Online Bootstrap Inference For Policy Evaluation In Reinforcement Learning

Pratik Ramprasad⁴, Yuantong Li³, Zhuoran Yang⁵, Zhaoran Wang⁶, Will Wei Sun⁷, and Guang Cheng⁸

⁴Department of Statistics, Purdue University, West Lafayette, IN; ³Department of Statistics, UCLA, Los Angeles, CA; ⁵Department of Statistics and Data Science, Yale University, New Haven, CT; ⁶Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL; ⁷Krannert School of Management, Purdue University, West Lafayette, IN

ABSTRACT

The recent emergence of reinforcement learning (RL) has created a demand for robust statistical inference methods for the parameter estimates computed using these algorithms. Existing methods for inference in online learning are restricted to settings involving independently sampled observations, while inference methods in RL have so far been limited to the batch setting. The bootstrap is a flexible and efficient approach for statistical inference in online learning algorithms, but its efficacy in settings involving Markov noise, such as RL, has yet to be explored. In this article, we study the use of the online bootstrap method for inference in RL policy evaluation. In particular, we focus on the temporal difference (TD) learning and Gradient TD (GTD) learning algorithms, which are themselves special instances of linear stochastic approximation under Markov noise. The method is shown to be distributionally consistent for statistical inference in policy evaluation, and numerical experiments are included to demonstrate the effectiveness of this algorithm across a range of real RL environments. Supplementary materials for this article are available online.

1. Introduction

Reinforcement learning (RL) has achieved phenomenal success in diverse fields, such as robotics (Gu et al. 2017), video games (Mnih et al. 2015), autonomous driving (Sallab et al. 2017), precision medicine (Parekh and Jacobs 2019), ride-sharing (Xu et al. 2018), and recommendation systems (Chen et al. 2019). A fundamental task in RL is policy evaluation, where the goal is to estimate the value function associated with a given policy from a trajectory of dependent observations. These observations may be sequentially generated either by the same policy (on-policy), or by an unknown behavior policy (off-policy). Standard algorithms used to perform on-policy and off-policy tasks include temporal difference (TD) learning (Sutton 1988) and gradient temporal difference (GTD) learning (Sutton et al. 2009), respectively. Both are instances of linear stochastic approximation.

While RL has proven to be remarkably successful in various applications, there are still many challenges in real-world systems that prevent it from being applied at scale in practice (Dulac-Arnold et al. 2021). A primary bottleneck in real applications is that environmental interactions are prohibitively expensive. Offline RL (Levine et al. 2020) attempts to address this by training and evaluating multiple policies using pre-existing datasets collected using a single policy, thereby circumventing the need for additional environment interactions. Unfortunately, off-policy value estimates can be challenging in cases where the trained policy is substantially different from the behavior policy, that is, the policy that was used to collect the data. In such cases, it is important to evaluate policies in an online setting prior to final deployment. Moreover, in many real applications, we are often interested in obtaining not just the point estimate of the value function, but also a measure of the statistical uncertainty associated with the estimate. For example, online randomized experiments, for example, A/B testing, have been widely conducted by technological/pharmaceutical companies to compare a new product with an old one. Recent studies (Li et al. 2021; Shi et al. 2021, 2022) have used various online updating methods to form sequential testing procedures. In these online evaluation tasks, it is important to quantify the uncertainty of the point estimate for constructing a valid hypothesis testing. In recommender systems, a new policy is typically tested on a small fraction of user traffic in an online fashion prior to deployment. Providing a confidence interval of the predicted value estimation can help make a recommendation with more confidence because an unstable recommendation can potentially reduce users’ trust in the system (Adomavicius and Zhang 2012; Chen et al. 2019). Similarly, in autonomous driving, policies must be evaluated on real test tracks prior to real-world application and it is often risky to run a policy without a statistically sound estimate of its quality.

Existing inference methods in RL mainly focus on off-policy evaluation using batch updates, which are often computationally inefficient in sequential data scenarios. This motivates the development of online inferential tools for policy evaluation in RL.

Our Contributions

In this work, we present a fully online multiplier-type bootstrap algorithm that allows for statistical inference in RL policy eval-
ution. To the best of our knowledge, this is the first online uncertainty quantification method for sequential decision making under Markovian noise. Since our method is designed for the general framework of stochastic approximation, it applies to both the on-policy and off-policy evaluation tasks, and also to other online learning problems such as stochastic gradient descent (SGD) under Markov noise.

From a theoretical standpoint, our main contribution is to establish the distributional consistency of our bootstrap estimator within the Markovian noise setting. Existing inference methods for online learning (Fang, Xu, and Yang 2018; Chen et al. 2020) have mainly focused on the iid noise setting, which allows for asymptotic analysis using the martingale properties of the residual noise in the stochastic approximation update. In the present work, the Markovian noise assumption precludes the direct use of martingale central limit theory to characterize the asymptotic properties of our estimator, and necessitates a novel combination of techniques from the Markov Chain Monte-Carlo (MCMC), stochastic approximation, and RL finite-sample analysis literature. Notably, our theoretical results hold under standard assumptions from the RL literature, and do not require any kind of projection step in the stochastic approximation algorithm, despite the presence of Markovian noise.

Our numerical experiments demonstrate that the algorithm performs on par with the vanilla (offline) bootstrap method across a range of environments, but with substantial savings in terms of data storage and computational cost. This is especially pertinent to computationally demanding algorithms such as deep Q-learning (Mnih et al. 2015), where uncertainty quantification using the standard bootstrap method is often unfeasible due to the massive computational and storage costs involved.

To summarize, our contributions are 3-fold. First, we develop a fully online inference method for linear stochastic approximation under Markov noise, with applications to both on-policy (TD) and off-policy (GTD) reinforcement learning algorithms. Second, we prove that the confidence intervals constructed using our bootstrap algorithm are asymptotically valid. Third, we demonstrate the efficacy of this method through a number of numerical experiments, including on-policy evaluation with linear TD learning (Sutton and Barto 2018), deep Q-learning (Mnih et al. 2015), and off-policy evaluation with GTD learning (Sutton, Szepesvári, and Maei 2008; Sutton et al. 2009).

**Related Work**

There has recently been a growth of interest in developing inferential tools for RL and other online learning algorithms such as bandits and SGD. Li et al. (2018) proposed a statistical inference method for M-estimation problems based on fixed step-size SGD. Chen et al. (2020) derived two kinds of estimators for the asymptotic variance of the averaged SGD parameter estimate. Fang, Xu, and Yang (2018) proposed an online estimator for SGD based on randomly perturbing the parameter estimates at each time step. Chen, Lu, and Song (2021) developed an SGD-based algorithm for online decision making, derived an inverse probability weighted value estimator to estimate the optimal value, and proposed plug-in estimators to estimate the variance of the parameters. While all of these methods are well suited to cases where the data is generated by an iid sampling procedure, they are unable to account for the underlying dependence structure of the observations generated by RL algorithms.

A closely related field of study is that of dynamic treatment regimes (DTR). It generalizes personalized medicine to time-varying treatment settings where treatments are sequentially adapted to a patient’s temporal state. There are similarities to the online learning methods studied here, and recent works have explored the application of RL algorithms such as Q-learning to this domain (Chakraborty and Murphy 2014), with an emphasis on quantifying the statistical uncertainty associated with their estimators. Ertefaie and Strawderman (2018) develop an estimation procedure for the optimal DTR under an indefinite time period and derive associated large-sample results. Luckett et al. (2019) proposed a new RL method for estimating an optimal treatment regime applicable to the mobile health domain with an infinite time horizon, and established the consistency and asymptotic normality of their estimators under relevant assumptions. These works focus on the estimation of the optimal policy, while ours deals with the inference of the value function for a given policy.

Within the RL and bandit settings, there are a number of recent works that consider the problem of uncertainty quantification of the policy’s parameter estimates or value function, focusing mainly on the off-policy setting. Zhang, Janson, and Murphy (2021) showed how M-estimation can be extended to provide inferential methods for data collected with bandit algorithms. Kuzborskij et al. (2021) proposed a method for confidence interval estimation based on the Efron-Stein tail inequality within the off-policy contextual bandit setting. Both these methods apply to the bandit setting, where there is no sequential dependence in state transitions. Within the off-policy RL setting, Dai et al. (2020) proposed a method for computing confidence intervals for a target policy’s value based on an optimization formulation of the value estimation problem. Jiang and Huang (2020) derive minimax value intervals for off-policy evaluation that satisfy a certain double robustness criterion. However, both papers assume a generative model, where observation tuples are sampled independently from the stationary distribution of the underlying Markov decision process. Shi et al. (2021) proposed an inference method for the state-action value (Q) function via sieve methods to approximate the Q-function. This is an offline method that directly computes the value estimates using batch updates. On the other hand, ours is a fully online inference method.

A number of recent works in the RL literature (Bhansali, Russo, and Singal 2018; Srikanth and Ying 2019; Xu et al. 2020; Kaledin et al. 2020) have focused on establishing finite sample bounds for the value estimates in TD learning and related algorithms. While these methods provide tight non-asymptotic bounds for the value function under standard assumptions, they do not allow for statistical inference of the estimates. By contrast, the distributional consistency guaranteed by our method allows for the construction of asymptotically exact confidence intervals for the value estimates.

Finally, our online bootstrap algorithm is related to a few recent works that have applied bootstrapping to the bandit and RL settings in various contexts. Wang et al. (2020) proposed a
perturbation-based bootstrap exploration method in the bandit setting. Hao et al. (2019) used multiplier bootstrap to estimate the upper confidence bound for exploration in the bandit settings. Within the RL setting, White and White (2010) used the moving block bootstrap method to compute confidence intervals for value estimates in continuous Markov decision processes. Hanna, Stone, and Niekum (2017) presented two model-based bootstrap methods to compute confidence bounds for off-policy value estimates. Hao et al. (2021) proposed a subsampled bootstrap method for the statistical inference of value estimates computed using the fitted Q-evaluation algorithm. All of these methods require access to the entire dataset in order to carry out the resampling procedure and are therefore only applicable to the batch RL setting. In contrast, the method proposed here computes the bootstrap estimates in an online manner and does not require storage of past observations.

The rest of this article is organized as follows: In Section 2, we introduce the linear stochastic approximation algorithm and provide some relevant background on the RL algorithms. In Section 3, we present the online bootstrap algorithm. In Section 4, we discuss our theoretical results. In Section 5, we present some numerical simulations that demonstrate the efficacy of the algorithm in various settings, including on-policy TD learning in the FrozenLake RL environment, deep-Q learning in the Atari Pong RL environment, and off-policy GTD learning in a simulated MDP setting and a real healthcare setting. Finally, in Section 6, we summarize our work and discuss some interesting future directions.

2. Background

2.1. Linear Stochastic Approximation Under Markov Noise

Stochastic approximation is a classic algorithm with a long history in optimization (Robbins and Monro 1951). In its linear form, the algorithm is designed to solve the equation \( A \theta = b \), where \( A \in \mathbb{R}^{d \times d} \) and \( b \in \mathbb{R}^d \) are unknown deterministic quantities, and \( \theta \in \Theta \subseteq \mathbb{R} \) is the parameter of interest. In the present setting, we are given a sequence of observations of the form \( \{(A(X_t), b(X_t))\}_{t \geq 1} \), where \( \{X_t\}_{t \geq 1} \) is an ergodic Markov chain with state space \( \mathcal{X} \) and stationary distribution \( \mu \), and \( A : \mathcal{X} \to \mathbb{R}^{d \times d} \) and \( b : \mathcal{X} \to \mathbb{R}^d \) are matrix and vector-valued functions defined on the state space \( \mathcal{X} \), whose expectations under the stationary distribution \( \mu \) are \( A \) and \( b \), respectively.

The update step for this algorithm is given by

\[
\theta_{t+1} = \theta_t + \alpha_{t+1} (A(X_{t+1}) \theta_t - b(X_{t+1})),
\]

where \( \theta_t \in \Theta \) is the stochastic approximation iterate that is, the current estimate of \( \theta_t \), and \( \{\alpha_t\}_{t \geq 1} \) is a sequence of polynomially decaying step-sizes, that is, \( \alpha_t = \alpha_0 / t^\rho \), for some \( \alpha_0 > 0 \) and learning rate \( \rho \in \left( \frac{1}{2}, 1 \right) \).

Our algorithm applies not to the iterate \( \theta_t \) itself, but to the averaged iterate \( \bar{\theta}_t = \frac{1}{T} \sum_{t=1}^{T} \theta_t \). This averaging scheme is referred to as Polyak-Ruppert averaging, after Ruppert (1988) and Polyak and Juditsky (1992), who established the asymptotic normality of \( \bar{\theta}_t \) for strongly convex objective functions under Martingale noise.

2.2. Policy Evaluation In Reinforcement Learning

In this section, we briefly review some background theory on Markov reward processes and policy evaluation, and describe the temporal difference (TD) learning and Gradient TD (GTD) learning algorithms as instances of the linear stochastic approximation algorithm with Markov noise in (2.1). In addition to these, recent studies (Mou et al. 2021; Durmus et al. 2021; Chen et al. 2021) show that other TD variates like n-step TD and TD(\( \lambda \)) are also special cases of Markovian linear stochastic approximation algorithms, which enables their studies on the finite-sample convergence guarantees of these RL algorithms. Our setting can be viewed as a complementary to the martingale noise setting considered in offline policy evaluation problems (Luckett et al. 2019; Shi et al. 2021). These methods do not allow for an online update but instead consider a batch update or an offline update.

Markov Reward Processes

A Markov Decision Process (MDP) is denoted as \( M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma), \) where \( \mathcal{S} \) is the state space, \( \mathcal{A} \) is a finite set of actions, \( \mathcal{P} \) is the transition kernel, \( \mathcal{R} \) is the reward function, and \( \gamma \in (0, 1) \) is a discount factor.

A stationary policy \( \pi \) maps a state \( s \in \mathcal{S} \) to a probability space via the distribution \( \pi(\cdot|s) \). At time step \( t \), suppose the learner at state \( s_t \in \mathcal{S} \), following the policy \( \pi \), takes the action \( a_t \in \mathcal{A} \) with probability \( \pi(a_t|s_t) \). Then the transition kernel \( \mathcal{P}(s_{t+1}|s_t, a_t) \) determines the probability of being in the next state \( s_{t+1} \in \mathcal{S} \) at the next time step, and the reward \( r_{t+1} = \mathcal{R}(s_t, a_t, s_{t+1}) \), assumed to be bounded by \( r_{\text{max}} \), is obtained.

The stationary policy \( \pi \) and the MDP together induce a Markov Reward Process (MRP) \( M^\pi = (\mathcal{M}, \pi) \), with transition kernel \( \mathcal{P}^\pi(s'|s, a) \), \( \pi \)-expected reward function of the MRP is given by

\[
\mathcal{R}^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s, a) \mathcal{R}(s, a, s').
\]

The ergodicity of the Markov chain ensures the existence of a stationary state distribution \( \mu_\pi \) for the MRP over the state space under the stationary policy \( \pi \).

Value Functions

The value function associated with a policy \( \pi \), denoted as \( V^\pi : \mathcal{S} \to \mathbb{R} \), is the discounted sum of expected rewards from starting at a state and following policy \( \pi \):

\[
V^\pi(s) = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t \mathcal{R}^\pi(s_t) | s_0 = s \right], \quad s \in \mathcal{S},
\]

where the expectation is taken over the set of trajectories generated according to the transition kernel \( \mathcal{P}^\pi \).

The value function is the unique solution to the Bellman equation

\[
V^\pi(s) = \mathcal{R}^\pi(s) + \gamma \mathbb{E}_{s' \sim \mathcal{P}^\pi(\cdot|s)} \left[ V^\pi(s') \right], \quad s \in \mathcal{S}. \tag{2.2}
\]

We define the Bellman operator on the space of value functions as

\[
T^\pi V(s) = \mathcal{R}^\pi(s) + \gamma \mathbb{E}_{s' \sim \mathcal{P}^\pi(\cdot|s)} [V(s')],
\]

for any value function \( V : \mathcal{S} \to \mathbb{R} \). Then \( V^\pi \) is the unique fixed point of the operator \( T^\pi \).
**TD Learning With Linear Function Approximation**

TD learning is widely used for the estimation of the value function for a given policy. The classic version of this algorithm, now known as tabular TD learning (Sutton 1988), attempts to compute value estimates for every state in the state space. In modern RL applications, such an approach becomes infeasible, as real-world problems often involve very large state spaces. In those cases, a natural approach is to approximate the value function as \( V_\theta(s) = \sum_{i=1}^{\|S\|} \phi_i(s) \theta_i \), where \( \{\phi_i : S \rightarrow \mathbb{R}^{d_i}\} \) is a basis of functions, \( \phi(s) = [\phi_1(s), \ldots, \phi_{\|S\|}(s)]^T \), and \( \theta \in \mathbb{R}^d \) denotes the parameter vector. Then, given a sequence of observations of the form \((s_t, r_{t+1}, s_{t+1})\), the linear TD update is given by

\[
\theta_{t+1} \leftarrow \theta_t + \alpha_{t+1} \left( (\phi(s_t) - \gamma \phi(s_{t+1})^T \theta_t - r_{t+1}) \phi(s_t) \right).
\]

When the state space \( S \) is finite, the states may be enumerated as \( \{1, \ldots, |S|\} \), with the integer \( i \) corresponding to \( s^{(i)} \), the \( i \)th state in \( S \). We can then express the transition kernel \( P^\pi \) as the matrix \( P^\pi \in \mathbb{R}^{|S| \times |S|} \), with \( P^\pi_{ij} = P(s^{(i)})|s^{(j)} \rangle \), and the expected rewards at each state as the vector \( r^\pi \in \mathbb{R}^{|S|} \), with \( r^\pi_i = \mathbb{E}[R(s^{(i)}, a)|s^{(i)}] \).

Using this formulation, linear TD learning may be seen as an instance of the linear stochastic approximation update (2.1), solving the linear equation \( \bar{A} \theta = \bar{b} \), with

\[
\bar{A} = \Phi^T \Xi (I - \gamma P^\pi) \Phi, \quad \text{and} \quad \bar{b} = \Phi^T \Xi r^\pi,
\]

where \( \Phi \in \mathbb{R}^{|S| \times d} \) denotes the feature matrix, containing the features \( \phi(s^{(i)}) \), \( 1 \leq i \leq |S| \) in its rows, and \( \Xi \in \mathbb{R}^{|S| \times |S|} \) is a diagonal matrix with elements corresponding to the entries of the stationary distribution \( \mu_{\pi} \).

For samples \( X_t = (s_t, r_{t+1}, s_{t+1}) \) generated sequentially from the MRP \( \mathcal{M}^\pi \), we can run linear stochastic approximation using the quantities

\[
\hat{A}(X_t) = \phi(s_t)(\phi(s_t) - \gamma \phi(s_{t+1}))^T, \quad \text{and} \quad \hat{b}(X_t) = r_{t+1} \phi(s_t).
\]

**GTD Learning For Off-Policy Evaluation**

In off-policy evaluation, the goal is to estimate the value of a target policy \( \pi_t \), given a set of observations generated by a behavior policy \( \pi_b \). This is an important problem in RL, as it enables us to evaluate several policies using data generated by a different behavior policy.

Unlike the on-policy setting, the traditional TD learning is no longer feasible due to its convergence issue in the off-policy setting (Sutton and Barto 2018). On the other hand, Gradient TD (GTD) algorithms (Sutton, Szepesvári, and Maei 2008; Sutton et al. 2009) are guaranteed to converge even in the off-policy setting.

The algorithm uses a form of importance sampling to correct for the discrepancy between the target and behavior policy. This is done by scaling the updates by an importance sampling ratio

\[
\rho_t = \frac{\pi_t(a_t|s_t)}{\pi_b(a_t|s_t)}.
\]

There are two variants of this algorithm, GTD1 and GTD2, which seek to minimize the norm of the Expected TD Update (NEU), and the Mean-Square Projected Bellman Error (MSPE), respectively. These loss functions have a similar structure, and can be unified as

\[
J(\theta) = \|\Phi^T \Xi (r^\pi + \gamma P^\pi \tilde{\theta}_0 - \tilde{\theta}_0)\|_M^{-2},
\]

where \( M \in \mathbb{R}^{d \times d} \) is the identity under NEU, and \( M = \Phi^T \Xi \Phi \) under MSPE.

In order to minimize this loss function, a pseudo-stochastic gradient method was proposed, involving two simultaneous updates:

\[
y_{t+1} = y_t + \alpha_{t+1} \left( b_t + A_t \theta_t - M_t y_t \right),
\]

\[
\theta_{t+1} = \theta_t + \alpha_{t+1} A_t^T y_t,
\]

where \( b_t = \rho_t r_{t+1} \phi(s_t) \) and \( A_t = \rho_t \phi(s_t)(\phi(s_t) - \gamma \phi(s_{t+1}))^T \) are unbiased estimators for \( \tilde{b} = \Phi^T \Xi r^\pi \) and \( \hat{A} = \Phi^T \Xi (I - \gamma P^\pi) \Phi \), respectively. Similarly, \( M_t \) is an unbiased estimator for \( M \), with \( M_t = I_d \) under NEU and \( M_t = \phi(s_t)\phi(s_t)^T \) under MSPE.

These two steps can be combined into a single linear stochastic approximation update solving the linear equation \( \bar{A} \Theta = \bar{b} \), where

\[
\bar{A} = \begin{pmatrix} 0 & -A^T \\ M \end{pmatrix}, \quad \bar{b} = \begin{pmatrix} 0 \\ b \end{pmatrix}, \quad \text{and} \quad \Theta = \begin{pmatrix} \theta \\ y \end{pmatrix}.
\]

As in the case of TD learning with linear function approximation, for samples generated sequentially under the behavior policy \( \pi_b \), we denote the observed tuples as \( X_t = (s_t, r_{t+1}, s_{t+1}) \). Then we can run linear stochastic approximation using the quantities

\[
\tilde{A}(X_t) = \begin{pmatrix} 0 \\ A_t \end{pmatrix}, \quad \tilde{b}(X_t) = \begin{pmatrix} 0 \\ b_t \end{pmatrix}, \quad \text{and} \quad \Theta_t = \begin{pmatrix} \theta_t \\ y_t \end{pmatrix}.
\]

**3. Method**

**3.1. Online Bootstrap Algorithm**

Given a dataset \( \{(\tilde{A}(X_i), \tilde{b}(X_i))\}_{i \geq 1} \) of sequentially generated observations, the Polyak-Ruppert averaged iterate estimates the parameter \( \theta_t \) as \( \tilde{\theta}_t = \frac{1}{t} \sum_{i=1}^{t} \theta_i \), where \( \theta_i \) is defined in (2.1). Under the assumptions listed in Section 4, \( \tilde{\theta}_t \) is a consistent estimator of \( \theta_{\pi} \), and its distribution is asymptotically Gaussian with mean \( \theta_{\pi} \) and a certain covariance matrix. The estimation of this asymptotic covariance is crucial for statistical inference, but there is presently no straightforward way to estimate it in the presence of Markov noise. This motivates the development of the online bootstrap method.

The online bootstrap method is based on the following perturbed stochastic approximation update, which is performed in parallel to the main update (2.1):

\[
\tilde{\theta}_{t+1} = \tilde{\theta}_t + \alpha_{t+1} W_{t+1} (\tilde{A}(X_{t+1}) \tilde{\theta}_t - \tilde{b}(X_{t+1})),
\]

where \( \{W_t\}_{t \geq 1} \) is a bounded sequence of iid random variables with mean 1 and variance 1, with \( W_1 < W_{\text{max}} \), a.s. for some finite constant \( W_{\text{max}} > 0 \). The same notation is used here to distinguish the perturbed iterates from those of the standard update (2.1).

Let \( \hat{\theta}_t \) denote the averaged iterate of the sequence \( \{\tilde{\theta}_t\}_{t \geq 0} \). In Theorem 4.2 of the Section 4, we show that the distributions of \( \hat{\theta}_t \) and \( \tilde{\theta}(\hat{\theta}_t - \tilde{\theta}_t) \) are asymptotically equivalent. This is a fundamental result for the validity of our online bootstrap algorithm, as it enables us to conduct inference on the former distribution by using the latter as a proxy. The latter distribution may be approximated by bootstrapping \( B \) samples of \( \tilde{\theta}_t - \tilde{\theta}_t \).
each $b = 1, \ldots, B$, at time step $t + 1$, we update the perturbed SGD iterates $\hat{\theta}^{(b)}_t$ as follows:

$$
\hat{\theta}^{(b)}_{t+1} = \hat{\theta}^{(b)}_t + \alpha_{t+1} W_t^{(b)} (\tilde{A}(X_{t+1}) \hat{\theta}^{(b)}_t - b(X_{t+1})),
$$

where $W_t^{(b)}$ are iid random variates with mean one and variance one. The updates can be performed in a fully online manner, as they only rely on the latest available data point $X_{t+1}$. Furthermore, since all $B$ trajectories of perturbed iterates depend on a single trajectory of the Markov chain $\{X_t\}$, the iterates can be updated in parallel. Intuitively, the bootstrap method proposed here circumvents the issue of higher-order dependence in the SA iterates by enabling us to perform statistical inference using the cross-section of perturbed iterates generated at each time step, as opposed to using the highly dependent iterates generated by a single trajectory. Algorithm 1 presents the entire online update scheme.

### Algorithm 1: Online Bootstrap For Linear Stochastic Approximation

**Input**: Number of bootstrap samples $B$, Initial step size $\alpha_0 > 0$, Learning rate $\eta \in (\frac{1}{2}, 1)$, Initial estimates $\theta_0 = \hat{\theta}_0$, $b = 1, \ldots, B$

1. for $t = 0, 1, 2, \ldots$ do
2.   Observe $\tilde{A}(X_{t+1}), \tilde{b}(X_{t+1})$.
3.   Compute $\alpha_{t+1} = \alpha_0 \eta^{-(t+1)}$.
4.   Update $\tilde{\theta}_{t+1} \leftarrow \tilde{\theta}_t + \alpha_{t+1} (\tilde{A}(X_{t+1}) \tilde{\theta}_t - \tilde{b}(X_{t+1}))$.
5.   Update $\hat{\theta}_{t+1} \leftarrow \frac{1}{t+1} (\hat{\theta}_t + \tilde{\theta}_{t+1})$.
6. for $b = 1, 2, \ldots, B$ do
7.   Update $\hat{\theta}^{(b)}_{t+1} \leftarrow \hat{\theta}^{(b)}_t + \alpha_{t+1} W_{t+1}^{(b)} (\tilde{A}(X_{t+1}) \hat{\theta}^{(b)}_t - \tilde{b}(X_{t+1}))$.
8.   Update $\bar{\theta}^{(b)}_{t+1} \leftarrow \frac{1}{t+1} (\bar{\theta}^{(b)}_t + \hat{\theta}^{(b)}_{t+1})$.
9. end
10 end
11 **Output**: Bootstrap estimates $\hat{\theta}^{(b)}_{t+1}$ for $b = 1, 2, \ldots, B$.

### 3.2. Constructing Confidence Intervals

We use two approaches to construct confidence intervals using the bootstrap empirical distribution—the quantile and the standard error-based confidence intervals. Let $q_\delta$ denote the $\delta$th quantile of the empirical distribution $\hat{\theta}^{(b)}_t - \bar{\theta}_t$. Then the $(1 - \alpha)$ quantile-based confidence interval for $\theta$ is given by $(\bar{\theta}_t + q_{\alpha/2}, \bar{\theta}_t + q_{1-\alpha/2})$.

Similarly, let $\hat{\Sigma}$ denote the sample covariance matrix of the bootstrapped value estimates. Then the $(1 - \alpha)$ standard error-based confidence interval for the value estimate is given by

$$
\left( V_{\hat{\theta}_t}^\pi(v) + z_{\alpha/2} \sqrt{\hat{\Sigma}}(v), V_{\hat{\theta}_t}^\pi(v) + z_{1-\alpha/2} \sqrt{\hat{\Sigma}}(v) \right).
$$

### 4. Distributional Consistency Result

#### 4.1. Asymptotic Theory For Linear Stochastic Approximation Under Markov Noise

In this section, we list our assumptions and state some relevant results pertaining to the asymptotic theory of linear stochastic approximation.

(A1). The Markov chain $\{X_t\}$ is uniformly ergodic, with transition kernel $P$ and unique stationary distribution $\mu$.

Assumption (A1) is standard in the RL literature for the policy evaluation setting, especially in recent works on the finite-sample analysis of policy evaluation algorithms (Bhandari, Russo, and Singal 2018; Srikant and Ying 2019; Xu et al. 2020). It always holds for irreducible, aperiodic Markov chains (Levin, Peres, and Wilmer 2017).

As a direct consequence of (A1), there exist constants $M > 0$ and $\kappa \in (0, 1)$ such that

$$
\sup_{x \in \mathcal{X}} \| P^t(x, \cdot) - \mu \| \leq M \kappa^t,
$$

(4.1)
where $D^t(x, \cdot)$ denotes the $t$-step transition kernel, starting from state $x$ (see, e.g., Meyn and Tweedie (2009); Douc et al. (2018)). Equation (4.1) characterizes the rate at which the Markov chain $\{X_t\}$ approaches its stationary distribution $\mu$, when starting from an arbitrary initial distribution.

Next, we define conditions on the noisy observations:

**(A2).** (i) There exists a matrix $\tilde{A}$ and a vector $\tilde{b}$ such that
\[
\tilde{A} = \lim_{t \to \infty} \mathbb{E}[\tilde{A}(X_t)], \quad \text{and} \quad \tilde{b} = \lim_{t \to \infty} \mathbb{E}[\tilde{b}(X_t)].
\]
(ii) The matrix $\tilde{A}$ is full rank and Hurwitz, that is, all its eigenvalues have strictly negative real parts.
(iii) There exist constants $A_{\text{max}}$ and $b_{\text{max}}$ such that
\[
\sup_{x \in X} \|\tilde{A}(x)\|_F \leq A_{\text{max}}, \quad \sup_{x \in X} \|\tilde{b}(x)\|_2 \leq b_{\text{max}}.
\]

The conditions listed in (A2) are imposed in order to ensure convergence of the sequence of iterates $\{\theta_t\}$. (A2)(i) is self-explanatory, while (A2)(ii) is a standard assumption in the stochastic approximation literature that ensures the stability of the algorithm (Polyak and Juditsky 1992; Srikanth and Ying 2019; Chen et al. 2020). It is generally considered a reasonable assumption in the RL setting, both for TD learning (Bhandari, Russo, and Singal 2018; Hu and Syed 2019) and for GTD learning (Gupta, Srikanth, and Ying 2019). (A2)(iii) controls the behavior of the Markov noise. In the RL setting, it holds whenever the feature maps $\phi$ and the reward function $R$ are bounded (Sutton and Barto 2018). By (A2)(i) and (A2)(iii), we also have that $\tilde{A} \leq A_{\text{max}}$ and $\tilde{b} \leq b_{\text{max}}$.

By (A2), there exists a unique solution $\theta_* \in \Theta$ to the linear equation $A\theta = \tilde{b}$. Furthermore, we can now write (2.1) as
\[
\theta_{t+1} = \theta_t + \alpha_{t+1}(\tilde{A}\theta_t - \tilde{b}) + \alpha_{t+1} \epsilon_{t+1},
\]
where $\epsilon_{t+1} = (\tilde{A}(X_{t+1}) - \tilde{A})\theta_t - (\tilde{b}(X_{t+1}) - \tilde{b})$ is a residual noise term.

Our final assumption has to do with the step sizes $\{\alpha_t\}$:

**(A3).** The step sizes are of the form $\alpha_t = \alpha_0 / t^\gamma$, $t \geq 1$, where $\alpha_0 > 0$ and the learning rate $\gamma \in (\frac{1}{2}, 1)$.

Polynomially decaying step sizes are standard in the case of iterate averaging (Polyak and Juditsky 1992).

Under these conditions, we have the following almost sure rate of convergence result for the update (4.2):

**Proposition 4.1.** Suppose that (A1)-(A3) hold. Let $\eta \in (1/2, 1)$ be defined as in (A3), and let $\gamma \in (0, \eta - 1/2)$. Then the iterates of update (4.2) satisfy $\|\theta_t - \theta_*\|_2 = o(t^{-\gamma})$, a.s.

This result establishes the consistency of the LSA iterate $\theta_t$ in the Markov noise setting. Additionally, it ensures that the iterates are bounded within a compact set without the need for a projection scheme. A proof of the result is provided in the supplementary materials.

By Lemma A.5 of Liang (2010), we can split the noise term in (4.2) into three parts as
\[
\epsilon_t = \epsilon_1 + \nu_t + \zeta_t,
\]
where $\epsilon_t$ is a martingale difference sequence, that is, $\mathbb{E}[\epsilon_t | F_{t-1}] = 0$, where $F_t$ is the natural filtration associated with the Markov chain $\{X_t\}$, while $\nu_t$ and $\zeta_t$ are decaying residual noise terms.

**Proposition 4.2.** Assume conditions (A1)-(A3) hold. Then
\[
\sqrt{t}(\theta_t - \theta_*) \Rightarrow N(0, \tilde{A}^{-1}Q(\tilde{A}^{-1})^T),
\]
where $Q = \lim_{t \to \infty} \mathbb{E}[\epsilon_t^2]$, with $\epsilon_t$ defined as in (4.3).

Proposition 4.2 establishes a central limit theorem for the averaged iterate and provides an explicit form for the asymptotic variance. Under iid noise, the asymptotic variance may be estimated using a plug-in estimator (Chen, Lu, and Song 2021). However, in the present setting, we have $Q = \lim_{t \to \infty} \mathbb{E}[\epsilon_t^2]$, where $\epsilon_t$ is the martingale component of the Markov noise term $\epsilon_t$, as per the decomposition (4.3). To the best of our knowledge, there is no existing method to estimate $\epsilon_t$ by separating it from the other components of $\epsilon_t$. Our online bootstrap algorithm provides an efficient and theoretically sound way to estimate the distribution of $\theta_t$. The properties of its key component, the perturbed linear stochastic approximation, are discussed in the following section.

### 4.2. Perturbed Linear Stochastic Approximation

We now study the asymptotic behavior of the perturbed linear stochastic approximation update (3.1). In the standard case, the mean field here is $h(\theta) = \bar{A}\theta - \bar{b}$, while the observations are of the form $H(\theta, X_{t+1}) = W_{t+1}(\bar{A}(X_{t+1})\theta - \bar{b}(X_{t+1})$. By the independence and boundedness of $W_t$, $H(\theta, X_{t+1})$ is an unbiased estimator of $h(\theta)$ under the stationary distribution of $X_t$.

We may rewrite (3.1) in terms analogous to (4.2), as follows:
\[
\theta_{t+1} = \tilde{\theta}_t + \alpha_{t+1}(\tilde{A}\tilde{\theta}_t - \tilde{b}) + \alpha_{t+1} \epsilon_t,
\]
where $\epsilon_t = (W_{t+1}\tilde{A}(X_{t+1}) - \tilde{A})\tilde{\theta}_t - (W_{t+1}\tilde{b}(X_{t+1}) - \tilde{b})$ is the noise term.

Having established the almost sure rate of convergence for the standard LSA update (4.2), the result may be straightforwardly extended to the perturbed update (4.4):

**Proposition 4.3.** Suppose that (A1)-(A3) hold. Let $\eta \in (1/2, 1)$ be defined as in (A3), and let $\gamma \in (0, \eta - 1/2)$. Then the iterates of update (4.4) satisfy $\|\tilde{\theta}_t - \theta_*\|_2 = o(t^{-\gamma})$, a.s.

Next, we establish the theoretical validity of the online bootstrap algorithm of Section 3. For this, we need the following lemma:

**Lemma 4.1.** Assume that (A1)–(A3) hold. Then
\[
\sqrt{t}(\tilde{\theta}_t - \theta_*) = -\frac{1}{\sqrt{t}} \tilde{A}^{-1} \sum_{i=1}^{t} W_{t+1} (\tilde{A}(X_{t+1})\theta_* - \tilde{b}(X_{t+1})) + o_p(1).
\]

The proof is provided in the supplementary materials. Note that Lemma 4.1 also holds for the update (4.2), since (4.4)
reduces to (4.2) when $W_i \equiv 1$ for all $i$. Hence, we have
\[
\sqrt{t}(\hat{\theta}_t - \theta_s) = -\frac{1}{\sqrt{t}}\hat{A}^{-1}\sum_{i=1}^{t}(\hat{A}(X_{i+1})\theta_s - \tilde{b}(X_{i+1})) + o_p(1).
\] (4.6)

Subtracting (4.6) from (4.5), we get
\[
\sqrt{t}(\tilde{\theta}_t - \hat{\theta}_t) = -\frac{1}{\sqrt{t}}\hat{A}^{-1}\sum_{i=1}^{t}(W_{i+1} - 1)
(\hat{A}(X_{i+1})\theta_s - \tilde{b}(X_{i+1})) + o_p(1).
\] (4.7)

The final step is to establish the distributional consistency of the online bootstrap estimator in the Kolmogorov metric. Suppose that the data $D$ is generated under the probability space $(X', A, P_D)$, while the bootstrap weights $W = \{W_i\}_{i=1}^{t}$ are generated under an independent probability space $(\Omega, B, P_W)$. Let $P_W|D$ denote the conditional distribution given the observed data $D$.

**Theorem 4.2.** Assume that (A1)-(A3) hold. Then, as $B \to \infty$ and $t \to \infty$, we have
\[
\sup_{v \in \mathbb{R}^d} P_W|D\left(\sqrt{t}(\hat{\theta}_t - \hat{\theta}_t) \leq v\right) - P_D\left(\sqrt{t}(\tilde{\theta}_t - \hat{\theta}_t) \leq v\right) \to 0, \text{ in probability.}
\] (4.8)

The proof is provided in the supplementary materials. We note that the above result applies to the sequence of probability measures $P_W|D$, where the data size and the number of bootstrap samples grows to infinity. In practice, there is a bootstrap Monte-Carlo error associated with resampling from a finite number of bootstrap samples $B$. By choosing $B$ adequately large, the bootstrap Monte-Carlo error is generally ignored (DasGupta 2010).

**Theorem 4.2** establishes the theoretical foundation for using our online bootstrap estimator for statistical inference. The distributional consistency of the bootstrap estimator in terms of the Kolmogorov metric (4.8) enables us to construct asymptotically exact confidence intervals for the estimator $\tilde{\theta}$ or functions of the estimator, such as the value estimate in TD learning under linear function approximation.

5. Experiments

In this section, we evaluate the performance of the online bootstrap algorithm through numerical simulations across various settings. We construct two types of confidence intervals using the online bootstrap algorithm—namely, the quantile estimator and the standard error estimator, as defined in the previous section. All confidence intervals were constructed to have a nominal coverage of 95%. In all cases, we set the number of bootstrap samples $B$ to 200, the learning rate $\eta$ to 3/4, and the initial parameter $\theta_0$ to the zero vector. The random variates $W_t$ are sampled from the uniform distribution over the interval $(1 - 1/\sqrt{3}, 1 + 1/\sqrt{3})$, so they are bounded and have mean 1, variance 1.

While our results apply most naturally to the infinite-horizon (continual) setting due to our assumption of ergodicity and the existence of a unique stationary distribution, they can be applied to the finite-horizon (episodic) setting in a straightforward manner. To accomplish this, we simply concatenate the individual episodes to form a trajectory of infinite length, and perform inference on the parameters with respect to this infinite-horizon embedding of the MDP. Similar approach has also been applied in the experiments of other RL literature (Dai et al. 2020; Xu et al. 2020). The theoretical validity of such a concatenation procedure has been studied by Bojun (2020).

The main benchmark we use to measure our algorithm’s performance is the vanilla bootstrap, which is an offline method that requires the entire batch of samples for computation. Bootstrap-based inference is highly versatile with regard to the choice of policy evaluation algorithm, and is known to provide better second-order accuracy even when the asymptotic distribution is available (Hall 1992). It has already been studied in the RL literature in various contexts (White and White 2010; Hanna, Stone, and Niekum 2017; Hao et al. 2021), and naturally allows for a like-for-like comparison with our method. Therefore, we opt to use the vanilla bootstrap as our primary comparison method. In order to ensure a fair comparison, we use the same number of bootstrap samples, and within each sample, we use the same number of resampled observations as the size of the original dataset.

5.1. On-Policy Value Inference For FrozenLake RL Environment

Next, we consider the FrozenLake environment from OpenAI gym (Brockman et al. 2016). Here the RL agent controls the movement of a character in an $8 \times 8$ grid world. The starting point is the first tile of the grid, and the goal is to reach the end tile of the grid. The rest of the tiles are either walkable or absorbing states. A reward of 1 is awarded if the character reaches the target tile, and the reward for any other state transition is 0.

We use linear TD learning to estimate the value function associated with a near-optimal policy trained using Q-learning (Sutton and Barto 2018). For the compared vanilla (offline) bootstrap method, we choose to resample the observations by episodes, rather than by sample transitions, as suggested by Hao et al. (2021), as the sample transitions may fail to capture the sequential dependence in state transitions.

Figure 1(a) shows an example of the confidence intervals generated for the value estimate of the initial state, using both the online and offline bootstrap methods. The true value function for that state, computed analytically using the transition probability matrix, is included for reference. As in the previous experiment, we also examine the empirical coverage probabilities, in Figure 1(b). Both methods are seen to achieve the nominal coverage of 95% within approximately 1000 episodes.

Figure 2 shows the sensitivity of the widths of the CIs generated by the online bootstrap method with respect to the initial step size $\alpha_0$ and the learning rate $\eta$. Similarly, Figure 3 shows the sensitivity of the empirical coverage of the generated CIs with respect to the step size parameters. These figures demonstrate that the online bootstrap method is quite robust with respect to the step size, which is one of the only user-defined parameters in the algorithm (alongside the choice of distribution for the random perturbations $W_t$).
5.2. On-policy Value Inference For Atari Pong RL Environment

In this experiment, we consider the problem of deep Q-learning (Mnih et al. 2015) in the Atari Pong environment (Brockman et al. 2016). Here the agent controls a paddle in a game of pong, competing against a system paddle, as depicted in Figure 4(a). At the end of each round, a score of 1 is awarded when a paddle hits the ball past the other paddle. The game is won by the paddle that first reaches a score of 21.

In terms of the RL environment, a reward of 1 is awarded (deducted) when the agent wins (loses) a round. An episode ends when the game is won or lost. The total reward for the episode is the net score accumulated by the agent. Rewards are scaled by 0.1 before being used as inputs to the RL agent. The state inputs to the agent are the raw pixels (RGB images).
generated by the game engine. So each state is an array of shape (210, 160, 3). There are six discrete actions available to the agent at each step.

Our goal here is to learn a linear function approximation of the value function associated with a given policy. However, in this case, since the states are represented as three-way tensors, we cannot use these states as raw features. Instead, we use a neural network to transform the states into feature vectors. We first train the agent using a Deep Q Network (DQN) with the same configuration as in Mnih et al. (2013), with three convolutional layers, and two fully connected hidden layers. The input is the preprocessed state tensor, while the outputs are the Q-function estimates for each state-action pair associated with that state.

In order to apply our algorithm to this case, we drop the output layer of the pretrained DQN, so that the output of the resulting network is a 512-dimensional vector. These are the features we use for linear TD learning. We construct the true value function corresponding to the generated policy as the inner product of the state features and a randomly generated true parameter. The expected rewards at each state under the given policy are then computed using the Bellman equation (2.2). Observations are sequentially drawn according to the MDP under the generated policy, and we use this data to estimate the value of an $\epsilon$-greedy version of the generated policy, with $\epsilon = 0.2$.

Within this framework, we implement three off-policy evaluation methods - the online bootstrap, the offline (vanilla) bootstrap, and the SAVE estimator (Shi et al. 2021). We compare the methods in terms of their CI widths and empirical coverage probabilities (computed over 200 runs) for the value estimate of one of the states. Our online bootstrap method is applied in conjunction with GTD learning (Sutton et al. 2009) to obtain the point estimates. This an online method that updates the parameters sequentially. For the offline bootstrap method, to ensure a fair comparison, we use the same GTD estimator to obtain the point estimate, and use the vanilla bootstrap to compute the confidence intervals. A similar offline bootstrap method has been used by Hanna, Stone, and Niekum (2017) and Hao et al. (2021) for constructing their confidence intervals. The SAVE method is used here in conjunction with LSTD-Q (Lagoudakis 2003), that is, without the sieve basis functions. Note that LSTD-Q is a batch learning method, and therefore has different properties to GTD learning.

Figure 6 shows the results of our comparisons. Figure 6(a) compares the widths of the confidence intervals, while Figure 6(b) compares the empirical coverage probabilities over 200 simulated runs. The results show that the three methods perform roughly on par, although our online bootstrap method is much faster than the offline bootstrap, and, unlike the other
two methods, does not need to store the data. Since the SAVE method uses batch LSTD-Q for its point estimates, it is expected to reach the desired 0.95 coverage probability in fewer observations than the other two online GTD-based methods.

5.4. Off-Policy Value Inference For A Healthcare Application

Finally, we consider a real-world example from the healthcare domain, namely, the problem of treating sepsis in the intensive care unit. For this experiment, we use the septic management simulator by Oberst and Sontag (2019). It simulates the patient’s vital signs, such as the heart rate, blood pressure, oxygen concentration, and glucose levels. There are three treatment actions (antibiotics, vasopressors, and mechanical ventilation) which the RL agent chooses from at each time step. The reward is +1 if the patient is discharged and −1 if the patient reaches a critical state. These are both absorbing states. The reward is 0 for transitions to non-absorbing states.

As in the previous experiment, we train a near-optimal policy using Q-learning. We then generate a dataset using an $\epsilon$-greedy Q-policy, with $\epsilon = 0.05$. For the target policy, we use the optimal policy learned using Q-learning. To estimate the value function of the target policy in the off-policy setting, we use GTD learning with the Mean-Squared Projected Bellman Error (MSPBE) objective function (Sutton et al. 2009).

Since we do not have the true transition matrix, it is not possible to estimate coverage frequencies for our algorithm using the true value function. Figure 7(a) shows the empirical MSPBE, which is a proxy for the estimation error for the of the point estimates. Figure 7(b) shows the width of the confidence intervals computed using the quantile and standard error estimators. The estimation error decreases gradually over 20,000 episodes, and the CI widths for both estimators decrease correspondingly, reflecting the decreasing uncertainty in the estimates as more data becomes available.

6. Discussion and Future Work

In this article, we present a fully online bootstrap algorithm for statistical inference of policy evaluation in reinforcement learning. We establish its distributional consistency in terms of the underlying algorithm, linear stochastic approximation under Markov noise. Our experimental results suggest that the online bootstrap method is efficient and effective across a range of tasks, from linear SGD with Markov noise to off-policy value estimation with GTD learning. Next, we discuss a few additional topics and interesting future directions.
6.1. Nonstationary Behavior Policy

Our online inference method is built upon linear stochastic approximation, and focuses on two applications in RL: (a) on-policy evaluation with standard TD learning, where the target and behavior policies are the same, and (b) off-policy evaluation with GTD learning, where the target and behavior policies are different. An interesting additional application to consider would be the case where the behavior policy changes over time, as this would enable the estimation of the value of a target policy from observations generated under a nonstationary behavior policy. To our knowledge, all existing stochastic approximation-based off-policy evaluation algorithms (GTD learning: Sutton et al. (2009), Emphatic TD: Sutton, Mahmood, and White (2016), and related algorithms) require the behavior policy to be stationary in order to ensure theoretically valid point estimates. That said, if these point methods were proven to be convergent under a nonstationary behavior policy, our online bootstrap-based inference method should easily be adaptable to that setting, as it only requires knowledge of the importance sampling ratio at each time step.

6.2. Semi-parametric Efficiency

The main purpose of this article is to propose a provable online bootstrap inference method for the existing point estimators - TD learning and GTD learning, which are both special cases of linear stochastic approximation under Markov noise. In the context of iid observations, the asymptotic variance of the averaged iterate under this scheme is known to achieve the Cramer-Rao lower bound (Polyak and Juditsky 1992; Moulines and Bach 2011).

There are a few recent works studying the semi-parametric efficiency of reinforcement learning algorithms. For example, Ueno et al. (2011) proposed a generalized form of TD learning and studied its semi-parametric efficiency. Using this framework, they derived an optimal estimating function with the minimal asymptotic variance. In addition, a recent work by Kallus and Uehara (2021) established a lower bound for the asymptotic mean-squared error (MSE) of the Q-function estimate under a general function approximation scheme. They provide an explicit form for the approximating function that achieves the MSE lower bound. Their results require an assumption of transition sampling, that is, state-action-reward-state tuples are drawn independently from the generative model, rather than sequentially according to the trajectory of the MDP.

To our knowledge, there are currently no results for the efficiency of the linear stochastic approximation iterate under Markov noise. Since our focus is on the bootstrap method for uncertainty quantification, rather than the algorithm for point estimation, we leave the question of efficiency for linear stochastic approximation under Markov noise as interesting future work.

6.3. Extension to Nonlinear Function Approximation

In this work we mainly focus on the linear function approximation setting, as this is the fundamental function approximation scheme for policy evaluation algorithms in RL, serving as the basis for more elaborate approximation schemes. Here we briefly discuss two practical ways in which we can use the online bootstrap approach within the nonlinear function approximation setting.

The first approach is to transform the feature basis to another space using nonlinear basis expansion, and then use bootstrapping in conjunction with linear TD learning on the transformed basis. The basis expansion can be done via sieve basis functions (Shi et al. 2021) or neural networks to learn the representation of the feature space. In our Atari Deep Q-learning experiment of Section 5.2, the raw state features are high-dimensional 3-way tensors. Here, we applied linear TD learning on the features obtained from the last layer of the pretrained DQN to handle the nonlinear function approximation. This is similar to the approach used by Chung et al. (2019), who proposed a two-timescale network architecture that enables linear methods to learn values at the top layer, with a nonlinear representation learned at a slower timescale at the bottom layers.

The second approach is to directly apply our method to a nonlinear function approximation scheme. The general form of the stochastic approximation is

\[ \theta_{t+1} = \theta_t + \alpha_{t+1} H(\theta_t, X_{t+1}), \]  

where \( H(\theta, X) \) is a noisy observation of the mean field \( h(\theta) = \mathbb{E}[H(\theta, X)] \). Here, the goal is to estimate the root \( \theta^* \) of the

nonlinear equation \( h(\theta) = 0 \). Under some assumptions on the nonlinear function \( h \) and the iterates \( \{\theta_t\} \), Andrieu, Moulines, and Priouret (2005) proved the consistency of the \( \{\theta_t\} \), while Liang (2010) established a central limit theorem for the averaged iterate \( \bar{\theta}_t \). In practice, we can apply our online bootstrap procedure in Algorithm 1 to produce bootstrap estimates for the iterates \( \theta_t \) under the nonlinear stochastic approximation update (6.1). This allows us to perform online inference for nonlinear SGD under Markov noise, which extends the online inference results for nonlinear SGD under iid noise (Fang, Xu, and Yang 2018; Chen et al. 2020). Since the focus of this work is on linear function approximation, we leave a rigorous investigation of this nonlinear setting to future work.

### 6.4. Approximation Error and Model Misspecification

This article focuses on the linear function approximation of the value function. When the linear model assumption is violated, it is important to consider the model mis-specification issue in the analysis of RL algorithms with function approximation. Here we briefly discuss the implications of approximation error within the context of TD learning with linear function approximation. Our discussion starts from the least-false parameter in the linear space and then connects it with the point estimator from TD learning.

**Least-false parameter in the linear space:** We follow the notation defined in Section 2 of the article. Denote \( S \) as the state space, \( \Phi \) as the feature matrix. Let \( \Pi \) denote the projection operator onto the space spanned by the linear basis functions. Then, for a given policy \( \pi \) with a stationary distribution \( \mu \), we have

\[
\Pi V^\pi = \arg\min_{\theta} \mathbb{E}_{(s,\phi) \in \mathcal{S} \times \Phi} \left[ D_{v} V^\pi - V^\pi \right]_D,
\]

where \( D \) denotes the diagonal matrix with elements corresponding to the entries of the stationary distribution \( \mu \), and \( \|v\| = \sqrt{\sum_{d} v_d^2} \) denotes the norm under the stationary distribution. In other words, the projected value function \( \Pi V^\pi \) is the best approximation to the true value function \( V^\pi \) within the subspace spanned by the linear basis functions.

**Point estimator from TD learning:** Tsitsiklis and Van Roy (1997) showed that the limiting point \( \theta_\ast \) of the linear TD update is the unique solution to the projected Bellman equation \( V_\ast = \Pi T^\pi V_\ast \), where \( V_\ast = \Phi \theta_\ast \) is the value estimate corresponding to the parameter \( \theta_\ast \), while \( T^\pi \) denotes the Bellman operator under the policy \( \pi \). In other words, the limiting point \( \theta_\ast \) of the TD estimator can be seen as the global minimizer of the Mean-Squared Projected Bellman Error (MSPBE), that is,

\[
\theta_\ast = \arg\min_{\theta \in \mathbb{R}^d} \|V_\theta - \Pi T^\pi V_\theta\|_2^2.
\]

Our online bootstrap method provides a way to perform inference on the minimizer of the MSPBE, which is a quantity of interest in its own right (Sutton and Barto 2018).

**Connection:** Although the limiting point of the linear TD estimator is not the least-false parameter in the linear space, these two have a nice connection. Tsitsiklis and Van Roy (1997) showed that the value function \( V_\ast \), corresponding to the limiting point \( \theta_\ast \) satisfies

\[
\|V_\ast - V^\pi\|_D \leq \frac{1}{\sqrt{1 - \gamma^2}} \|\Pi V^\pi - V^\pi\|_D.
\]

Thus, the approximation error for the limiting point of the TD value estimator is bounded by a constant times the approximation error for the projected value function \( \Pi V^\pi \), which represents the best possible approximation in the span of \( \Phi \). So, while the confidence intervals generated by the online bootstrap method provide a coverage only for the value function \( V_\theta \) corresponding to the minimizer of the MSPBE, this value function itself comes with a competitive guarantee.

### Supplementary Materials

In the supplementary material, we provide detailed proofs for the lemmas and main theorems in Sections 4.1 and 4.2, as well as additional experiments.

### Acknowledgments

The authors thank the editor Professor Marina Vannucci, the associate editor, and three anonymous reviewers for their valuable comments and suggestions which led to a much improved article.

### Disclosure Statement

The authors report there are no competing interests to declare.

### Funding

Zhaoran Wang acknowledges National Science Foundation (Awards 2048075, 2008827, 2015568, 1934931), Simons Institute (Theory of Reinforcement Learning), Amazon, J.P. Morgan, and Two Sigma for their supports. Will Wei Sun’s research was partially supported by ONR grant N00014-18-1-2759. Guang Cheng acknowledges support from the National Science Foundation (NSF – SCALE MoDL (2134209)). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of National Science Foundation and Office of Naval Research.

### ORCID

Will Wei Sun  [http://orcid.org/0000-0002-8412-6430](http://orcid.org/0000-0002-8412-6430)

### References

Adomavicius, G., and Zhang, J. (2012), ”Stability of Recommendation Algorithms,” ACM Transactions on Information Systems (TOIS), 30, 1–31. [2901]

Andrieu, C., Moulines, E., and Priouret, P. (2005), ”Stability of Stochastic Approximation Under Verifiable Conditions,” SIAM Journal on Control and Optimization, 44, 283–312. [2912]

Bhandari, J., Russo, D., and Singal, R. (2018), ”A Finite Time Analysis of Temporal Difference Learning with Linear Function Approximation,” in Proceedings of the 31st Conference On Learning Theory, Volume 75 of Proceedings of Machine Learning Research, eds. S. Bubeck, V. Perchet, and P. Rigollet, pp. 1691–1692. PMLR. [2902,2905,2906]

Bojun, H. (2020), ”Steady State Analysis of Epidemic Reinforcement Learning,” in Advances in Neural Information Processing Systems (Vol. 33), eds. H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, pp. 9335–9345. Curran Associates, Inc. [2907]

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016), ”Openai gym,” arXiv preprint arXiv:1606.01540. [2907,2908]

Chakraborty, B., and Murphy, S. A. (2014), ”Dynamic Treatment Regimes,” Annual Review of Statistics and its Application, 1, 447–464. [2902]
Parekh, V. S., and Jacobs, M. A. (2019), “Deep Learning and Radiomics in Precision Medicine,” Expert Review of Precision Medicine and Drug Development, 4, 59–72. [2901]
Polyak, B. T., and Juditsky, A. B. (1992), “Acceleration of Stochastic Approximation by Averaging,” SIAM Journal on Control and Optimization, 30, 838–855. [2903,2906,2911]
Robbins, H., and Monro, S. (1951), “A Stochastic Approximation Method,” The Annals of Mathematical Statistics, 22, 400–407. [2903]
Ruppert, D. (1988), “Efficient Estimations from a Slowly Convergent Robbins-Monro Process,” Technical Report, Cornell University Operations Research and Industrial Engineering. [2903]
Sallab, A. E., Abdou, M., Perot, E., and Yogamani, S. (2017), “Deep Reinforcement Learning Framework for Autonomous Driving,” Electronic Imaging, 2017, 70–76. [2901]
Shi, C., Luo, S., Zhu, H., and Song, R. (2021), “An Online Sequential Test for Qualitative Treatment Effects,” Journal of Machine Learning Research, 22, 1–51. [2901]
Shi, C., Wang, X., Luo, S., Zhu, H., Ye, J., and Song, R. (2022), “Dynamic Causal Effects Evaluation in a/b Testing with a Reinforcement Learning Framework,” Journal of the American Statistical Association (just-accepted), 1–29, DOI: 10.1080/01621459.2022.2027776. [2901]
Srikant, R., and Ying, L. (2019), “Finite-Time Error Bounds for Linear Stochastic Approximation and TD Learning,” in Conference on Learning Theory, pp. 2803–2830. PMLR. [2902,2905,2906]
Sutton, R. S. (1988), “Learning to Predict by the Methods of Temporal Differences,” Machine Learning, 3, 9–44. [2901,2904]
Sutton, R. S., and Barto, A. G. (2018), Reinforcement Learning: An Introduction, Cambridge, MA: MIT press. [2902,2904,2906,2907,2912]
Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C., and Wiewiora, E. (2009), “Fast Gradient-Descent Methods for Temporal-Difference Learning with Linear Function Approximation,” in Proceedings of the 26th Annual International Conference on Machine Learning, pp. 993–1000. [2901,2902,2904,2909,2910,2911]
Sutton, R. S., Mahmood, A. R., and White, M. (2016), “An Emphatic Approach to the Problem of Off-policy Temporal-Difference Learning,” Journal of Machine Learning Research, 17, 1–29. [2911]
Sutton, R. S., Szepesvári, C., and Maei, H. R. (2008), “A Convergent o(n) Algorithm for Off-policy Temporal-Difference Learning with Linear Function Approximation,” in Proceedings of the 21st International Conference on Neural Information Processing Systems, NIPS’08, USA, pp. 1609–1616, Curran Associates Inc. [2902,2904]
Tsitsiklis, J. N., and Van Roy, B. (1997, May), “An Analysis of Temporal-Difference Learning with Function Approximation,” IEEE Transactions on Automatic Control, 42, 674–690. [2912]
Ueno, T., Maeda, S.-i., Kawanabe, M., and Ishii, S. (2011), “Generalized TD Learning,” Journal of Machine Learning Research, 12, 1977–2020. [2911]
Wang, C.-H., Yu, Y., Hao, B., and Cheng, G. (2020), “Residual Bootstrap Exploration for Bandit Algorithms,” arXiv preprint arXiv:2002.08436. [2902]
White, M., and White, A. (2010), “Interval Estimation for Reinforcement-Learning Algorithms in Continuous-State Domains,” in Advances in Neural Information Processing Systems (Vol. 23), eds. J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Curran Associates, Inc. [2903,2907]
Xu, T., Wang, Z., Zhou, Y., and Liang, Y. (2020), “Reanalysis of Variance Reduced Temporal Difference Learning,” in International Conference on Learning Representations. [2902,2905,2907]
Xu, Z., Li, Z., Guan, Q., Zhang, D., Li, Q., Nan, J., Liu, C., Bian, W., and Ye, J. (2018), “Large-Scale Order Dispatch in On-demand Ride-Hailing Platforms: A Learning and Planning Approach,” in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 905–913. [2901]
Zhang, K. W., Janson, L., and Murphy, S. A. (2021), “Statistical Inference with m-estimators on Bandit Data,” arXiv preprint arXiv:2104.14074. [2902]