \max_a \left[r(x,a) + \frac{\gamma}{M} \sum_{i=1}^{M} v(x_i^{x,a}) \right]. \tag{1}$$

where $x_i^{x,a} \sim P(\cdot|x,a)$ for $i = 1, 2 \ldots M$. Note that the next state samples, x', are i.i.d. If the environment is deterministic, like Atari games or locomotion tasks, having a single next state suffices and we don't need a generative model.

Given the data $\{(x_n,\, \widehat{v}\,(x_n))\}_{n=1}^N$, we fit the value function over the state space by computing a best fit within $\widehat{\mathcal{F}}\left(\theta^{1:J}\right)$ by solving

$$\begin{aligned} &\min_\alpha \frac{1}{N} \sum_{n=1}^{N} |\sum_{j=1}^{J} \alpha_j \phi\,(x_n;\, \theta_j) - \widehat{v}\,(x_n)\,|^2 \qquad (2) \\ &\text{s.t.} \quad \|\,(\alpha_1, \ldots,\, \alpha_J)\,\|_\infty \leq C/J. \end{aligned}$$

This optimization problem only optimizes over weights $\alpha^{1:J}$ since parameters $\theta^{1:J}$ have already been randomly sampled from a given distribution ν. Let $\Pi_{\widehat{\mathcal{F}}}(J,N)$ denote this optimization problem which we denote as $\Pi_{\widehat{\mathcal{F}}}$ for compact notation. We are now ready to present our algorithm ONEVaL, shown in Algorithm 1. Step 1 selects the samples at which we compute approximate value function. Step 2 samples the next states for a given state and action. This is not difficult when we have a generative model for various learning tasks, for, instance, in robotics. Step 3 computes the approximate labels which Step 4 generalizes via function approximation. ONEVaL can be seen as an iteration of a composition of two random operators. Let us define $\widehat{G}(N, M, J, \mu, \nu) = \Pi_{\widehat{\mathcal{F}}}(N, J) \circ \widehat{T}_M$ where $\circ$ denotes the composition. Let $\widehat{G}$ be a compact notation for this operator. Hence, in our algorithm $v_{k+1} = \widehat{G}\, v_k$. We will use the random operator framework developed in [4], [5] for analysis. Denote $\bar{v}_\epsilon = \left(512\, v_{\max}^2/(\epsilon/5)^4\right)$,

Algorithm 1 ONEVaL

```
Input: sample sizes $N, M, J \geq 1$; initial seed $v_0$; interactions from policy $\pi_g$: $(x_1, x_2, \ldots, x_N, x_{N+1}, \ldots x_{2N-1}, x_{2N}, \ldots)$

For k=0,1, 2, ...
  1) Select samples $(x_n)_{n=k+1}^{k+N}$ from given interactions
  2) For each $n$ and action $a$, sample i.i.d. next states $x_i^{x_n,a} \sim P(\cdot|x_n, a)$ for $i = 1, 2, \ldots m$
  3) Empirical value iteration: $\widehat{v}_{k+1}(x_n) = \widehat{T}_M\, v_k(x_n)$ for each $n$ according to (1)
  4) Function approximation: $v_{k+1} = \Pi_{\widehat{\mathcal{F}}}\, \widehat{v}_{k+1}(x_n)$ according to (2)
```

$$J_0(\epsilon, \delta) := \left[\frac{5\,C}{\epsilon} \left(1 + \sqrt{2\,\log\frac{5}{\delta}} \right) \right]^2,$$

$$M_0(\epsilon, \delta) := \bar{v}_\epsilon \log \left[\left(\frac{4\,e\,v_{\max}}{(\epsilon/5)^2} \right)^J \frac{40\,e\,N\,|\mathcal{A}|\,(J+1)}{\delta} \right]$$

$$\eta_{N_0}(\epsilon, \delta) := \bar{v}_\epsilon \log \left[\frac{160\,e\,(J+1)}{\delta} \left(\frac{200\,C\,\gamma\,e\,v_{\max}}{\epsilon^3} \right)^J \right]$$

Theorem 1: Suppose Assumptions 1, 2 and 3 hold and let $N = 2\,l_N\,\eta_N$. Choose an $\epsilon > 0$ and $\delta \in (0,1)$. Set $\delta' = 1 - (1/2 + \delta/2)^{1/(K^*-1)}$ and denote $\widetilde{C} := 4\left(\frac{1-\gamma^{K+1}}{1-\gamma}\right)^{1/2} C_\mu^{1/2}$ and

$$K^* := \left\lceil \frac{\log\left(C_\mu^{1/2}\epsilon\right) - \log\left(2\,v_{\max}\right)}{\log\gamma} \right\rceil. \tag{3}$$

Then, if $\eta_N \geq \eta_{N_0}(\epsilon, \delta')$, $l_N(\epsilon, \delta) \geq \left(\frac{1}{b} \log \frac{20\,\eta_N c_1}{\delta'} \right)^{1/c}$, $M \geq M_0(\epsilon, \delta')$, $J \geq J_0(\epsilon, \delta')$ and $K \geq \log\left(4/\left((1/2 - \delta/2)\,(1-q)\,q^{K^*-1}\right)\right)$, we have that with probability at least $1 - \delta$,

$$\|v_K - v^*\|_{2,\,\mu} \leq \widetilde{C}\,\epsilon.$$

a) Remark:: The above theorem states that if we have sufficiently large samples, in particular, $J = O\left(1/\epsilon^2 \log 1/\delta\right)$, $\eta_N = O\left(1/\epsilon^4 \log 1/\epsilon^{3J}\delta\right)$, $M = O\left(1/\epsilon^4 \log 1/\epsilon^{2J}\delta\right)$ and $l_N = O\left(\log 1/\delta\right)$ where $N = 2\,\eta_N\, l_N$, then for sufficiently large iterations, the approximation error can be made arbitrarily small with high probability. Moreover, if Lipschitz continuity assumption is not satisfied then the result can be presented in a more general form: $\|v_K - v^*\|_{2,\,\mu} \leq 2\left(\frac{1-\gamma^{K+1}}{1-\gamma}\right)^{1/2}\left[C_\mu^{1/2} d_{2,\,\mu}\left(T\,\mathcal{F}(\Theta),\, \mathcal{F}(\Theta)\right) + 2\epsilon\right]$.

IV. Proof of Main Theorem

There are two approximations interleaved in `ONEVaL`: approximation by sample average and function approximation. In addition to that, the samples for function approximation are not independent. We will use the technique developed in [24] to transform a sequence of dependent variables into a subsequence of nearly i.i.d. ones. Recall that the state samples for function approximation in iteration k are $(x_{k+1}, x_{k+2}, \ldots x_{k+N})$. For convenience, let us rewrite the sequence as $(\tilde{x}_{k,n})_{n=1}^N$ for a given iteration k. Let η_N and l_N be positive integers such that this sequence is divided in to $2\eta_N$ blocks, each of length l_N for all $k \geq 0$. We assume that $2\,\eta_N\, l_N = N$ (the remainder terms become insignificant as the sample size increases). Let us now construct the blocks for $1 \leq j \leq \eta_N$:

$$H_{k,j} = \{\tilde{x}_{k,i} : 2(j-1)l_N + 1 \leq i \leq (2j-1)l_N\}$$
$$T_{k,j} = \{\tilde{x}_{k,i} : (2j-1)l_N + 1 \leq i \leq 2jl_N\}$$

Let H_k be the entire sequence of odd blocks $\{H_{k,j}\}$ and let T_k be the sequence of even blocks $T_{k,j}$. Finally, let H'_k be a sequence of blocks which are independent of $\tilde{x}_{k,1}^N$ but such that each block has the same distribution as a block from the original sequence. Now, all the results for i.i.d variables can now be applied to the blocks H'_k. This technique has been used to extend the convergence of function approximation to a β-mixing sequence.

Step 1: Bound on one-step error.: Let us now bound the error in one iteration of `ONEVaL` and then we will use the stochastic dominance argument to analyze the iteration of the random operator. Let ϵ_k be the gap between the random operator $\widehat{G}$ and the exact Bellman operator at iteration k, i.e.,

$$v_{k+1} = \widehat{G}\, v_k = T\, v_k + \epsilon_k.$$

We want to bound the error term ϵ_k. Note that the functions from previous iteration, v_k, are random. Also, they are not independent of function in the next iteration, v_{k+1}, because the interactions samples are dependent. Next lemma bounds the error in one iteration of `ONEVaL`.

Lemma 2: Choose $\epsilon > 0$, and $\delta \in (0, 1)$. Let $N = 2\eta_N l_N$ for some positive integers η_N and l_N. Also choose $N \geq N_0(\epsilon, \delta), M \geq M_0(\epsilon, \delta)$ and $J \geq J_0(\epsilon, \delta)$. Then, for $\widehat{G}\,(N,\, M,\, J,\, \mu,\, \nu)\, v$, the output of one iteration of ONEVaL, we have with probability at least $1-\delta$,

$$\|\widehat{G}\, v - T\, v\|_{2,\,\mu} \leq d_{2,\,\mu}\left(T\,\mathcal{F}\,(\Theta)\,,\, \mathcal{F}\,(\Theta)\right) + \epsilon$$

for any arbitrary function $v \in \mathcal{F}\,(\Theta)$.

Step 2: Stochastic dominance.: After bounding the error in one-step, we will now bound the error when the random operator $\widehat{G}$ is applied iteratively by constructing a dominant Markov chain. Since we do not have a contraction with respect to L_2 norm, we need an upper bound on how the errors propagate with iterations. Recall that $\widehat{G}\, v_k = T\, v_k + \epsilon_k$, we use the point-wise error bounds as [15, Lemma 3]. For a given error tolerance, it gives a bound on number of iterations which we call K^* as shown in equation (3). We then construct a stochastic process as follows. We call iteration k "good" if the error $\|\epsilon_k\|_{2,\,\mu} \leq \epsilon$ and "bad" otherwise. We then construct a stochastic process $\{X_k\}_{k\geq 0}$ with state space $\mathcal{K}$ as $\triangleq \{1,\ 2, \ldots,\ K^*\}$ such that

$$X_{k+1} = \begin{cases} \max\{X_k - 1,\ 1\}\,, & \text{if iteration } k \text{ is "good"}, \\ K^*, & \text{otherwise.} \end{cases}$$

The stochastic process $\{X_k\}_{k\geq 0}$ is easier to analyze than $\{v_k\}_{k\geq 0}$ because it is defined on a finite state space, however $\{X_k\}_{k\geq 0}$ is not necessarily a Markov chain. Whenever $X_k = 1$, it means that we just had a string of K^* "good" iterations in a row, and that $\|v_k - v^*\|_{2,\,\mu}$ is as small as desired.

We next construct a "dominating" Markov chain $\{Y_k\}_{k\geq 0}$ to help us analyze the behavior of $\{X_k\}_{k\geq 0}$. We construct $\{Y_k\}_{k\geq 0}$ and we let $\mathcal{Q}$ denote the probability measure of $\{Y_k\}_{k\geq 0}$. Since $\{Y_k\}_{k\geq 0}$ will be a Markov chain by construction, the probability measure $\mathcal{Q}$ is completely determined by an initial distribution on $\mathbb{R}$ and a transition kernel for $\{Y_k\}_{k\geq 0}$. We now use the bound on one-step error as presented in Lemma 2 which states that when the samples are sufficiently large enough for all k,

$$\mathbb{P}\left(\|\epsilon_k\|_{2,\mu} \leq \epsilon\right) > q(N, M, J, L).$$

Denote this probability by q for a compact notation. Initialize $Y_0 = K^*$, and construct the process

$$Y_{k+1} = \begin{cases} \max\{Y_k - 1,\ 1\}\,, & \text{w.p. } q, \\ K^*, & \text{w.p. } 1-q, \end{cases}$$

where q is the probability of a "good" iteration which increases with sample sizes N, M, J and L. We now describe a stochastic dominance relationship between the two stochastic processes $\{X_k\}_{k\geq 0}$ and $\{Y_k\}_{k\geq 0}$. We will establish that $\{Y_k\}_{k\geq 0}$ is "larger" than $\{X_k\}_{k\geq 0}$ in a stochastic sense. This relationship is the key to our analysis of $\{X_k\}_{k\geq 0}$.

Definition 2: Let X and Y be two real-valued random variables, then X is *stochastically dominated* by Y, $X \leq_{st} Y$, when $\Pr\{X \geq \theta\} \leq \Pr\{Y \geq \theta\}$ for all θ in the support(Y).

The next lemma uses stochastic dominance to show that if the error in each iteration is small, then after sufficient iterations we will have small approximation error.

Lemma 3: Choose $\epsilon > 0$, and $\delta \in (0,\ 1)$, and suppose N, M, J and L are chosen sufficiently large enough such that $\mathbb{P}\left(\|\epsilon_k\|_{2,\mu} \leq \epsilon\right) > q$ for all $k \geq 0$. Then for $q \geq (1/2 + \delta/2)^{1/(K^*-1)}$ and $K \geq \log\left(4/\left((1/2 - \delta/2)\,(1-q)\,q^{K^*-1}\right)\right)$, we have with probability at least $1-\delta$,

$$\|v_K - v^*\|_{2,\mu} \le 2\left(\frac{1-\gamma^{K+1}}{1-\gamma}\right)^{1/2}\left[C_\mu^{1/2}\epsilon + \gamma^{K/2}\,(2\,v_{\max})\right]. \tag{4}$$

Combining Lemma 2 and Lemma 3 concludes the proof of Theorem 1.

V. Numerical Experiments

In this section, we present the performance of our proposed algorithm on two benchmark problems: optimal replacement and cart-pole.

a) Optimal replacement: As a proof of concept for the performance of our proposed algorithm, we pick the optimal replacement problem where we can explicitly compute the optimal value function. In the optimal machine replacement problem, there are two actions: keep the machine, or replace it. The state comprises of non-negative real numbers and the transition kernel is exponential. Further details can be found in [15]. For this experiment, we choose $J = 10$ random parameterized Fourier functions $\phi(x;\theta_j) = \cos(\theta_j^T x + b)$ with $\theta_j \sim \mathcal{N}(0, 0.01)$ and $b = \text{Unif}[-\pi, \pi]$. Fig. 1 shows the performance of `ONEVaL` for different N and M. Additionally, it shows the performance of two different methods of selecting the samples from interactions. The method consecutive is same as that presented in Algorithm 1. The method sampled refers to a variant where the instead of moving the window one sample, we uniformly sample from the interactions. We observe that increasing N and M improves the performance of the algorithm as we would expect. We also verify if the β-mixing assumption as stated in Assumption 2. Table I presents the estimated β-mixing coefficients for $N = 40000$ and $N = 80000$. To compute these coefficients, we follows the procedure in [11]. It shows that indeed increasing the lengths of the block indeed decreases the mixing coefficients.

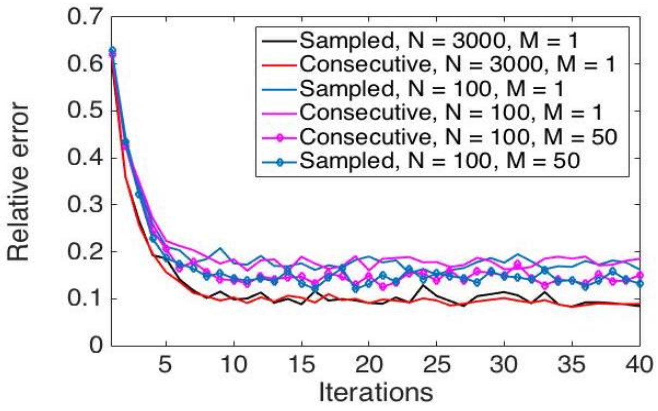

Fig. 1: Performance of ONEVaL for different sample sizes

l_N	N	Consecutive	Sampled
1	40000	15.88	0.93
10	40000	0.9281	0.77
100	40000	0.714	0.89
1000	40000	0.702	0.87
10	80000	0.52	0.59
100	80000	0.49	0.55
1000	80000	0.47	0.54

TABLE I: β-mixing coefficients

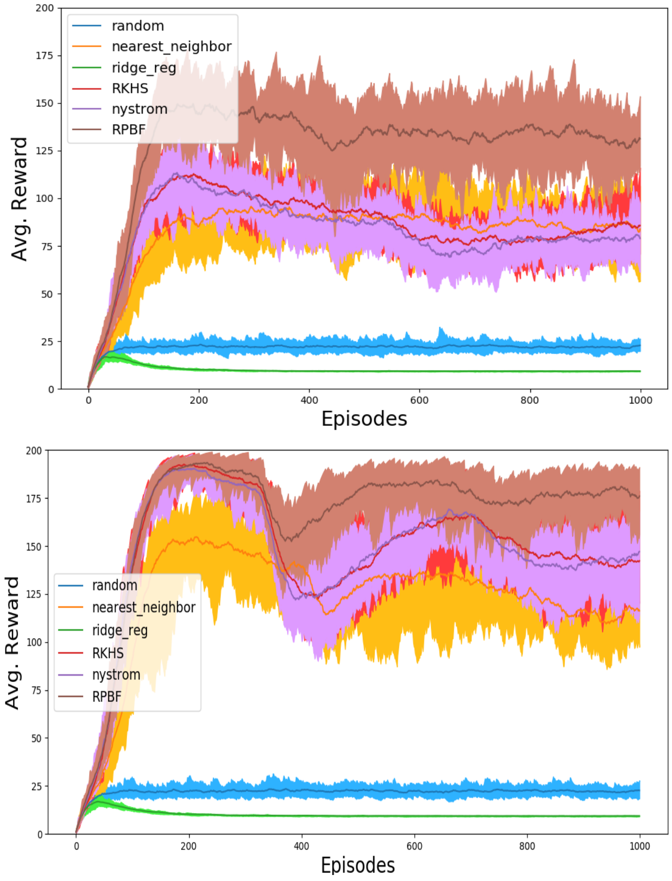

Fig. 2: Performance of ONEVaL on Cart-Pole. Expected (discounted) reward, averaged over 100 trials (solid), with the range of highest and lowest values in faded colors for $N = 1000$ and $N = 5000$ with $M = 1$

b) Cart-Pole: We now evaluate our methods on the cart-pole problem, using the OpenAI gym codebase. We heavily borrow from the environment constructed by the OpenAI gym for CartPole-v0 (see `https://gym.openai.com/envs/CartPole-v0/` for documentation). The state space contains four variables: position and velocity of the cart, and angle and angular velocity of the pole. We modify the code slightly to inject noise, such that actual force = force $+ z, \quad z \sim \mathcal{N}(0,1)$. The reward function is +1 if the pole is balanced (between ±12 degrees from vertical), and 0 otherwise. Since this is an episodic task, we uniformly sample from the interactions similar to experience replay buffer [14]. The exploration scheme used is ϵ-greedy with decaying ϵ. We compare the approximation methods against linear methods (ridge regression) popular nonlinear methods (10-nearest neighbors (NN)), kernel ridge regression (RKHS), Nystrom method ([23]) and the random parametric basis function (RPBF). We experimented with 1, 2, 5, 10, and 20 neighbors, finding 10 neighbors usually gave the best performance in these tests. For RKHS, we use the standard Guassian kernel. We also add in a random policy as a baseline to see if a method has learned anything at all.

	Batchsize 1000	Batchsize 5000
method	TPI	TPI
Ridge Reg	0.21	0.21
RKHS	0.70	1.18
Nystrom	0.97	0.94
RPBF	0.38	0.37
NN	2.21	2.23

TABLE II: CPU per-iteration runtime (in milliseconds).

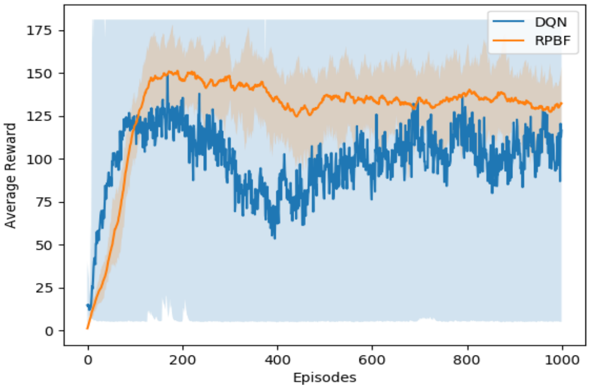

Fig. 3: ONEVaL vs DQN

Fig. 2 give the average achieved value at each episode for all methods, for different sample sizes. The only methods that do significantly better than random are nonlinear methods. Of these, RPBF appears a clear winner, with full kernel and Nystrom regression in second. Nearest neighbor can perform very well when the sample size is low, but has trouble when the sample size is high. Table II shows the average time per iteration (averaged between episodes 500 and 1000 of a single run) for each method, for sample sizes of 1000 and 5000.The runtime complexity of the approximation methods have about the same per-iteration runtime for batch sizes of 1000 and 5000, suggesting scalability; in contrast, the full kernel method runtime almost doubles. To compare the memory requirement, for KR it requires $O(N^2)$ for the kernel matrix, and for Nystrom (with L selected columns) and RPBF are $O(NL)$ and $O(NJ)$ respectively to store the approximation parameters. For $L \ll N$ and $J \ll N$, there is clear memory improvement of the approximation methods over the full kernel method. We also compare our algorithm with DQN as presented in [13] in Fig. 3 which shows the average performance with shaded region as the variance for 50 experiments . We use a 2 hidden layer architecture with 24 nodes each with ReLU activation. We notice that when the underlying environment is noisy, there is a lot of variance in the performance of DQN as also noted in [7].

REFERENCES

[1] András Antos, Csaba Szepesvári, and Rémi Munos. Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path. In *International Conference on Computational Learning Theory*, pages 574–588. Springer, 2006.

[2] Mohammad Gheshlaghi Azar, Remi Munos, M Ghavamzadaeh, and Hilbert J Kappen. Speedy q-learning. 2011.

[3] Adithya M Devraj and Sean Meyn. Zap q-learning. In *Advances in Neural Information Processing Systems*, pages 2235–2244, 2017.

[4] William B Haskell, Rahul Jain, and Dileep Kalathil. Empirical dynamic programming. *Mathematics of Operations Research*, 41(2):402–429, 2016.

[5] William B Haskell, Pengqian Yu, Hiteshi Sharma, and Rahul Jain. Randomized function fitting-based empirical value iteration. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2467–2472. IEEE, 2017.

[6] David Haussler. Decision theoretic generalizations of the pac model for neural net and other learning applications. *Information and Computation*, 100(1):78–150, 9 1992.

[7] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[8] Riashat Islam, Peter Henderson, Maziar Gomrokchi, and Doina Precup. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *arXiv preprint arXiv:1708.04133*, 2017.

[9] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[10] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning. *arXiv preprint arXiv:1803.07055*, 2018.

[11] Daniel Mcdonald, Cosma Shalizi, and Mark Schervish. Estimating beta-mixing coefficients. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 516–524, 2011.

[12] Ron Meir. Nonparametric time series prediction through adaptive model selection. *Machine learning*, 39(1):5–34, 2000.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[14] 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):529–533, 2015.

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

[16] Esa Nummelin and Pekka Tuominen. Geometric ergodicity of harris recurrent marcov chains with applications to renewal theory. *Stochastic Processes and Their Applications*, 12(2):187–202, 1982.

[17] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2005.

[18] Ali Rahimi and Benjamin Recht. Uniform approximation of functions with random bases. In *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pages 555–561. IEEE, 2008.

[19] Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in neural information processing systems*, pages 1313–1320, 2009.

[20] 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.

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

[22] Mathukumalli Vidyasagar. *A theory of learning and generalization*. Springer-Verlag, 2002.

[23] Tianbao Yang, Yu-Feng Li, Mehrdad Mahdavi, Rong Jin, and Zhi-Hua Zhou. Nyström method vs random fourier features: A theoretical and empirical comparison. In *Advances in neural information processing systems*, pages 476–484, 2012.

[24] Bin Yu. Rates of convergence for empirical processes of stationary mixing sequences. *The Annals of Probability*, pages 94–116, 1994.