Stochastic Optimization Methods for Policy Evaluation in Reinforcement Learning

Stochastic Optimization Methods for Policy Evaluation in Reinforcement Learning

Yi Zhou
University of Utah

Shaoconq Ma
University of Utah
s.ma@utah.edu

the essence of knowledge
Boston — Delft

Foundations and Trends® in Optimization

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

Y. Zhou and S. Ma. *Stochastic Optimization Methods for Policy Evaluation in Reinforcement Learning*. Foundations and Trends® in Optimization, vol. 6, no. 3, pp. 145–192, 2024.

ISBN: 978-1-63828-371-3

Part II

The Policy Evaluation Problem and Value-Based RL Algorithms

3

Introduction to The Policy Evaluation Problem

In this part, we introduce various RL algorithms that are developed based on value functions of states and actions. These RL algorithms aim to address the RL problem in the formulation I, and they are based on classic theories on MDP and Bellman type equations [36], [37], [39]. Specifically, we focus on the policy evaluation problem under the formulation I, which aims to learn the state value function associated with a given policy.

Recall that we have defined the state value function $V_\pi(s)$, which corresponds to the expected total reward that one can obtain by starting from state s and following policy π. This state value function plays a fundamental role in RL, as both the formulations I and II are based on it. In particular, one of the most fundamental question is the followng policy evaluation problem.

Given a fixed policy π, how to evaluate its state value function V_π?

We note that the above problem only aims to evaluate the value function of a given policy, and it does not talk about how to improve the policy. Nevertheless, as we show in later sections, policy evaluation algorithms are widely exploited by many policy optimization algorithms.

There are many algorithms for solving the policy evaluation problem, and they generally fall into two major categories that adopt different settings. Specifically, the first class of algorithms consider the setting where the transition kernel P of the environment is known and accessible, and the algorithms are based on the Bellman type equations. Other algorithms consider the complementary setting where the transition kernel P of the environment is unknown, and these algorithms utilize stochastic samples drawn from the MDP to iteratively learn the value function. Next, we provide a comprehensive introduction to both classes of policy evaluation algorithms.

4

Policy Evaluation with Known Transition Kernel

We first consider the scenario where the transition kernel P of the underlying environment is known a priori. This usually requires a good knowledge and modeling of the dynamic environment. In this case, by definition of the state value function, we can rewrite it as follows.

$$\begin{aligned} V_\pi(s) &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots | s_0 = s, \pi] \\ &= \mathbb{E}[r_0 + \gamma(r_1 + \gamma r_2 + \cdots) | s_0 = s, \pi] \\ &= \mathbb{E}[r_0 + \gamma V_\pi(s_1) | s_0 = s, \pi] \\ &= \sum_{a,s'} \mathsf{P}(s'|s,a)\pi(a|s)\big(r(s,a,s') + \gamma V_\pi(s')\big), \end{aligned}$$

where the third equality uses the definition of state value function starting at s_1, and the last equality expands the expectation over the randomness of a_0, s_1. Therefore, we obtain the following fundamental so-called Bellman equation for state value functions.

(Bellman equation and Bellman operator):

$$V_\pi(s) = \sum_{a,s'} \mathsf{P}(s'|s,a)\pi(a|s)\big(r(s,a,s') + \gamma V_\pi(s')\big) \stackrel{\triangle}{=} \mathsf{T}_\pi V_\pi(s).$$

For simplicity, we define the right hand side of the Bellman equation as an operator T_π applied to the state value function $V_\pi(s)$, and we call it the Bellman operator. It can be seen that the state value function is essentially a fixed point of the Bellman operator T_π, and in fact it is the unique fixed point. Moreover, the Bellman operator T_π is a linear operator that requires the knowledge of the transition kernel P. Based on these observations, the following classic algorithms can be applied to solve the above equation for the state value function.

- *Solving the linear equation.* Since the Bellman equation is a linear equation, we can directly solve it by computing a proper inverse. However, this requires high computation complexity;

- *Value iteration.* Viewing the Bellman equation as a fixed point equation, we can apply iterative fixed point updates to compute the value function. Specifically, starting with any $V_0(s)$, we update it iteratively as

$$V_{t+1}(s) = \mathsf{T}_\pi V_t(s), \quad \forall s.$$

  It can be shown that T_π is a contraction operator and therefore the above fixed point iterations converge to the true value function at a linear convergence rate [30].

The above methods can effectively learn the state value function given full knowledge of the transition kernel. However, in most cases, it is not possible to have such full knowledge, nor do we have a perfect model of the environment. This further motivates us to develop model-free approaches for policy evaluation.

5

Model-Free Policy Evaluation Algorithms

In this section, we consider the complementary setting, in which we do not assume access to the environment transition kernel P. Instead, we let the RL agent interact with the environment and collect samples from the MDP to learn the state value function. There are many model-free algorithms developed for policy evaluation, and they can be categorized into two categories based on the samples used: on-policy TD learning algorithms and off-policy TD learning algorithms.

5.1 On-Policy TD Learning

We first consider the on-policy setting. To elaborate, let π be the policy of which we want to evaluate the state value function V_π. Since we do not have any knowledge of the environment model, we follow the policy π and interact with the environment to generate a trajectory of samples queried from the MDP. Such a trajectory of samples is referred to as the on-policy data, which means that the data is generated by following the target policy π. We formally define it as follows.

Definition 5.1 (On-policy data). Let π be the target policy to be evaluated. Then, the MDP samples $\{s_t, a_t, r_t, s_{t+1}\}_{t=0}^{\infty}$ collected following π is called on-policy data.

We note that on-policy data must be collected under the target policy π. In particular, these samples are usually not independent as the transition of the states are based on the action generated by the policy and the underlying transition kernel. As we discuss later, such data dependence is a key challenge for analyzing the convergence of model-free policy evaluation algorithms. Next, we introduce several popular model-free algorithms for policy evaluation in the on-policy setting.

5.1.1 On-Policy TD(0) Algorithm

Recall the following Bellman equation for the state value function.

$$V_\pi(s) = \mathbb{E}\big[r(s,a,s') + \gamma V_\pi(s')\big].$$

Note that the expectation involves the transition kernel, which we do not have access to in the model-free setting. One simple idea is to leverage the on-policy data to approximate the above expectation and perform fixed point type updates. Specifically, suppose we start with an initialized value function V. Then, for every on-policy data sample (s_t, a_t, r_t, s_{t+1}), we can use $r_t + \gamma V(s_{t+1})$ to approximate the above expectation and perform the following update.

$$V(s_t) = \underbrace{r_t + \gamma V(s_{t+1})}_{\text{target}}$$

This makes sense because s_{t+1} is a realization of the follow-up state of s_t. We refer to the quantity $r_t + \gamma V(s_{t+1})$ as the 'target' for the current state value $V(s_t)$. The classic on-policy TD(0) algorithm is essentially a damped version of the above update, as shown in the following equation, where $0 < \eta < 1$ is a learning rate hyperparameter.

$$\begin{aligned}\text{TD(0):}\quad V(s_t) &= \eta V(s_t) + (1-\eta)\big(r_t + \gamma V(s_{t+1})\big)\\ &= V(s_t) + \eta\underbrace{\big(r_t + \gamma V(s_{t+1}) - V(s_t)\big)}_{\text{temporal difference (TD)}}\end{aligned}$$

The update quantity $r_t + \gamma V(s_{t+1}) - V(s_t)$ is referred to as the temporal difference, i.e., the difference between the target $r_t + \gamma V(s_{t+1})$ and the current estimate of the value function $V(s_t)$.

5.1.2 On-Policy TD(λ) Algorithm

The target $r_t + \gamma V(s_{t+1})$ used in TD(0) is based on one follow-up state transition, which usually results in a high variance estimation. TD(λ) [39] further improves the TD(0) algorithm by leveraging multi-step state transitions to form the target. Specifically, consider the following generalized n-step target

$$G_t^{(n)} := r_t + \gamma r_{t+1} + \cdots + \gamma^{n-1} r_{t+n-1} + \gamma^n V(s_{t+n}).$$

It can be seen that $G_t^{(1)}$ reduces to the target used in the TD(0) algorithm. Then, in TD(λ), we use a linear combination of n-step targets with geometric decaying coefficients to form the following G_t^λ-return, which is used as the target in the temporal difference term.

$$\text{TD}(\lambda)\text{:} \;\; V(s_t) = V(s_t) + \eta\big(G_t^\lambda - V(s_t)\big), \;\; \text{where} \;\; G_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}.$$

By leveraging the n-step targets over more state transition samples queried from the on-policy data, TD(λ) can effectively reduce the variance in its stochastic updates.

5.1.3 TD learning with Function Approximation

When we implement the aforementioned TD learning algorithms, we need to keep a table $V(\cdot)$ for the state value function whose size equals the cardinality $|\mathcal{S}|$ of the state space. This can be costly in both computation and storage aspects for many practical applications that have a large or even infinite state space. To address this curse of dimensionality issue, an effective and widely used technique is function approximation, i.e., we can use proper parameterized models to track and approximate the state value function. In particular, the following three types of parameterized models are popular choices in both theory and practice.

- *Linear model.* We model the value of any state as a parameterized linear function $V_\theta(s) = \phi_s^\top \theta$, where θ is the model parameter (to be learned) and ϕ_s is the feature vector associated with state s (fixed and pre-specified). Normally, the feature vector ϕ_s is designed either by domain knowledge or generated via certain distribution (Gaussian), and its dimensionality is much smaller than that for encoding the original state s.

- *Neural network model.* Since the state value function can be highly non-linear in the high dimensional state space, it is natural and motivating to exploit the great expressive power of modern neural networks (NN) to approximate it [35], [41]. Specifically, the state value function is modeled as $V_\theta(s) = \mathrm{NN}_\theta(s)$, where NN_θ denotes a general deep neural network with parameter θ. It takes the (encoded) state s as the input and outputs a scalar that approximates the corresponding state value.

- *General non-linear model.* In recent years, there have been modeling approach beyond the constraints of linearity and neural networks to encompass a broader class of functions [28], [45]. Here, we define the state value function as $V_\theta(s) = f_\theta(s)$, where f_θ represents a general non-linear function parameterized by θ. This structure considers the arbitrary smooth function approximation and allows for greater flexibility in capturing complex relationships within the state space, accommodating scenarios where neither linear nor neural network models suffice.

When adopting function approximation, the updates of TD learning algorithms need to be adjusted. For example, consider the TD(0) algorithm with function approximation. Suppose we start with an initialization θ_0 of the model parameters for the state value function V_θ. Then, in time t we observe an MDP sample (s_t, a_t, r_t, s_{t+1}) and define the target $G_t := r_t + \gamma V_{\theta_t}(s_{t+1})$. To update the model parameters, we define the quadratic loss $\ell_t(\theta) := \frac{1}{2}\big(V_\theta(s_t) - G_t\big)^2$ and consider applying gradient descent to optimize it. This leads to the following TD(0) update rule under function approximation.

TD(0) with function approximation:

$$\theta_{t+1} = \theta_t - \eta g_t(\theta_t), \text{ where } g_t(\theta_t) = \big(V_\theta(s_t) - G_t\big)\nabla_\theta V_\theta(s_t).$$

It can be seen that the updates take place in the model parameter space, as opposed to the state space in the standard TD(0) algorithm. Moreover, the temporal difference term in the update $g_t(\theta_t)$ is multiplied by the gradient term $\nabla_\theta V_\theta(s_t)$ to transform the update into the parameter space.

5.1.4 Analysis of TD(0) with Linear Function Approximation

In this section, we sketch the convergence analysis of the TD(0) algorithm with linear function approximation. Although TD(0) with linear function approximation is a simple and basic algorithm that has been widely used for policy evaluation, understanding their convergence behavior turns out to be highly non-trivial due to several major challenges. First, the TD update term g_t corresponds to the gradient of a time varying loss function ℓ_t, and such a non-fixed objective function makes it hard to apply the standard analysis of gradient descent algorithm. Second, unlike conventional stochastic optimization algorithms that use i.i.d. samples to compute the stochastic updates, TD(0) uses highly *dependent* samples queried from the underlying MDP, which cause additional statistical bias that needs to be quantified in the convergence analysis.

Summary of existing work. The asymptotic convergence of TD(0) has been extensively studied in earlier works by Benveniste *et al.* [3], Borkar [5], Tadić [40], and Tsitsiklis *et al.* [42]. In these studies, TD(0) was shown to asymptotically converge to the true state value function, but no convergence rate was established. Recently, the non-asymptotic (finite-time) convergence of TD(0) was established in Dalal *et al.* [12] with i.i.d. samples, and in Bhandari *et al.* [4] and Srikant *et al.* [34] with dependent Markovian samples queried from the MDP. In the following discussion, we sketch the key technical proof of non-asymptotic convergence of TD(0) with dependent Markovian samples developed in Bhandari *et al.* [4].

Throughout, we consider the following variant of TD(0) with the linear function approximation model $V_\theta(s) := \phi_s^\top \theta$ in a finite state space $\mathcal{S}$. Here, the operator Π_R is a projection operator onto the Euclidean ball with radius $R > 0$, which is to guarantee boundedness of model parameters in the learning process.

$$\theta_{t+1} = \Pi_R(\theta_t + \eta g_t(\theta_t)),$$
$$\text{where } g_t(\theta_t) = (r_t + \gamma\phi_{s_{t+1}}^\top \theta_t - \phi_{s_t}^\top \theta_t)\phi_{s_t}$$

As we elaborated before, one of the major challenge in the analysis is how to deal with the dependent samples queried from the MDP. To address this challenge, Bhandari *et al.* [4] considered the following geometric mixing property of the Markov chain associated with the underlying MDP.

Assumption 5.1 (Geometric mixing). Denote μ_π as the state stationary distribution associated with the MDP under the target policy π. Then, there exists $\kappa > 0$ and $\rho \in (0, 1)$ such that for all $t \geq 0$,

$$\sup_{s\in\mathcal{S}} d_{\mathrm{TV}}(\mathbb{P}(s_t \in \cdot | s_0 = s), \mu_\pi) \leq \kappa\rho^t,$$

where $d_{\mathrm{TV}}(P, Q)$ denotes the total-variation distance between the probability measures P and Q.

To explain, note that $\mathbb{P}(s_t \in \cdot | s_0 = s)$ denotes the distribution of state s_t conditioning on the initial state s_0, whereas μ_π is the state stationary distribution under the target policy π. Therefore, the above property assumes that the state distribution in the future t time steps converges to the state stationary distribution at a geometric rate, i.e., the more we look ahead, the closer we are to the state stationary distribution. In particular, such an assumption has been shown to hold for all irreducible and aperiodic Markov chains, which is the case for many RL scenarios.

To proceed, assume we have in total n distinct states and define the feature matrix $\Phi := [\phi_{s_1}^\top; \ldots; \phi_{s_n}^\top]$ where $\phi_s \in \mathbb{R}^d$ corresponds to the feature vector associated with state s. We assume that all the column feature vectors are independent, i.e., Φ has full column rank. Then, the entire value table can be written as $V_\theta := [V_\theta(s_1); \ldots; V_\theta(s_n)] = \Phi\theta$, which is a linear model. It has been shown in Bhandari *et al.* [4] that TD(0) with linear function approximation converges to the optimal model parameter θ^* that satisfies the following projected Bellman equation.

$$V_{\theta^*} = \Pi_{\mathcal{L}} \mathsf{T}_\pi V_{\theta^*}, \text{ where } \mathcal{L} := \{\Phi x | x \in \mathbb{R}^d\}.$$

We first state the main theorem on the finite-time convergence rate.

Theorem 5.1 (Finite-time convergence, [4])**.** Suppose Assumption 5.1 hold and choose learning rate $\eta \leq \mathcal{O}(\frac{1}{1-\gamma})$ for TD(0) with linear function approximation. Then, after T iterations, it holds that

$$\mathbb{E}\big[\|\theta_T - \theta^*\|^2\big] \leq \mathcal{O}\Big(\exp(-c\eta T)\|\theta_0 - \theta^*\|^2 + \eta \frac{\tau_{\text{mix}}(\eta)}{1-\gamma}\Big),$$

where $c > 0$ is a universal constant and $\tau_{\text{mix}}(\eta) := \min\{t \mid \kappa\rho^t \leq \eta\}$ is the mixing time of Markov chain.

The above theorem shows that with a constant learning rate, TD(0) with linear function approximation converges to a small neighborhood of the optimal model parameter θ^* at a linear convergence rate. Specifically, the radius of the neighborhood is proportional to the choice of learning rate η and the mixing time $\tau_{\text{mix}}(\eta)$ of the Markov chain. In particular, a larger learning rate η would imply a faster linear convergence factor $\exp(-c\eta T)$, but will lead to a larger convergence error $\eta\frac{\tau_{\text{mix}}(\eta)}{1-\gamma}$. Moreover, the mixing time $\tau_{\text{mix}}(\eta)$ characterizes how fast the state distribution $\mathbb{P}(s_t \in \cdot | s_0 = s)$ converges to the stationary distribution μ_π and affects the convergence error. In the extreme case where the samples are i.i.d. generated from μ_π (i.e., the mixing time is zero), the convergence error vanishes and we can achieve exact convergence.

Proof Sketch. Here, we provide a sketch of the proof to illustrate the key technical steps. Recall the TD(0) update with linear function approximation: $\theta_{t+1} = \Pi_R(\theta_t + \eta g_t(\theta_t))$, where the TD update $g_t(\cdot)$ depends on the sample $O_t = \{s_t, a_t, r_t, s_{t+1}\}$ queried from the underlying MDP. Then, we define the expected update $\bar{g}(\theta) := \mathbb{E}[g_t(\theta)]$, where $\mathbb{E}$ is taken over the randomness of the sample $O_t \sim \mu_\pi$. By the update rule of TD(0) and following some standard analysis steps, we can establish the following key inequality.

$$\begin{aligned} \mathbb{E}\big[\|\theta_{t+1} - \theta^*\|^2\big] \leq \mathbb{E}\big[\|\theta_t - \theta^*\|^2\big] - 2\eta(1-\gamma)\mathbb{E}\big[\|V_{\theta_t} - V_{\theta^*}\|_D^2\big] \quad (5.1)\\ + \eta \underbrace{\mathbb{E}\big[\langle g_t(\theta_t) - \bar{g}(\theta_t), \theta_t - \theta^*\rangle\big]}_{\text{Bias } \zeta(\theta_t, O_t)} + \mathcal{O}(\eta^2), \quad (5.2) \end{aligned}$$

where $\|\cdot\|_D$ is a norm defined via a certain positive definite matrix D, and it can be shown that $\mathbb{E}[\|V_{\theta_t} - V_{\theta^*}\|_D^2] \geq \sigma\|\theta_t - \theta^*\|^2$ for some

$\sigma > 0$. Therefore, in order to unroll the above inequality over t and establish the final convergence result, the key is to bound the bias term $\zeta(\theta_t, O_t) := \mathbb{E}\big[\langle g_t(\theta_t) - \bar{g}(\theta_t), \theta_t - \theta^* \rangle\big]$. To elaborate, if the sample O_t is sampled from the state stationary distribution μ_π independently, then we have $\mathbb{P}(O_t|\theta_t) = \mathbb{P}(O_t) = \mu_\pi$ and therefore $\mathbb{E}[g_t(\theta_t)|\theta_t] = \bar{g}(\theta_t)$, which implies that the bias term $\zeta(\theta_t, O_t) = 0$. However, in an MDP, the sample O_t is actually correlated with the previous samples $\{O_k\}_{k=0}^{t-1}$ and therefore $\mathbb{P}(O_t|\theta_t) \neq \mathbb{P}(O_t)$ (note that θ_t is generated by using previous samples).

To bound the bias term $\zeta(\theta_t, O_t)$ with dependent samples, Bhandari *et al.* [4] introduced the following de-correlation technique. To elaborate, we first rewrite the bias term as the following summation series.

$$\zeta(\theta_t, O_t) = \zeta(\theta_{t-\tau}, O_t) + \sum_{i=t-\tau}^{t-1} \big(\zeta(\theta_{i+1}, O_t) - \zeta(\theta_i, O_t)\big).$$

In particular, by using the update rule, the last summation term $\sum_{i=t-\tau}^{t-1} \big(\zeta(\theta_{i+1}, O_t) - \zeta(\theta_i, O_t)\big)$ can be upper bounded by $G^2\eta\tau$ for some $G > 0$, and this bound can be controlled by choosing a proper learning rate $\eta > 0$ and hyperparameter $\tau > 0$. On the other hand, the other bias term $\zeta(\theta_{t-\tau}, O_t)$ now involves $\theta_{t-\tau}$, which depends on the samples that are observed before time $t - \tau - 1$. By the geometric mixing property of the Markov chain, the correlation between these old samples and the sample O_t diminishes geometrically with regard to τ, and therefore the bias term can be effectively bounded as follows.

$$\mathbb{E}[\zeta(\theta_{t-\tau}, O_t)] \leq 2\|\zeta\|_\infty \sup_s d_{TV}\big(\mathbb{P}(s_t|s_{t-\tau} = s), \mu\big) \leq 4G^2\kappa\rho^\tau.$$

Substituting the above two inequalities into (5.2) and rearranging terms, we can obtain a recursion on $\mathbb{E}\big[\|\theta_t - \theta^*\|^2\big]$, which derives the final finite-time convergence rate via elementary calculation.

$\square$

5.1.5 Connection to Linear Stochastic Approximation

Under linear function approximation, the TD(0) algorithm has a close connection to the classic linear stochastic approximation (SA) algorithm. To explain, the classic linear SA algorithm takes the following update

$$\text{(Linear SA):} \quad \theta_{t+1} = \theta_t + \eta\big(A(O_t)\theta_t + b(O_t)\big),$$

where $A(O_t)$ is a matrix and $b(O_t)$ is a vector, both of which depend on a certain observation O_t from an underlying Markov chain. Then, the TD(0) algorithm with linear function approximation can be rewritten as the linear SA update by defining the following quantities.

$$\begin{aligned} O_t &= (s_t, s_{t+1})^\top \\ A(O_t) &= -\phi_{s_t}(\phi_{s_t}^\top - \gamma\phi_{s_{t+1}}^\top) \\ b(O_t) &= r_t\phi_{s_t} \end{aligned}$$

Therefore, the TD(0) algorithm with linear function approximation is a special case of the linear SA algorithm with Markovian samples, whose convergence has been established using Lyapunov-type analysis in Srikant *et al.* [33].

5.1.6 Variance-Reduced TD Learning

The variance-reduced TD learning is a variant of TD(0), addressing the high variance issue inherent in standard TD learning methods. This approach leverages techniques for variance reduction from SVRG [16], resulting in more stable and efficient learning algorithms. Based on the standard TD(0), the variance-reduced TD learning uses past information to estimate the "full gradient" to reduce the variance as described as follows: At the begining of each epoch m, the VRTD algorithm evaluates the pseudo-gradient of a large batch data to estimate such "full gradient". Then for each iteration, it ultilizes the batch peudo-gradient to reduce the variance of each stochastic gradient as

$$\theta_{m,t+1} = \theta_{m,t} + \alpha\left(g_{x_{j_{m,t}}}(\theta_{m,t}) - g_{x_{j_{m,t}}}\left(\tilde{\theta}_{m-1}\right) + g_m\left(\tilde{\theta}_{m-1}\right)\right) \quad (5.3)$$

Here, $g_x(\theta) := A_x\theta + b_x$ is the standard semi-gradient of TD-learning, $g_m\left(\tilde{\theta}_{m-1}\right) = \frac{1}{M}\sum_{x_i\in B_m} g_{x_i}\left(\tilde{\theta}_{m-1}\right)$ is the estimation of the "full gradient" using trajectory averaging, M and α are the batch size and the learning rate, respectively.

Summary of existing work. The convergence analysis of variance-reduced TD learning has been actively explored in Korda *et al.* [19], Mustafin *et al.* [29], and Xu *et al.* [47]. More explicitly, Xu *et al.* [47] provided a detailed analysis of the non-asymptotic convergence of a variance-reduced TD algorithm, establishing its superiority over traditional TD in terms of convergence rate and error reduction. It is worthy mention that as pointed out by Mustafin *et al.* [29], without applying the variance reduction techniques, the standard TD learning may achieve the sub-optimal convergence rate, and applying the variance reduction technqiue will fill this gap and achieve the minimax optimal dependence on the factor $\frac{1}{1-\gamma}$. In the following discussion, we sketch the key technical proof of non-asymptotic convergence of VRTD with i.i.d. samples developed in Xu *et al.* [47]. To be adapted to the linear approximation setting, the VRTD algorithm makes the following additional assumptions compared to the convergence analysis of TD-learning:

Assumption 5.2 (Bounded feature). $\|\phi_s\| \le 1$ for all $s \in \mathcal{S}$.

Assumption 5.3 (Non-singularity). The following matrix is non-singular

$$A := \mathbb{E}_{\mu_b,\pi_b}[\rho_{s,a}(\gamma\phi_s\phi_{s'}^\top - \phi_s\phi_s^\top)].$$

Assumption 5.2 is sufficiently mild since all bounded features can be normalized to 1. Assumption 5.3 ensures that the optimal parameter $\theta^* = -A^{-1}b$ exists and is unique. For convenience, in the convergence analysis of VRTD, we assume the data sample (s, a) are directly sampled from the distribution μ_π instead of following a stochastic process; the main idea of VRTD algorithm is the same. Here, we re-state its main theorem:

Theorem 5.2. Consider the VRTD algorithm in (5.3). Suppose Assumptions 5.2 and Assumption 5.3 hold. If the learning rate $\alpha < \frac{\lambda_A}{8(1+\gamma)^2}$ and the batch size $M > \frac{4(1+\gamma)^2\alpha^2+1}{\alpha[\lambda_A - 8\alpha(1+\gamma)^2]}$, then for all $m \in \mathbb{N}$,

$$\mathbb{E}\left[\left\|\tilde{\theta}_m - \theta^*\right\|^2\right] \le C_1^m \left\|\tilde{\theta}_0 - \theta^*\right\|^2 + \mathcal{O}(\alpha/M),$$

where $C_1 := \left(4\alpha(1+\gamma)^2 + \frac{4(1+\gamma)^2\alpha^2+1}{\alpha M}\right)\frac{1}{\lambda_A - 4\alpha(1+\gamma)^2}$ (with $C_1 < 1$ by appropriately choosing α and M).

This convergence upper bound demonstrates that, with a judiciously chosen learning rate α and batch size M, the VRTD algorithm linearly converges to a neighborhood of the optimal parameters, with an error order of $\mathcal{O}(\frac{\alpha}{M})$. Traditional convergence analysis of TD-learning, such as that in Bhandari *et al.* [4], presents an error term of $\mathcal{O}(\alpha)$, indicating the necessity for a small learning rate to ensure convergence to a desired approximation of the optimal parameters. In contrast, the VRTD algorithm permits the use of a constant-level learning rate. By choosing a sufficiently large batch size M, the algorithm can achieve high-accuracy solutions without the need to diminish the learning rate.

Proof Sketch. We briefly introduce the main steps of proving this theorem. Since the variance-reduction algorithm mainly focuses on the gap between the optimal parameter θ^* and the epoch-wise parameter $_m$, our first step is to bound the iteration within the m-th epoch. By considering the decomposition of the last update (i.e. the M-th iteration in the epoch), we have

$$\begin{aligned}\|\theta_{m,M} - \theta^*\|^2 =& \|\theta_{m,M-1} + \alpha(g_{x_{j_{m,M}}}(\theta_{m,M-1}) - g_{x_{j_{m,M}}}(\tilde{\theta}_{m-1}) + g_m(\tilde{\theta}_{m-1})) \\ &- \theta^*\|^2 \\ =& \|\theta_{m,M-1} - \theta^*\|^2 + 2\alpha(\theta_{m,M-1} - \theta^*)^\top (g_{x_{j_{m,M}}}(\theta_{m,M-1}) \\ &- g_{x_{j_{m,M}}}(\tilde{\theta}_{m-1}) + g_m(\tilde{\theta}_{m-1})) \\ &+ \alpha^2 \|g_{x_{j_{m,M}}}(\theta_{m,M-1}) - g_{x_{j_{m,M}}}(\tilde{\theta}_{m-1}) + g_m(\tilde{\theta}_{m-1})\|^2. \end{aligned} \tag{5.4}$$

Since we target to solve the recursion for the inner loop iteration over M, we aim to bound everything in the form of either $\|\theta_{m,M-1} - \theta^*\|^2$ or $\|\tilde{\theta}_m - \theta^*\|^2$. The inner product term contributes to the linear convergence as

$$\begin{aligned}&(\theta_{m,M-1} - \theta^*)^\top \left(g_{x_{j_{m,M}}}(\theta_{m,M-1}) - g_{x_{j_{m,M}}}\left(\tilde{\theta}_{m-1}\right) + g_m\left(\tilde{\theta}_{m-1}\right)\right) \\ \leq& -\lambda_A \|\theta_{m,M-1} - \theta^*\|^2 \end{aligned} \tag{5.5}$$

due the the negative definite of the matrix $A^T + A$ which largest eigenvalue is denoted as λ_A. The last term introduces the variance error:

$$\begin{aligned}
&\mathbb{E}\left[\left\|g_{x_{j_{m,M}}}\left(\theta_{m,M-1}\right) - g_{x_{j_{m,M}}}\left(\tilde{\theta}_{m-1}\right) + g_m\left(\tilde{\theta}_{m-1}\right)\right\|^2 \mid F_{m,M-1}\right] \\
&\leq 4(1+\gamma)^2\mathbb{E}\left[\|\theta_{m,M-1} - \theta^*\|_2^2 \mid F_{m,M-1}\right] + 4(1+\gamma)^2\mathbb{E}\left[\left\|\tilde{\theta}_{m-1} - \theta^*\right\|_2^2 \mid F_{m,M-1}\right] \\
&+ 2\mathbb{E}\left[\left\|g_m\left(\tilde{\theta}_{m-1}\right) - g\left(\tilde{\theta}_{m-1}\right)\right\|_2^2 \mid F_{m,M-1}\right]. \qquad (5.6)
\end{aligned}$$

Summarizing them together and solving the iteration, we get

$$\begin{aligned}
\mathbb{E}\left[\left\|\tilde{\theta}_m - \theta^*\right\|_2^2 \mid F_{m,0}\right] \leq &\frac{1/M + 4\alpha^2(1+\gamma)^2}{\alpha\lambda_A - 4\alpha^2(1+\gamma)^2}\left\|\tilde{\theta}_{m-1} - \theta^*\right\|_2^2 \\
&+ \frac{2\alpha}{\lambda_A - 4\alpha(1+\gamma)^2}\mathbb{E}\left[\left\|g_m\left(\tilde{\theta}_{m-1}\right) - g\left(\tilde{\theta}_{m-1}\right)\right\|_2^2 \mid F_{m,0}\right].
\end{aligned} \qquad (5.7)$$

The last term represents the variance error. It can be bounded as

$$\mathbb{E}\left[\left\|g_m\left(\tilde{\theta}_{m-1}\right) - g\left(\tilde{\theta}_{m-1}\right)\right\|_2^2 \mid F_{m,0}\right] \leq \frac{1}{M}\left(D_1\|\tilde{\theta}_{m-1} - \theta^*\|^2 + D_2\right), \qquad (5.8)$$

where $D_1 = 2(1+\gamma)^2$ and $D_2 = 4\left((1+\gamma)^2 R_\theta^2 + r_{\max}^2\right)$. The term differentiate the VRTD algorithm from the standard TD-learning analysis. Here, the variance error decreases as the batch size M increases. For $M = 1$, this analysis reduces to the standard TD-learning algorithm and have a constant variance. Solving the iteration over m leads to the final bound

$$\mathbb{E}\left[\left\|\tilde{\theta}_m - \theta^*\right\|_2^2\right] \leq C_1^m\left\|\tilde{\theta}_0 - \theta^*\right\|_2^2 + \frac{2D_2\alpha}{\left(1-C_1\right)\left(\lambda_A - 4\alpha(1+\gamma)^2\right)M}. \qquad (5.9)$$

$\square$

As highlighted in Mustafin *et al.* [29], the proof presented in Xu *et al.* [47] deviates from the traditional convex optimization framework, specifically regarding the convergence upper bound. The convergence upper bound should linearly depend on the condition number of the problem and SVRG algorithm should inherit this property. The derived complexity from the previous theorem gives $\tilde{\mathcal{O}}(\frac{1}{\lambda_A^2\epsilon})$, does not match the optimal dependence $\tilde{\mathcal{O}}(\frac{1}{\lambda_A\epsilon})$. This discrepancy has been addressed with an improved technique developed in Mustafin *et al.* [29].

5.2 Off-Policy TD Learning

In on-policy TD learning, the algorithm leverages the samples queried by following the target policy π. However, this usually requires implementing the target policy and interacting with the environment for a long episode, which can be time-consuming and unrealistic in many cases. In this section, we consider a complementary setting, i.e., the off-policy setting, where we have access to the MDP samples generated by following a certain fixed behavior policy π_b. We formally define it as follows.

Definition 5.2 (Off-policy data)**.** Let π_b be a behavior policy, which is in general different from the target policy π. Then, the MDP samples $\{s_t, a_t, r_t, s_{t+1}\}_{t=0}^{\infty}$ collected following π_b is called off-policy data.

We note that off-policy data is collected under the behavior policy π_b, and therefore the data distribution is different from that collected under the target policy π. Hence, in the algorithm design for off-policy TD learning, we must adjust the data distribution so that the learned value function is associated with the target policy. On the other hand, one may propose to apply the TD(0) with linear function approximation to off-policy evaluation. However, it has been shown that there exists simple examples for which this algorithm diverges in the off-policy setting [2]. To summarize, special TD learning algorithms need to be designed in the off-policy setting.

5.2.1 Gradient TD Framework for Off-Policy Evaluation

The gradient TD framework is a foundation for developing several advanced TD learning algorithms for off-policy evaluation [37]. Consider linear function approximation of the form $V_\theta(s) = \phi_s^\top \theta$ and recall that the optimal θ^* satisfies the projected Bellman equation $V_{\theta^*} = \Pi_{\mathcal{L}} \mathsf{T}^\pi V_{\theta^*}$. Then, it is natural to measure the quality of a parameter θ in state s based on the square error $(V_\theta - \Pi_{\mathcal{L}} \mathsf{T}^\pi V_\theta)^2$. In the off-policy setting, since the MDP data is collected following the behavior policy π_b, the states that we visit follow the state stationary distribution μ_b associated with π_b. Therefore, we can only evaluate the square error over the state stationary distribution μ_b, and this motivates us to consider the following mean-square projected Bellman error (MSPBE) [37].

$$\text{(MSPBE):} \quad J(\theta) := \mathbb{E}_{s\sim\mu_b}\big[V_\theta(s) - \Pi_{\mathcal{L}}\mathsf{T}^\pi V_\theta(s)\big]^2.$$

Since the above objective function takes a quadratic form, it is natural to consider applying gradient descent algorithms to optimize it over θ. Specifically, define the TD error term $\delta_t(\theta) := r_t + \gamma\phi_{s_{t+1}}^\top\theta - \phi_{s_t}^\top\theta$, then the above MSPBE can be rewritten as

$$J(\theta) = \mathbb{E}_{\mu_b,\pi}[\delta_t(\theta)\phi_{s_t}]^\top\mathbb{E}_{\mu_b}[\phi_{s_t}\phi_{s_t}^\top]^{-1}\mathbb{E}_{\mu_b,\pi}[\delta_t(\theta)\phi_{s_t}],$$

where $\mathbb{E}_{\mu_b}$is taken over $s_t \sim \mu_b$ and $\mathbb{E}_\pi$ is taken over the distribution of the next state s_{t+1}, which depends on $s_t, \pi(a_t|s_t)$ and $\mathsf{P}(s_{t+1}|a_t, s_t)$. However, the expectation $\mathbb{E}_{\mu_b,\pi}$ is hard to approximate using samples, because μ_b is induced by the behavior policy π_b while π is the target policy that we do not implement. In order to approximate these expectations using the off-policy data, we first rewrite the expectation into the following weighted sampling form.

$$\text{(Importance sampling):} \quad \mathbb{E}_{\mu_b,\pi}[\delta_t(\theta)\phi_{s_t}] = \mathbb{E}_{\mu_b,\pi_b}\Big[\frac{\pi(a_t|s_t)}{\pi_b(a_t|s_t)}\delta_t(\theta)\phi_{s_t}\Big],$$

where we define the ratio $\rho_t = \frac{\pi(a_t|s_t)}{\pi_b(a_t|s_t)}$ as the importance sampling ratio, which can be computed before hand given both policies π and π_b. In view of the importance sampling form, the expectation $\mathbb{E}_{\mu_b,\pi_b}$ now involves only the behavior policy and therefore can be estimated using the off-policy data (collected under the behavior policy). Then, we can compute the gradient of the MSPBE as follows, where the expectation is taken over μ_b and π_b.

$$-\frac{1}{2}\nabla J(\theta) = \mathbb{E}[\rho_t(\phi_{s_t} - \gamma\phi_{s_{t+1}})\phi_{s_t}^\top]\mathbb{E}[\phi_{s_t}\phi_{s_t}^\top]^{-1}\mathbb{E}[\rho_t\delta_t(\theta)\phi_{s_t}]. \quad (5.10)$$

To apply gradient descent, we need to estimate the expectation terms involved in the above gradient formula. However, as those expectation terms take a product form, using a single set of samples to estimate all of them will lead to a large bias. On the other hand, querying multiple independent sets of samples is usually unrealistic. In the following two subsections, we will introduce advanced two-timescale algorithm designs that allow us to accurately track the product of expectations using a single trajectory of off-policy data.

5.2.2 GTD2 Algorithm for Off-Policy Evaluation

The GTD2 algorithm, first developed in Sutton *et al.* [37], [38], leverages a two-timescale update design to decouple and estimate the product of expectations involved in the gradient formula in (5.10). More specifically, the main idea is to view the product of the last two expectations in (5.10) as an auxiliary variable, i.e., $\omega^*(\theta) := \mathbb{E}\big[\phi_{s_t}\phi_{s_t}^\top\big]^{-1}\mathbb{E}[\rho_t\delta_t(\theta)\phi_{s_t}]$. Then, the gradient of the MSPBE can be rewritten as

$$-\frac{1}{2}\nabla J(\theta) = \mathbb{E}\big[\rho_t(\phi_{s_t} - \gamma\phi_{s_{t+1}})\phi_{s_t}^\top\big]\omega^*(\theta). \tag{5.11}$$

Interestingly, the auxiliary variable $\omega^*(\theta)$ turns out to be the solution of the following special least-mean-square (LMS) problem.

$$\text{(LMS)}: \quad \omega^*(\theta) = \operatorname{argmin}_u \ \mathbb{E}\big[\phi_{s_t}^\top u - \rho_t\delta_t(\theta)\big]^2. \tag{5.12}$$

Therefore, to estimate the expectation term in (5.11), we can use off-policy data samples to approximate it. Moreover, to estimate the auxiliary variable $\omega^*(\theta)$ in (5.11), we can apply online stochastic gradient descent (SGD) to solve the LMS problem in (5.12) to obtain an approximated solution. This constitutes to the following two-timescale updates of the GTD2 algorithm.

$$\begin{aligned}\text{(GTD2)}: \quad \theta_{t+1} &= \theta_t + \alpha_t\rho_t(\phi_{s_t} - \gamma\phi_{s_{t+1}})\phi_{s_t}^\top\omega_t,\\ \omega_{t+1} &= \omega_t + \beta_t(\rho_t\delta_t(\theta_t)\phi_{s_t} - \phi_{s_t}\phi_{s_t}^\top\omega_t).\end{aligned}$$

To explain, in the first update, the term $\rho_t(\phi_{s_t} - \gamma\phi_{s_{t+1}})\phi_{s_t}^\top$ approximates the expectation term $\mathbb{E}\big[\rho_t(\phi_{s_t} - \gamma\phi_{s_{t+1}})\phi_{s_t}^\top\big]$ in (5.11). Moreover, the second update corresponds to an online SGD update that solves the LMS problem. We also note that $\alpha_t, \beta_t > 0$ are learning rates, and usually we set $\beta_t > \alpha_t$ so that the second timescale update adapts faster than the first timescale update. Intuitively, this is because ω_t tracks $\omega^*(\theta_t)$ while θ_t keeps being updated.

5.2.3 TDC Algorithm for Off-Policy Evaluation

The TDC algorithm also adopts two-timescale updates and is similar to the GTD2 algorithm, but it decouples the gradient formula in (5.10) in a slightly different way. Specifically, from the gradient formula in (5.11) we obtain that

$$\begin{aligned}-\frac{1}{2}\nabla J(\theta) &= \mathbb{E}\big[\rho_t(\phi_{s_t} - \gamma\phi_{s_{t+1}})\phi_{s_t}^\top\big]\omega^*(\theta) \\ &= \mathbb{E}\big[\rho_t\delta_t(\theta)\phi_{s_t}\big] - \gamma\mathbb{E}\big[\rho_t\phi_{s_{t+1}}\phi_{s_t}^\top\big]\omega^*(\theta),\end{aligned}$$

where the second equality leverages the definition of $\omega^*(\theta)$. The above gradient formula also decouples the product of expectations. Therefore, we can use off-policy samples to estimate the two expectations and then use online SGD to track $\omega^*(\theta)$ as we did in GTD2. This leads to the following two-timescale TDC updates.

$$\begin{aligned}\text{(TDC):}\quad \theta_{t+1} &= \theta_t + \alpha_t\rho_t(\delta_t(\theta_t)\phi_{s_t} - \gamma\phi_{s_{t+1}}\phi_{s_t}^\top\omega_t), \\ \omega_{t+1} &= \omega_t + \beta_t(\rho_t\delta_t(\theta_t)\phi_{s_t} - \phi_{s_t}\phi_{s_t}^\top\omega_t).\end{aligned}$$

It can be seen that both GTD2 and TDC use the same second timescale update to track the $\omega^*(\theta)$.

5.2.4 Analysis of TDC with Linear Function Approximation

To help understand the convergence behavior of two-timescale TD learning algorithms for off-policy evaluation, we provide a sketch of the finite-time convergence analysis of TDC with linear function approximation. In particular, we consider the following variant of TDC, which adopts two additional projection operations to avoid divergence of the parameters.

$$\begin{aligned}\theta_{t+1} &= \Pi_{R_\theta}\big(\theta_t + \alpha_t\rho_t(\delta_t(\theta_t)\phi_{s_t} - \gamma\phi_{s_{t+1}}\phi_{s_t}^\top\omega_t)\big), \\ \omega_{t+1} &= \Pi_{R_\omega}\big(\omega_t + \beta_t(\rho_t\delta_t(\theta_t)\phi_{s_t} - \phi_{s_t}\phi_{s_t}^\top\omega_t)\big).\end{aligned}$$

Here, $R_\theta, R_\omega > 0$ are the radius of Euclidean balls. There are two major challenges in the analysis. The first challenge is that the off-policy data samples are correlated with each other, as they are queried from the MDP induced by the behavior policy π_b. The second challenge is that the TDC has two updates that are intertwined with each other.

Summary of existing work. Two time-scale policy evaluation algorithms such as TDC and GTD2 were first introduced in Sutton *et al.* [37], [38], where the asymptotic convergence of both algorithms with i.i.d. samples were established. Their non-asymptotic convergence rates were established in Dalal *et al.* [11] as special cases of a two time-scale linear stochastic approximation (SA) algorithm. Recently, the non-asymptotic convergence analysis of TDC and two-time scale linear SA over Makovian samples were established in Xu *et al.* [48] and Kaledin *et al.* [17], respectively.

The finite-time analysis of TDC in the off-policy setting is based on the following two key assumptions.

Assumption 5.4 (Geometric mixing)**.** Denote μ_b as the state stationary distribution associated with the MDP under the behavior policy π_b. Then, there exists $\kappa > 0$ and $\rho \in (0, 1)$ such that for all $t \geq 0$,

$$\sup_{s\in\mathcal{S}} d_{\mathrm{TV}}(\mathbb{P}(s_t \in \cdot | s_0 = s), \mu_b) \leq \kappa\rho^t,$$

where $d_{\mathrm{TV}}(P, Q)$ denotes the total-variation distance between the probability measures P and Q.

Assumption 5.5 (Non-singularity)**.** The following matrices are non-singular

$$A := \mathbb{E}_{\mu_b,\pi_b}[\rho_{s,a}(\gamma\phi_s\phi_{s'}^\top - \phi_s\phi_s^\top)], \quad C := -\mathbb{E}_{\mu_b}[\phi_s\phi_s^\top].$$

These assumptions are critical for ensuring the convergence of TDC in the off-policy setting. As introduced in the previous section (Assumption 5.1) The geometric mixing assumption gives that $\mathbb{P}(s_t \in \cdot | s_0 = s)$, the distribution of state s_t conditioning on the initial state s_0, converges to the stationary distribution μ_π at a geometric rate. The non-singularity assumption guarantees that the optimal parameter $\theta^* = -A^{-1}b$ for $b = \mathbb{E}_{\mu_b,\pi_b}[\rho_s, ar(s, a, s')\phi_s]$ is well-defined and all feature vectors are linearly independent.

Under these assumptions, it has been shown in Xu *et al.* [48] and Kaledin *et al.* [17] that TDC with linear function approximation exhibits a finite-time convergence behavior in the off-policy setting. We re-state the main theorem as follows:

Theorem 5.3 (Finite-time convergence of TDC, [48])**.** Suppose Assumption 5.4 and Assumption 5.5 hold and choose diminishing learning rate $\alpha_t = \frac{c_\alpha}{(1+t)^\sigma}$ and $\beta_t = \frac{c_\beta}{(1+t)^\nu}$ with $0 < \nu < \sigma < 1$ for some constant c_α

and c_β. Let ϵ and ϵ' be sufficiently small. Then, after T iterations, it holds that

$$\mathbb{E}\,\|\theta_t-\theta^*\|_2^2 \le \mathcal{O}\left(e^{\frac{-|\lambda_\theta|c_\alpha}{1-\sigma}\left(t^{1-\sigma}-1\right)}\right)+\mathcal{O}\left(\frac{\log t}{t^\sigma}\right)+\mathcal{O}\left(\frac{\log t}{t^\nu}+h(\sigma,\nu)\right)^{1-\epsilon'}$$
$$\mathbb{E}\,\|z_t\|_2^2 \le \mathcal{O}\left(\frac{\log t}{t^\nu}\right)+\mathcal{O}(h(\sigma,\nu))$$

where $h(\sigma,\nu)=\begin{cases}\frac{1}{t^\nu}, & \sigma>1.5\nu\\ \frac{1}{t^{2(\sigma-\nu)-\epsilon}}, & \nu<\sigma\le 1.5\nu\end{cases}$. Moreover, if $0<\nu<\sigma=1$, there exists c_α and c_β such that

$$\mathbb{E}\,\|\theta_t-\theta^*\|_2^2 \le \mathcal{O}\left(\frac{(\log t)^2}{t}\right)+\mathcal{O}\left(\frac{\log t}{t^\nu}+h(1,\nu)\right)^{1-\epsilon'}$$

This theorem demonstrates that the two time-scale TDC algorithm with diminishing stepsizes converges to the optimal parameter θ^* at a rate depending on the choices of σ, ν for the two-time scale learning rates. The convergence rate is influenced by the tracking error z_t, which is a unique aspect of two time-scale algorithms. The terms in the bounds reflect the balance between the convergence rate and the accumulated error due to the diminishing stepsizes. In particular, the tracking error term $h(\sigma,\nu)$ captures how closely w_t follows the stationary point $\psi(\theta_t)=-C^{-1}(b+A\theta_t)$ at each step.

Proof Sketch. In this proof sketch, we outline the key technical steps to demonstrate the convergence of the two time-scale TDC algorithm. Instead of directly analyzing the convergence of $\{\theta_t\}$ and $\{w_t\}$, we incorporate the tracking error z_t into the TDC updates and reformulate the analysis in terms of $\{\theta_t\}$ and the tracking error $\{z_t\}$ as shown as follows:

$$\theta_{t+1}=\Pi_{R_\theta}\left(\theta_t+\alpha_t\left(f_1\left(\theta_t,O_t\right)+g_1\left(z_t,O_t\right)\right)\right),$$
$$z_{t+1}=\Pi_{R_w}\left(z_t+\beta_t\left(f_2\left(\theta_t,O_t\right)+g_2\left(z_t,O_t\right)\right)-C^{-1}\left(b+A\theta_t\right)\right)+C^{-1}\left(b+A\theta_{t+1}\right),$$

where

$$f_1\left(\theta_t,O_t\right)=\left(A_t-B_tC^{-1}A\right)\theta_t+\left(b_t-B_tC^{-1}b\right),g_1\left(z_t,O_t\right)=B_tz_t$$
$$f_2\left(\theta_t,O_t\right)=\left(A_t-C_tC^{-1}A\right)\theta_t+\left(b_t-C_tC^{-1}b\right),g_2\left(z_t,O_t\right)=C_tz_t$$

withe the observation at t-th time step $O_t=(s_a,a_t,r_t,s_{t+1})$.

Then we derive preliminary bound on $\mathbb{E}\|z_t\|_2^2$. We decompose the mean square tracking error $\mathbb{E}\|z_t\|_2^2$ into several components: an exponentially decaying term, a variance term, a bias term, and a slow drift term. Each of these components is then individually bounded, leading to a preliminary upper bound on $\mathbb{E}\|z_t\|_2^2$ of the order $\mathcal{O}(1/t^{\sigma-\nu})$.

To obtain a tighter bound, Xu *et al.* [48] recursively substituting the preliminary bound of $\mathbb{E}\|z_t\|_2^2$ into its own expression, particularly focusing on the slow drift term, refine the decay rate to $\mathbb{E}\|z_t\|_2^2 = \mathcal{O}(h(\sigma,\nu))$ as shown in the following bound:

$$\mathbb{E}\,\|z_t\|_2^2 \le \mathcal{O}\left(\frac{\log t}{t^\nu}\right) + \mathcal{O}(h(\sigma,\nu)),$$

where $h(\sigma,\nu) = \begin{cases} \frac{1}{t^\nu}, & \sigma > 1.5\nu \\ \frac{1}{t^{2(\sigma-\nu)-\epsilon}}. & \nu < \sigma \le 1.5\nu \end{cases}$ for any sufficiently small constant $\epsilon \in (0, \sigma-\nu]$.

Lastly, we can derive the final bound on $\mathbb{E}\|\theta_t - \theta^*\|_2^2$. The training error $\mathbb{E}\|\theta_t - \theta^*\|_2^2$ is decomposed into similar components as the tracking error: an exponentially decaying term, a variance term, a bias term, and a tracking error term. Each term is individually bounded. The decay rate of $\mathbb{E}\|z_t\|_2^2$ and $\mathbb{E}\|\theta_t - \theta^*\|_2^2$ are then recursively substituted into the expression for the tracking error term. This process yields an upper bound on the training error of the order $\mathcal{O}(h(\sigma,\nu)^{1-\epsilon'})$. By combining each of these terms, we obtain the final bound for $\mathbb{E}\|\theta_t - \theta^*\|_2^2$ as stated in Theorem 5.3. □

5.2.5 Mini-Batch and Variance-Reduced TDC

We further introduce two popular variants of the TDC algorithm with linear function approximation. The first variant is the mini-batch TDC algorithm analyzed in Xu *et al.* [46]. Different from the standard TDC, mini-batch TDC uses a batch of M off-policy samples queried from the MDP to generate one update in each iteration, as shown in the following update rule.

(Mini-batch TDC):

$$\theta_{t+1} = \theta_t + \frac{\alpha_t}{M} \sum_{i=tM}^{(t+1)M-1} \rho_i(\delta_i(\theta_t)\phi_{s_i} - \gamma\phi_{s_{i+1}}\phi_{s_i}^\top \omega_t),$$
$$\omega_{t+1} = \omega_t + \frac{\beta_t}{M} \sum_{i=tM}^{(t+1)M-1} (\rho_i\delta_i(\theta_t)\phi_{s_i} - \phi_{s_i}\phi_{s_i}^\top \omega_t).$$

The mini-batch updates help substantially reduce the variance of the updates and therefore larger learning rates can be used. Moreover, in the analysis of its convergence, we do not need to add the extra projection steps as adopted in the standard TDC.

The second variant of the TDC algorithm with linear function approximation is the variance-reduced TDC (VRTDC) [27]. As used in the analysis of TDC, for the t-th sample $x_t = (s_t, a_t, r_t, s_{t+1})$ and the behavior policy π_b, we define the following shortcut notations:

$$A_t := \rho(s_t, a_t)\phi(s_t)(\gamma\phi(s_{t+1}) - \phi(s_t))^\top, \quad b_t := r_t\rho(s_t, a_t)\phi(s_t), \tag{5.13}$$
$$B_t := -\gamma\rho(s_t, a_t)\phi(s_{t+1})\phi(s_t)^\top, \quad C_t := -\phi(s_t)\phi(s_t)^\top,$$

where $\rho(s,a) := \frac{\pi(a|s)}{\pi_b(a|s)}$ is the importance sampling ratio. The VRTDC algorithm iterates through m outer-loops, each involving M inner-loops. In each m-th outer-loop, initial parameters $\theta_0^{(m)}, w_0^{(m)}$ are updated from $\tilde{\theta}^{(m-1)}, \tilde{w}^{(m-1)}$. After querying M independent samples, batch pseudo-gradients $\widetilde{G}^{(m)}, \widetilde{H}^{(m)}$ are computed for inner-loop updates using the SVRG scheme. The final parameters $\tilde{\theta}^{(m)}, \tilde{w}^{(m)}$ for each loop are the averaged parameters from the inner-loops. Projections are applied in updates to maintain parameters within predefined bounds, ensuring stability and convergence. We note that the projection operators are widely used in the literature, e.g., Bhandari *et al.* [4], Bubeck [9], Dalal *et al.* [11], [12], Kushner [21], Lacoste-Julien *et al.* [22], Xu *et al.* [48], and Zou *et al.* [49]. Throughout this section, we assume the radius R_θ, R_w of the projected Euclidean balls satisfy that $R_\theta \geq \max\{\|A\|\|b\|, \|\theta^*\|\}$, $R_w \geq 2\|C^{-1}\|\|A\|R_\theta$.

Since the TDC algorithm has the natural two time-scale structure, the design of VRTDC also considers the variance reduction on both parameters θ and ω:

$$\theta_{t+1}^{(m)} = \Pi_{R_\theta}\left[\theta_t^{(m)} + \alpha\left(G_t^{(m)}\left(\theta_t^{(m)}, w_t^{(m)}\right) - G_t^{(m)}\left(\tilde{\theta}^{(m-1)}, \tilde{w}^{(m-1)}\right) + \tilde{G}^{(m)}\right)\right], \tag{5.14}$$

$$w_{t+1}^{(m)} = \Pi_{R_w}\left[w_t^{(m)} + \beta\left(H_t^{(m)}\left(\theta_t^{(m)}, w_t^{(m)}\right) - H_t^{(m)}\left(\tilde{\theta}^{(m-1)}, \tilde{w}^{(m-1)}\right) + \tilde{H}^{(m)}\right)\right],$$

where for a batch of M samples B_m and

$$\tilde{G}^{(m)} = \frac{1}{M}\sum_{x\in B_m}\left(A_x\tilde{\theta}^{(m-1)} + b_x + B_x\tilde{w}^{(m-1)}\right),$$

$$\tilde{H}^{(m)} = \frac{1}{M}\sum_{x\in B_m}\left(A_x\tilde{\theta}^{(m-1)} + b_x + C_x\tilde{w}^{(m-1)}\right).$$

The convergence analysis of VRTDC relies on the same assumption as the TDC algorithm, Assumption 5.4 and Assumption 5.5. We re-state its main result as follows:

Theorem 5.4. Suppose Assumption 5.4 and Assumption 5.5 hold. For the VRTDC algorithm with Markovian samples, there exists learning rates α, β with $\beta = \mathcal{O}(\alpha^{\frac{2}{3}})$ and the batch size M such that the output of the algorithm satisfies

$$\mathbb{E}\|\tilde{\theta}^{(m)} - \theta^*\|^2 \le \mathcal{O}(D^m + M^{-1} + \beta^2),$$

where $D \in (0,1)$ for the given selected learning rates and batch size.

Proof Sketch. The proof consists of the following three key steps: First, we develop the *preliminary bound for* $\sum_{t=0}^{M-1}\|\theta_t^{(m)} - \theta^*\|^2$. Denote $\mathbb{E}_{\theta_0^{(m)}}$ as the expectation conditional on $\theta_0^{(m)}$. Let $\mathcal{O}(f(\alpha, M))$ represent some variable $C \cdot f(\alpha, M)$ for some constant C. Then there exists learning rates α, β, and the batch size M such that

$$\begin{aligned}\mathbb{E}_{\theta_0^{(m)}}\sum_{t=0}^{M-1}\|\theta_t^{(m)} - \theta^*\|^2 \le &\mathcal{O}(\frac{1}{\alpha} + \alpha M)\|\tilde{\theta}^{(m-1)} - \theta^*\|^2 + \mathcal{O}(1)\\ &+ \mathcal{O}(1)\mathbb{E}_{\theta_0^{(m)}}\|z_t^{(m)}\|^2 + \mathcal{O}(\alpha M)\mathbb{E}_{\theta_0^{(m)}}\|\tilde{z}^{(m-1)}\|^2,\end{aligned}$$

This step bounds $\sum_{t=0}^{M-1}\|\theta_t^{(m)} - \theta^*\|^2$ in terms of $\sum_{t=0}^{M-1}\|z_t^{(m)}\|^2$, $\|\tilde{z}^{(m-1)}\|^2$, and $\|\tilde{\theta}^{(m-1)} - \theta^*\|^2$.

Second, we develop the *preliminary bound for* $\sum_{t=0}^{M-1}\|z_t^{(m)}\|^2$.

$$\begin{aligned}\sum_{t=0}^{M-1} \mathbb{E}_{\theta_0^{(m)}} \|z_t^{(m)}\|^2 \leq & \mathcal{O}(\frac{1}{\beta} + \beta M + \frac{\alpha^2}{\beta^2} M) \mathbb{E}_{\theta_0^{(m)}} \|\tilde{z}^{(m-1)}\|^2 + \mathcal{O}(1) \\ & + \mathcal{O}\Big(\beta + \frac{\alpha^2}{\beta^2}\Big) \Big(\sum_{t=0}^{M-1} \mathbb{E}_{\theta_0^{(m)}} \|\theta_t^{(m)} - \theta^*\|^2 \\ & + M \mathbb{E}_{\theta_0^{(m)}} \|\tilde{\theta}^{(m-1)} - \theta^*\|^2 \Big).\end{aligned}$$

This step bounds $\sum_{t=0}^{M-1} \|z_t^{(m)}\|^2$ in terms of $\sum_{t=0}^{M-1} \|\theta_t^{(m)} - \theta^*\|^2$, $\|\tilde{z}^{(m-1)}\|^2$, and $\|\tilde{\theta}^{(m-1)} - \theta^*\|^2$. Plugging it into the preliminary bound of $\sum_{t=0}^{M-1} \|\theta_t^{(m)} - \theta^*\|^2$, we obtain an upper bound of $\sum_{t=0}^{M-1} \|\theta_t^{(m)} - \theta^*\|^2$ in terms of $\|\tilde{z}^{(m-1)}\|^2$, and $\|\tilde{\theta}^{(m-1)} - \theta^*\|^2$.

Lastly, we develop the final *non-asymptotic bound for* $\|\tilde{z}^m\|^2$.

$$\mathbb{E}\|\tilde{z}^{(m)}\|^2 \leq \Big(\frac{1}{M\beta} \cdot \frac{12}{\lambda_C}\Big)^m \mathbb{E}\|\tilde{z}^{(0)}\|^2 + \mathcal{O}(\beta + \frac{\alpha^2}{\beta^2} + \frac{1}{M}).$$

By plugging this bound into the previous upper bounds, we will obtain a relation between $\mathbb{E}\|\tilde{\theta}^{(m)} - \theta^*\|$ and $\mathbb{E}\|\tilde{\theta}^{(m-1)} - \theta^*\|$. Recursively telescoping this inequality leads to our final result.

□

References

[1] D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio, "An actor-critic algorithm for sequence prediction," in *Proc. International Conference on Learning Representations*, 2017.

[2] L. Baird, "Residual algorithms: Reinforcement learning with function approximation," in *Machine Learning Proceedings 1995*, Elsevier, 1995, pp. 30–37.

[3] A. Benveniste, P. Priouret, and M. Métivier, *Adaptive Algorithms and Stochastic Approximations*. Springer-Verlag, 1990.

[4] J. Bhandari, D. Russo, and R. Singal, "A finite time analysis of temporal difference learning with linear function approximation," *arXiv preprint arXiv:1806.02450*, 2018.

[5] V. S. Borkar, *Stochastic approximation: a dynamical systems viewpoint*, vol. 48. Springer, 2009.

[6] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*, Springer, pp. 177–186, 2010.

[7] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, *Openai gym*, 2016. eprint: arXiv:1606.01540.

[8] G. Brunner, O. Richter, Y. Wang, and R. Wattenhofer, "Teaching a machine to read maps with deep reinforcement learning," in *Proc. Association for the Advancement of Artificial Intelligence (AAAI*, Nov. 2017.

[9] S. Bubeck, "Convex optimization: Algorithms and complexity," *Foundations and Trends® in Machine Learning*, vol. 8, no. 3-4, 2015, pp. 231–357.

[10] X. Chen, S. Li, H. Li, S. Jiang, Y. Qi, and L. Song, "Generative adversarial user model for reinforcement learning based recommendation system," in *Proc. International Conference on Machine Learning*, vol. 97, pp. 1052–1061, 2019.

[11] G. Dalal, B. Szorenyi, G. Thoppe, and S. Mannor, "Finite sample analysis of two-timescale stochastic approximation with applications to reinforcement learning," in *Proc. Conference on Learning Theory (COLT)*, 2018.

[12] G. Dalal, B. Szörényi, G. Thoppe, and S. Mannor, "Finite sample analysis of two-timescale stochastic approximation with applications to reinforcement learning," *Proceedings of Machine Learning Research*, vol. 75, 2018, pp. 1–35.

[13] A. Defazio, F. Bach, and S. Lacoste-Julien, "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1646–1654, 2014.

[14] M. P. Deisenroth, G. Neumann, and J. Peters, *A Survey on Policy Search for Robotics*. 2013.

[15] R. M. Gower, N. Loizou, X. Qian, A. Sailanbayev, E. Shulgin, and P. Richtárik, "Sgd: General analysis and improved rates," in *International conference on machine learning*, PMLR, pp. 5200–5209, 2019.

[16] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in neural information processing systems*, pp. 315–323, 2013.

[17] M. Kaledin, E. Moulines, A. Naumov, V. Tadic, and H.-T. Wai, "Finite time analysis of linear two-timescale stochastic approximation with markovian noise," *arXiv:2002.01268*, 2020.

[18] J. Kober Jens and Peters, "Reinforcement learning in robotics: A survey," in pp. 9–67, Springer, 2014.

[19] N. Korda and P. La, "On TD (0) with function approximation: Concentration bounds and a centered variant with exponential convergence," in *Proc. International Conference on Machine Learning (ICML)*, pp. 626–634, 2015.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[21] H. Kushner, "Stochastic approximation: A survey," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 1, 2010, pp. 87–96.

[22] S. Lacoste-Julien, M. Schmidt, and F. Bach, "A simpler approach to obtaining an $O(1/t)$ convergence rate for the projected stochastic subgradient method," *arXiv preprint arXiv:1212.2002*, 2012.

[23] J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley, and J. Gao, "Deep reinforcement learning for dialogue generation," in *Proc. Conference on Empirical Methods in Natural Language Processing*, pp. 1192–1202, 2016.

[24] F. Liu, S. Li, L. Zhang, C. Zhou, R. Ye, Y. Wang, and J. Lu, "3DCNN-DQN-RNN: A deep reinforcement learning framework for semantic parsing of large-scale 3D point clouds," in *Proc. International Conference on Computer Vision (ICCV)*, pp. 5679–5688, 2017.

[25] S. Ma, Z. Chen, Y. Zhou, K. Ji, and Y. Liang, "Data sampling affects the complexity of online sgd over dependent data," in *Uncertainty in Artificial Intelligence*, PMLR, pp. 1296–1305, 2022.

[26] S. Ma, Z. Chen, Y. Zhou, and S. Zou, "Greedy-gq with variance reduction: Finite-time analysis and improved complexity," in *International Conference on Learning Representations*, 2020.

[27] S. Ma, Y. Zhou, and S. Zou, "Variance-reduced off-policy tdc learning: Non-asymptotic convergence analysis," *Advances in neural information processing systems*, vol. 33, 2020, pp. 14 796–14 806.

[28] H. Maei, C. Szepesvari, S. Bhatnagar, D. Precup, D. Silver, and R. S. Sutton, “Convergent temporal-difference learning with arbitrary smooth function approximation,” *Advances in neural information processing systems*, vol. 22, 2009.

[29] A. Mustafin, A. Olshevsky, and I. C. Paschalidis, “Closing the gap between svrg and td-svrg with gradient splitting,” *arXiv preprint arXiv:2211.16237*, 2022.

[30] D. P and Bertsekas, *Dynamic programming and optimal control*, vol. 1, 2. Athena scientific Belmont, MA, 1995.

[31] M. L. Puterman, *Markov Decision Processes*. Wiley, 1994.

[32] N. Sharma, S. Zhang, S. R. S. Venkata, F. Malandra, N. Mastronarde, and J. Chakareski, “Deep reinforcement learning for delay-sensitive lte downlink scheduling,” in *IEEE Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, IEEE, pp. 1–6, 2020.

[33] R. Srikant and L. Ying, “Finite-time error bounds for linear stochastic approximation andtd learning,” in *Proc. Conference on Learning Theory*, vol. 99, pp. 2803–2830, 25–28 Jun 2019.

[34] R. Srikant and L. Ying, “Finite-time error bounds for linear stochastic approximation andtd learning,” in *Conference on Learning Theory*, PMLR, pp. 2803–2830, 2019.

[35] T. Sun, D. Li, and B. Wang, “Finite-time analysis of adaptive temporal difference learning with deep neural networks,” *Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 19 592–19 604.

[36] R. S. Sutton, “Learning to predict by the methods of temporal differences,” *Machine Learning*, vol. 3, no. 1, 1988, pp. 9–44.

[37] R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora, “Fast gradient-descent methods for temporal-difference learning with linear function approximation,” in *Proc. International Conference on Machine Learning (ICML)*, pp. 993–1000, 2009.

[38] R. S. Sutton, C. Szepesvári, and H. R. Maei, “A convergent o(n) algorithm for off-policy temporal-difference learning with linear function approximation,” *In Proc. Advances in Neural Information Processing Systems (NIPS)*, vol. 21, no. 21, 2008, pp. 1609–1616.

[39] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction, Second Edition*. The MIT Press, Cambridge, Massachusetts, 2018.

[40] V. Tadić, "On the convergence of temporal-difference learning with linear function approximation," *Machine learning*, vol. 42, no. 3, 2001, pp. 241–267.

[41] H. Tian, I. C. Paschalidis, and A. Olshevsky, "On the performance of temporal difference learning with neural networks," *arXiv preprint arXiv:2312.05397*, 2023.

[42] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE transactions on automatic control*, vol. 42, no. 5, 1997, pp. 674–690.

[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[44] M. Wainwright, "Variance-reduced q-learning is minimax optimal," *arXiv:1906.04697*, Jun. 2019.

[45] Y. Wang, S. Zou, and Y. Zhou, "Non-asymptotic analysis for two time-scale tdc with general smooth function approximation," *Advances in Neural Information Processing Systems*, vol. 34, 2021, pp. 9747–9758.

[46] T. Xu and Y. Liang, "Sample complexity bounds for two timescale value-based reinforcement learning algorithms," in *Proc. International Conference on Artificial Intelligence and Statistics*, vol. 130, pp. 811–819, 13–15 Apr 2021.

[47] T. Xu, Z. Wang, Y. Zhou, and Y. Liang, "Reanalysis of variance reduced temporal difference learning," in *Proc. International Conference on Learning Representations (ICLR)*, 2020.

[48] T. Xu, S. Zou, and Y. Liang, "Two time-scale off-policy TD learning: Non-asymptotic analysis over Markovian samples," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pp. 10 633–10 643, 2019.

[49] S. Zou, T. Xu, and Y. Liang, "Finite-sample analysis for SARSA with linear function approximation," in *Advances in Neural Information Processing Systems*, pp. 8665–8675, 2019.