Variance Reduction via Resampling and Experience Replay

Jiale Han, Xiaowu Dai*, Yuhua Zhu*

Department of Statistics and Data Science, UCLA
{jialehan, daix, yuhuazhu}@ucla.edu

Abstract

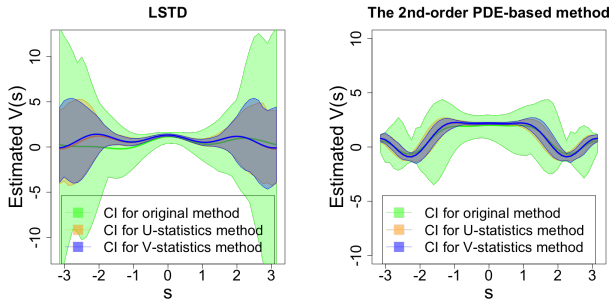
Experience replay is a foundational technique in reinforcement learning that enhances learning stability by storing past experiences in a replay buffer and reusing them during training. Despite its practical success, its theoretical properties remain underexplored. In this paper, we present a theoretical framework that models experience replay using resampled U- and V-statistics, providing rigorous variance reduction guarantees. We apply this framework to policy evaluation tasks using the Least-Squares Temporal Difference (LSTD) algorithm and a Partial Differential Equation (PDE)-based model-free algorithm, demonstrating significant improvements in stability and efficiency, particularly in data-scarce scenarios. Beyond policy evaluation, we extend the framework to kernel ridge regression, showing that the experience replay-based method reduces the computational cost from the traditional cubic time to quadratic time in the sample size, while also reducing variance. Extensive numerical experiments validate our theoretical findings, demonstrating the broad applicability and effectiveness of experience replay in diverse machine learning tasks.

Code — https://github.com/JialeHan22/Variance-Reduction-via-Resampling-and-Experience-Replay
Extended version — https://arxiv.org/pdf/2502.00520

1 Introduction

Experience replay is widely recognized for enhancing learning stability by storing past experiences in a memory buffer and reusing them during training (Lin 1992; Mnih et al. 2015). Rather than processing each experience only once, experience replay randomly samples batches of experiences to update learning targets, increasing sample efficiency and improving model performance. This approach has become a key component in modern reinforcement learning (RL), driving breakthroughs in applications such as Atari games (Mnih et al. 2015) and AlphaGo (Silver et al. 2016). However, despite its widespread success, the theoretical understanding of experience replay remains limited, often requiring extensive trial and error for effective application (Zhang and Sutton 2017; Fedus et al. 2020). To address this gap, we propose a theoretical framework that connects experience replay to resampled U- and V- statistics (Frees 1989; Shieh 1994). This framework establishes rigorous variance reduction guarantees, providing a deeper understanding of how experience replay enhances learning stability.

Building on prior work on U- and V- statistics (Zhou, Mentch, and Hooker 2021; Peng, Coleman, and Mentch 2022), which primarily focused on decision-tree-based methods like random forests, we extend this framework to encompass a broader class of learning functions. We derive the asymptotic variance of learned estimators, demonstrating that estimators employing experience replay achieve asymptotically lower variance compared to their original methods. To validate our framework, we analyze variance reduction through experience replay in two important machine-learning problems: policy evaluation in RL and supervised learning in reproducing kernel Hilbert space (RKHS).



(a) LSTD approach. (b) PDE-based approach.

Figure 1: Variance reduction achieved by experience replay in policy evaluation using two approaches. U- and V-statistics methods incorporate experience replay without and with replacement, respectively, into the original method. The solid lines represent the mean estimates, and the shaded areas denote the 95% confidence intervals (CIs), calculated from 50 data replications.

Policy evaluation is a critical component of RL, where the goal is to estimate the value function representing the expected cumulative reward under a given policy. Stable and accurate policy evaluation significantly impacts the overall

*Co-corresponding authors

performance of RL algorithms. We demonstrate the effectiveness of experience replay in two policy evaluation algorithms: (i) the Least-Squares Temporal Difference (LSTD) algorithm for Markov Decision Process (MDP) (Bradtke and Barto 1996), and (ii) the PDE-based algorithm for environments with continuous-time state dynamics (Zhu 2024). LSTD is a widely used data-efficient policy evaluation method that approximates value functions within a linear space by solving a least-squares problem derived from the Bellman equation (Bradtke and Barto 1996). The PDE-based approach is a novel method that employs a PDE framework to approximate the continuous-time value function and is tailored for environments where the state variable evolves continuously over time, which is a common scenario in applications such as autonomous driving (Kong et al. 2015) and robotics (Kober, Bagnell, and Peters 2013), where discrete-time models like MDP may fail to capture the complexity of the environment. Incorporating experience replay into these algorithms significantly enhances their stability by reducing variance, as illustrated in Figure 1. Rather than the original method which uses the entire dataset at once, the experience replay method resamples subsets, either without or with replacement, from the original dataset. These subsets are used to generate predictions, which are then averaged using resampled U- or V-statistics to produce the final prediction. This resampling approach enables data duplication, mitigates variability in predictions due to changes in the dataset, and enhances stability through the averaging process. This improvement is particularly important in practice, as the numerical results in Zhu (2024) indicate that the original RL algorithm solution can exhibit substantial instability.

While experience replay methods have been extensively validated empirically in RL, our contribution lies in providing a theoretical framework that explains why experience replay is effective in practice, particularly for policy evaluation. The experience replay technique can be further improved by incorporating extensions such as prioritized experience replay based on importance sampling (Schaul 2015). Our theoretical framework can also be extended to analyze these variants.

Besides RL, we apply our framework to supervised learning tasks using kernel ridge regression, where each regression sample is treated as an experience. Kernel ridge regression enhances modeling flexibility by leveraging reproducing kernel methods to map data into RKHS (Wahba 1990; Shawe-Taylor and Cristianini 2004). Unlike existing divide-and-conquer strategies that partition datasets into disjoint subsets to reduce computational costs (Zhang, Duchi, and Wainwright 2013), our approach employs experience replay to repeatedly draw random subsamples, providing a novel strategy to improve computational efficiency. With appropriate parameter choices, our method reduces the computational cost of kernel ridge regression from the traditional $O(n^3)$ to $O(n^2)$. At the same time, our theoretical results guarantee that the variance of the predictions is lower than that of the original kernel ridge regression method. Hence, incorporating experience replay leads to more stable and faster predictions in supervised learning tasks.

We validate the effectiveness of our proposed framework through extensive experiments. The results consistently demonstrate that experience replay significantly reduces the variance of predictions compared to methods without it, highlighting its ability to enhance stability in both reinforcement learning and supervised learning tasks. Additionally, it reduces the computational cost in kernel ridge regression with an appropriate choice of parameters. Notably, experience replay generally leads to both reduced variance and lower mean squared error in predictions.

The rest of the paper is organized as follows. Section 2 introduces the background of experience replay and its connection to resampled U- or V-statistics. Section 3 defines the resampled estimators, establishes their variance reduction guarantees, and discusses applications in policy evaluation and supervised learning tasks. Section 4 presents numerical experiments to validate the theoretical findings. Section 5 concludes the paper with potential future directions. All technical proofs are included in the Appendix presented in the extended version of the paper.

2 Background

Experience Replay. Experience replay stores past data in a replay buffer, denoted as $\mathcal{D}_n = \{Z_1, \ldots, Z_n\}$, where n represents the sample size, commonly referred to as the replay capacity in the context of experience replay (Lin 1992). The replay ratio $B \geq 1$ denotes the number of batches sampled from the buffer during each update step. In practice, uniform sampling is the most common strategy for selecting data from the replay buffer, although more computationally expensive alternatives, such as prioritized experience replay, are also used (Zhang and Sutton 2017; Schaul 2015). This paper establishes theoretical guarantees for replay-based methods under uniform sampling. The proposed framework, however, can be extended to non-uniform (importance) sampling, as discussed in Appendix A.

At each update step, we sample B subsets of data points, $\{b_1, \ldots, b_B\}$, where each subset b_i $(i = 1, \ldots, B)$ contains $k \leq n$ data points. The learning method is represented by a function h_k, which takes k data points as input. The response with experience replay is then computed as the average over these B subsets:

$$\frac{1}{B} \sum_i h_k(b_i). \tag{1}$$

In experience replay for Q-learning, each data point in the replay buffer $\mathcal{D}_n$ corresponds to a single transition, and $k = 1$ (Fedus et al. 2020). This paper studies algorithms such as LSTD, where k could increase with n. LSTD is a foundational and actively studied RL algorithm for policy evaluation (e.g., Tu and Recht 2018; Duan, Wang, and Wainwright 2024), which serves as an essential step to theoretically understand experience replay in other RL methods, such as Q-learning.

Connection to Resampled Statistics. To analyze the properties of experience replay (1), we consider, for clarity of exposition, a setting where the replay buffer $\mathcal{D}_n$ contains n i.i.d. observations drawn from an underlying distribution F_Z over the space $\mathcal{Z}$. The i.i.d. assumption can be relaxed in

various ways without affecting our results (see Appendix B). We allow B and k to depend on n, with k increasing in n. This ensures that the function h_k can use more information as the data size grows.

When the sampling strategy is uniform sampling *without* replacement, the computation in (1) takes the form of an incomplete, infinite order (or *resampled*) U-statistics (Frees 1989; Zhou, Mentch, and Hooker 2021), defined as:

$$U_{n,k,B} = \frac{1}{B}\sum_{i} h_k(Z_{i_1}, \ldots, Z_{i_k}). \tag{2}$$

where infinite order means that k and B depend on the value of n, and $\{Z_{i_1}, \ldots, Z_{i_k}\}$ are drawn without replacement from $\{Z_1, \ldots, Z_n\}$. In contrast, with uniform sampling *with* replacement, the computation in (1) follows the form of an incomplete, infinite order (or *resampled*) V-statistics (Shieh 1994; Zhou, Mentch, and Hooker 2021), given by:

$$V_{n,k,B} = \frac{1}{B}\sum_{i} h_k(Z_{i_1}, \ldots, Z_{i_k}), \tag{3}$$

where k and B again depend on n, and the B subsets are drawn with replacement from all size-k permutations of $\{1, \ldots, n\}$.

Under appropriate regularity conditions, both resampled U-statistics and V-statistics are asymptotically normal (Mentch and Hooker 2016; Zhou, Mentch, and Hooker 2021). The variances of these statistics can be expressed as a linear combination of $\frac{k^2}{n}\zeta_{1,k}$ and $\frac{1}{B}\zeta_{k,k}$. For a given c, $1 \le c \le k$, the variance components $\zeta_{c,k}$ are defined as

$$\mathrm{Cov}\Big(h_k(Z_1, \ldots, Z_k), h_k(Z_1, \ldots, Z_c, Z'_{c+1}, \ldots, Z'_k)\Big),$$

where $Z'_{c+1}, \ldots, Z'_k$ are i.i.d. copies from F_Z, independent of the original data set $\mathcal{D}_n$.

Learning Target. We focus on estimating the quantity defined as,

$$\theta = \big[\mathbb{E}[g(Z)]\big]^{-1}\big[\mathbb{E}[f(Z)]\big] \in \mathbb{R}^q, \tag{4}$$

where $g(\cdot) : \mathcal{Z} \to \mathbb{R}^{q\times q}$ is a function returning an invertible matrix, and $f(\cdot) : \mathcal{Z} \to \mathbb{R}^q$. The target θ arises in various machine learning applications, including policy evaluation algorithms in reinforcement learning (Bradtke and Barto 1996; Zhu 2024), and supervised learning with kernel ridge regression (Wahba 1990; Rahimi and Recht 2007). We will discuss the application of experience replay to these methods in Section 3.2.

To estimate θ in (4), we use a function h_k based on $k \le n$ data points $Z_1^*, Z_2^*, \ldots, Z_k^*$ for any $Z_i^* \in \mathcal{D}_n$, $i = 1, \ldots, k$, where h_k in (1) is defined as:

$$h_k(Z_1^*, \ldots, Z_k^*) := \Big[\sum_{i=1}^{k} g(Z_i^*)\Big]^{-1}\Big[\sum_{i=1}^{k} f(Z_i^*)\Big] \in \mathbb{R}^q. \tag{5}$$

The learning function in (5) provides a unified framework that applies to several algorithms, including the LSTD algorithm in reinforcement learning and kernel ridge regression in supervised learning. We will theoretically show that incorporating the experience replay approach (1) reduces the variance of the estimate of θ and thus improves stability.

Algorithm 1 Estimating θ via Different Methods

1: **Input:** Replay buffer $\mathcal{D}_n = \{Z_1, \ldots, Z_n\}$; Functions f and g; Replay ratio (number of subsamples) B; Subsample size k.
2: **Original Estimator:** Compute $\tilde{\theta}_n$ using (6).
3: **Resampled Estimators Based on $U(V)$-statistics:**
4: **for** $i = 1$ to B **do**
5: Randomly drawn k samples $\{Z_{i_1}, \ldots, Z_{i_k}\}$ without (for U-statistics) or with replacement (for V-statistics).
6: **end for**
7: Compute $\hat{\theta}_U$ or $\hat{\theta}_V$ using (7) or (8), respectively.
8: **Output:** Estimators $\tilde{\theta}_n$, $\hat{\theta}_U$, and $\hat{\theta}_V$.

3 Main Results

3.1 Theoretical Guarantees

Estimators without Experience Replay. When the experience replay approach is not used, and each data point in the replay buffer $\mathcal{D}_n$ is used only once, a plug-in estimator for θ in (5) is:

$$\tilde{\theta}_n := \Big[\sum_{i=1}^{n} g(Z_i)\Big]^{-1}\Big[\sum_{i=1}^{n} f(Z_i)\Big]. \tag{6}$$

The asymptotic property of $\tilde{\theta}_n$ is described in the following lemma. The proof relies on the central limit theorem and the delta method.

Lemma 1 *Let $Z_1, Z_2, \ldots, Z_n \overset{iid}{\sim} F_Z$ and $\tilde{\theta}_n$ defined in* (6)*, we have that $\sqrt{n}\big[\tilde{\theta}_n - \theta\big] \xrightarrow{d} N(0, \Sigma)$, where Σ is a constant matrix given by*

$$G\begin{pmatrix} Var(f(Z)) & Cov(f(Z), vec(g(Z))) \\ Cov(f(Z), vec(g(Z))) & Var(vec(g(Z)) \end{pmatrix} G^\top,$$

with $G = \Big([\mathbb{E}[g(Z)]]^{-1}, -\theta^\top \otimes [\mathbb{E}[g(Z)]]^{-1}\Big)$, where $\otimes$ denotes the Kronecker product, and $vec(A)$ reshapes a matrix A into a column vector by stacking its columns sequentially.

Estimators with Experience Replay. Using the experience replay approach, we propose two new estimators for θ that leverage resampling methods based on U- and V-statistics. These estimators are constructed using the learning method h_k defined in (5),

$$\hat{\theta}_U := U_{n,k,B} = \frac{1}{B}\sum_{i} h_k(Z_{i_1}, \ldots, Z_{i_k}), \tag{7}$$

$$\hat{\theta}_V := V_{n,k,B} = \frac{1}{B}\sum_{i} h_k(Z_{i_1}, \ldots, Z_{i_k}), \tag{8}$$

where $U_{n,k,B}$ and $V_{n,k,B}$ are resampled U- and V-statistics defined in (2) and (3), respectively. Algorithm 1 outlines the procedure for computing these estimators. The following theorem establishes that the U-statistics-based estimators achieve lower variances than the original estimator under general conditions.

Theorem 1 (Variance Reduction for U-Statistics) *Let $Z_1, Z_2, \ldots, Z_n \overset{iid}{\sim} F_Z$, and define $\hat{\theta}_U$ as in* (7) *and $\tilde{\theta}_n$ as in*

(6). *Under the assumption that* $\lim_{n\to\infty} \frac{1}{n}\zeta_{k,k}[\zeta_{1,k}]^{-1} \to 0$ *and* $\lim_{n\to\infty} n/(Bk) \to 0$*, we have*

$$\liminf_{n\to\infty}[Var(\tilde{\theta}_n) - Var(\hat{\theta}_U)] \geq 0.$$

The assumption $\lim_{n\to\infty} \frac{1}{n}\zeta_{k,k}[\zeta_{1,k}]^{-1} \to 0$ used by Peng, Coleman, and Mentch (2022), ensures the asymptotic normality of the resampled U-statistics. As noted in their work, this condition is typically satisfied if $\frac{1}{k}\zeta_{k,k}[\zeta_{1,k}]^{-1}$ remains bounded, with $k = o(n)$ being sufficient. Additionally, the theorem requires n/Bk to be small, which can be achieved by selecting a large replay ratio B.

To analyze the variance reduction for V-statistics-based estimators, we define the following class of functions $\mathcal{H} = \left\{h_k : \sup_k \|\mathbb{E}[h_k(Z_{i_1},\ldots,Z_{i_k})h_k(Z_{i_1},\ldots,Z_{i_k})^\top]\|_\infty < \infty\right\}$, where $(i_1,\ldots,i_k)$ are indices selected with replacement from $\{1,\ldots,k\}$. This condition, used by Zhou, Mentch, and Hooker (2021), ensures the boundedness of the expected outer product of h_k.

Theorem 2 (Variance Reduction for V-Statistics) *Let* $Z_1, Z_2, \ldots, Z_n \overset{iid}{\sim} F_Z$*, and define* $\hat{\theta}_V$ *as in* (8) *and* $\tilde{\theta}_n$ *as in* (6)*, with* $h_k \in \mathcal{H}$*. Under the assumptions* $k = o(n^{1/4})$*,* $\lim_{n\to\infty} k^2\zeta_{1,k} > 0$*, and* $\lim_{n\to\infty} n/(Bk) \to 0$*, we have*

$$\liminf_{n\to\infty}[Var(\tilde{\theta}_n) - Var(\hat{\theta}_V)] \geq 0.$$

The condition $\lim_{n\to\infty} k^2\zeta_{1,k} > 0$, which is satisfied by many base learners and has been used in prior work (Song, Chen, and Kato 2019; Zhou, Mentch, and Hooker 2021), is further discussed in Appendix C, where we show that it holds in our framework.

Theorems 1 and 2 show that incorporating experience replay via resampled U- and V-statistics asymptotically reduces variance compared to the original estimator, enhancing the stability of parameter estimation. Our results remain valid under more general data-generating processes beyond the i.i.d. setting, including dependent sequences such as stationary ergodic Markov chains, β-mixing processes with summable coefficients, and m-dependent sequences; see Appendix B for details.

3.2 Applications to Machine Learning Problems

Policy Evaluation for MDP. Consider a MDP defined by the tuple $(\mathbb{S}, \mathbb{A}, \gamma, r, \mathbb{P})$ (Sutton and Barto 2018). Here $s \in \mathbb{S}$ denotes the state space, $a \in \mathbb{A}$ represents the action space, $\gamma \in (0,1)$ is a given discounted factor, $r : \mathbb{S}\times\mathbb{A} \to \mathbb{R}$ is the reward function, and $\mathbb{P} : \mathbb{S}\times\mathbb{A} \to \Delta(\mathbb{S})$ denotes the probability distribution of the next state given the current state and action. The goal of MDP is to find the optimal policy $\pi^*(s)$ that maximizes the value function. Here the policy is a mapping from the state space $\mathbb{S}$ to action space $\mathbb{A}$, while the value function $V_*^\pi(s)$ measures the expected cumulative reward of an agent over the long run, defined as:

$$V_*^\pi(s) = \mathbb{E}\left[\sum_{j=0}^{\infty} \gamma^j r_*^\pi(s_j)\Big| s_0 = s\right], \tag{9}$$

where $s_0 = s$ is the initial state, $r_*^\pi(s) = r(s,\pi(s))$ is a known reward function under the current policy, and the state at time step $j+1$ follows the transition distribution under the policy π, $s_{j+1} \sim P^\pi(\cdot|s_j) = P(\cdot|s_j,\pi(s_j))$. In RL, one usually divides the RL problem into two parts, one is policy evaluation, which is given a policy $\pi(s)$, calculates the value function $V_*^\pi(s)$; Another is policy improvement, that improves the policy according to gradient ascent or policy iteration.

The focus of this paper is policy evaluation, which is one of the most fundamental RL problems. In the setting of RL, one does not have access to the transition distribution. Instead, the agent applies an action $a_j = \pi(s_j)$ according to the policy at each time step j, and observes the next step s_{j+1}, receives a numerical reward $r_*^\pi(s_{j+1})$. Due to the finite length of the trajectory data, it is usually impossible to compute the value function directly according to the cumulative sum (9). Note that the value function $V_*^\pi(s)$ also satisfies the following Bellman equation (BE),

$$V_*^\pi(s) = r_*^\pi(s) + \gamma\mathbb{E}_{s_{j+1}\sim P^\pi(s'|s_0)}[V_*^\pi(s_{j+1})|s_0 = s]. \tag{10}$$

Therefore, the goal of the policy evaluation problem is to find the value function that solves BE (10) given a set of trajectory data,

$$\mathcal{D}_n = \{(s_0^l, s_1^l, \ldots, s_L^l)\}_{l=1}^n. \tag{11}$$

Here the data set contains n independent trajectories and each contains $L+1$ data points. The initial state s_0^l of each trajectory is sampled from a distribution $\rho_0^\pi(s)$. Our method also extends to settings with dependent data and variable-length trajectories (see Appendix B).

LSTD (Bradtke and Barto 1996) is a popular RL algorithm for linear approximation and can be directly used to estimate $V_*^\pi(s)$ using the trajectory data. LSTD approximates the value function $V_*^\pi(s) = \Phi(s)^\top\theta$ in the space expanded by q given bases $\{\phi_i(s)\}_{i=1}^q$, where $\theta \in \mathbb{R}^q$ is a unknown parameter and $\Phi(s) = (\phi_1(s),\cdots,\phi_q(s))^\top$. By projecting the value function into the finite bases, LSTD solves the parameter θ in the form of

$$\left[\mathbb{E}_s[\Phi(s)(\Phi(s) - \gamma\mathbb{E}[\Phi(s_1)|s_0 = s])^\top]\right]^{-1}\mathbb{E}_s[r_*^\pi(s)\Phi(s)]. \tag{12}$$

Using any trajectory data subset with $k \leq n$ data points $\{(s_j^{l_{(1)}})_{j=0}^L, \ldots, (s_j^{l_{(k)}})_{j=0}^L\}$ for any $(s_j^{l_{(i)}})_{j=0}^L \in \mathcal{D}_n$, $i = 1,\ldots,k$, the estimator of θ is

$$\left[\sum_{i=1}^k g((s_j^{l_{(i)}})_{j=0}^L)\right]^{-1}\left[\sum_{i=1}^k f((s_j^{l_{(i)}})_{j=0}^L)\right],$$

corresponds to the structure of (5), where

$$g((s_j^{l_{(i)}})_{j=0}^L) = \sum_{j=0}^{L-1}\Phi(s_j^{l_{(i)}})[\Phi(s_j^{l_{(i)}}) - \gamma\Phi(s_{(j+1)}^{l_{(i)}})]^\top,$$

$$f((s_j^{l_{(i)}})_{j=0}^L) = \sum_{j=0}^{L-1} r_*^\pi(s_j^{l_{(i)}})\Phi(s_j^{l_{(i)}}). \tag{13}$$

This setup aligns with our framework, where $Z_i = (s_j^i)_{j=0}^L$ for $i = 1,\ldots,n$, θ is defined in (12), and the functions g and f are defined in (13).

Our theories also help explain prior empirical findings on experience replay in Q-learning (Zhang and Sutton 2017; Fedus et al. 2020); see Appendix D for details.

Policy Evaluation for Continuous-Time RL. In the second application, we aim to solve the policy evaluation problem for continuous-time RL (e.g., Zhu 2024). Given a policy $\pi(s)$, unlike the MDP where the value function is a cumulative sum over discrete time steps defined as (9), the value function in continuous-time RL is an expected integral over continuous time,

$$V^\pi(s) = \mathbb{E}\left[\int_0^\infty e^{-\beta t} r^\pi(s_t) dt \Big| s_0 = s\right]. \tag{14}$$

Here $\beta > 0$ is a given discounted coefficient, $r^\pi(s) \in \mathbb{R}$ is a known reward function under the current policy. When the state $s_t \in \mathbb{S} \subset \mathbb{R}^d$ is driven by the stochastic differential equation (SDE),

$$ds_t = \mu(s_t)dt + \sigma(s_t)dB_t, \tag{15}$$

by Feynman–Kac theorem (Stroock and Varadhan 1997), the value function $V(s)$ satisfies the equation $\beta V^\pi(s) = r^\pi(s) + \mathcal{L}_{\mu,\Sigma} V^\pi(s)$, where $\mathcal{L}_{\mu,\Sigma} = \mu(s)\cdot\nabla + \frac{1}{2}\Sigma : \nabla^2$ with $\Sigma = \sigma\sigma^\top$, and $\Sigma : \nabla^2 = \sum_{i,j}\Sigma_{ij}\partial_{s_i}\partial_{s_j}$. Similar to the classical RL setting, one does not have access to the drift function $\mu(s) \in \mathbb{R}^d$ and diffusion function $\sigma(s) \in \mathbb{R}^{d\times d}$. Therefore, one cannot solve the above equation directly. The goal of continuous-time policy evaluation is to find the value function satisfying (15) with a set of trajectory data $\mathcal{D}_n$ defined in (11). Here the data at time step j are collected at time $j\Delta t$ with a given time interval Δt.

Zhu (2024) introduced an algorithm to approximate the value function by solving a Physics-informed Bellman equation (PhiBE) defined as follows

$$\beta \bar{V}_\alpha^\pi(s) - \mathcal{L}_{\hat{\mu}_\alpha,\hat{\Sigma}_\alpha}\bar{V}_\alpha^\pi(s) = r^\pi(s), \quad \alpha = 1, 2 \tag{16}$$

where $\hat{\mu}_\alpha(s) = \frac{1}{\Delta t}\sum_{j=1}^\alpha \mathbb{E}_{s_j}\left[a_j^{(\alpha)}(s_j - s_0)|s_0 = s\right]$, $\hat{\Sigma}_\alpha(s) = \frac{1}{\Delta t}\sum_{j=1}^\alpha \mathbb{E}_{s_j}\left[a_j^{(\alpha)}(s_j - s_0)(s_j - s_0)^\top|s_0 = s\right]$ and

$$\alpha = 1:\ a_1^{(1)} = 1; \quad \alpha = 2:\ a_1^{(2)} = 2,\ a_2^{(2)} = -\frac{1}{2}. \tag{17}$$

Here $\bar{V}_\alpha^\pi(s)$ serves as α-th order approximation to the continuous-time value function $V^\pi(s)$, where $\alpha \in \{1, 2\}$.

Similar to (12), if one approximates the solution $\bar{V}(s) = \Phi(s)^\top\theta$ to the PhiBE (16) in the linear function space spanned by $\Phi(s)$, one ends up solving for the parameter θ in the following form

$$\left[\mathbb{E}_s[(\beta\Phi(s)^\top - \mathcal{L}_{\hat{\mu}_\alpha,\hat{\Sigma}_\alpha}\Phi(s)^\top)\Phi(s)]\right]^{-1}\mathbb{E}_s[r^\pi(s)\Phi(s)]. \tag{18}$$

Zhu (2024) gives the model-free algorithm to estimate the θ using only trajectory data. Specifically, for the α-th order case, using any data subset with $k \le n$ data points $\{(s_j^{l_{(1)}})_{j=0}^L, \ldots, (s_j^{l_{(k)}})_{j=0}^L\}$ for any $(s_j^{l_{(i)}})_{j=0}^L \in \mathcal{D}_n$, $i = 1, \ldots, k$, the estimator of θ is

$$\left[\sum_{i=1}^k g((s_j^{l_{(i)}})_{j=0}^L)\right]^{-1}\left[\sum_{i=1}^k f((s_j^{l_{(i)}})_{j=0}^L)\right],$$

corresponds to the structure of (5), where

$$g((s_j^{l_{(i)}})_{j=0}^L) = \sum_{j=0}^{L-\alpha}\Phi(s_j^{l_{(i)}})\left[\beta\Phi(s_j^{l_{(i)}}) - \mathcal{L}_{\bar{\mu}_\alpha,\bar{\Sigma}_\alpha}\Phi(s_j^{l_{(i)}})\right]^\top,$$

$$f((s_j^{l_{(i)}})_{j=0}^L) = \sum_{j=0}^{L-\alpha} r^\pi(s_j^{l_{(i)}})\Phi(s_j^{l_{(i)}}), \tag{19}$$

and the estimators of $\mu(s)$ and $\sigma(s)$ are defined as $\bar{\mu}_\alpha(s_j^l) = \frac{1}{\Delta t}\sum_{k=1}^\alpha a_k^{(\alpha)}(s_{(j+k)}^l - s_j^l)$, $\bar{\Sigma}_\alpha(s_j^l) = \frac{1}{\Delta t}\sum_{k=1}^\alpha a_k^{(\alpha)}(s_{(j+k)}^l - s_j^l)(s_{(j+k)}^l - s_j^l)^\top$ with $a^{(\alpha)}$ defined as (17). Compared to LSTD, the second-order PDE-based algorithm with $\alpha = 2$ incorporates two future steps, resulting in improved accuracy, as illustrated in Figure 1. This setup aligns with our framework with $Z_i = (s_j^i)_{j=0}^L$, $i = 1, \ldots, n$ and θ defined in (18), and functions g and f defined in (19).

For both LSTD and the PDE-based approach, once we obtain $\tilde{\theta}_n$, $\hat{\theta}_U$, and $\hat{\theta}_V$ by applying Algorithm 1, the corresponding estimations of the value function at a test point s are defined as $\tilde{V}(s) = \Phi(s)^\top\tilde{\theta}_n, \hat{V}_U(s) = \Phi(s)^\top\hat{\theta}_U$, and $\hat{V}_V(s) = \Phi(s)^\top\hat{\theta}_V$, where the superscript π is omitted. The variances are, $\mathrm{Var}(\tilde{V}(s)) = \Phi(s)^\top\mathrm{Var}(\tilde{\theta}_n)\Phi(s)$, $\mathrm{Var}(\hat{V}_U(s)) = \Phi(s)^\top\mathrm{Var}(\hat{\theta}_U)\Phi(s)$, and $\mathrm{Var}(\hat{V}_V(s)) = \Phi(s)^\top\mathrm{Var}(\hat{\theta}_V)\Phi(s)$. Thus the reduction of the variance of estimators of θ could be directly evaluated by the reduction in the variance of these estimations.

Kernel Ridge Regression. In the third application, we consider a supervised learning framework where $\mathcal{D}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ consists of i.i.d. samples drawn from a distribution F_Z. Our goal is to predict the outcome $Y \in \mathbb{R}$ based on the predictors $X \in \mathbb{R}^p$ using kernel methods in an RKHS (Wahba 1990). Let $K(\cdot,\cdot) : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ be a reproducing kernel function. We consider the model $Y = f(X) + \epsilon$, where f belongs to the RKHS defined by K, and ϵ represents random error independent of X. Following Rahimi and Recht (2007) and Dai, Lyu, and Li (2023), the kernel function $K(X_i, X_j)$ can be approximated as $\phi(X_i)^\top\phi(X_j)$ using a feature mapping $\phi : \mathbb{R}^p \to \mathbb{R}^q$, and $f(X)$ can be approximated as $\phi(X)^\top\theta$, where $\theta \in \mathbb{R}^q$ is a parameter vector, defined as

$$\theta = \left[\mathbb{E}[\phi(X)\phi(X)^\top]\right]^{-1}\mathbb{E}[\phi(X)Y]. \tag{20}$$

Using any k data points $\{(X_1^*, Y_1^*), \ldots, (X_k^*, Y_k^*)\}$ resampled from $\mathcal{D}_n$, the kernel ridge regression estimator of θ is obtained by solving the following optimization problem for a given $\lambda \ge 0$,

$$\underset{\theta\in\mathbb{R}^q}{\mathrm{argmin}}\left\{\sum_{i=1}^k[Y_i^* - \phi(X_i^*)^\top\theta]^2 + \lambda||\theta||_2^2\right\}.$$

The solution takes the form of

$$\left[\sum_{i=1}^k g(X_i^*, Y_i^*) + \lambda\mathbb{I}_p\right]^{-1}\left[\sum_{i=1}^k f(X_i^*, Y_i^*)\right], \tag{21}$$

where $g(X_i^*, Y_i^*) = \phi(X_i^*)\phi(X_i^*)^\top$ and $f(X_i^*, Y_i^*) = \phi(X_i^*)Y_i^*$. The setup in (20) and (21) aligns with our framework in (4) and (5), with an added regularization term $\lambda\mathbb{I}_p$. This term does not impact the derivation of our main results.

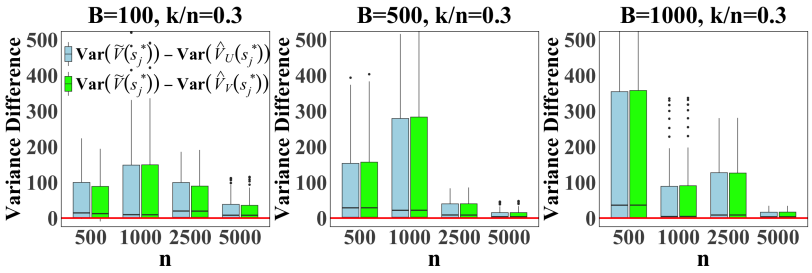


Figure 2: Variance differences among the predicted policy values using the LSTD algorithm with $m = 50$, $M = 50$, and $k/n = 0.3$, evaluated across various values of n and B. $\tilde{V}(s_j^*)$ represents the results without experience replay, while $\hat{V}_U(s_j^*)$ and $\hat{V}_V(s_j^*)$ represent the results with experience replay. The red line represents the baseline where the variance difference is 0.

The standard computational cost of the kernel ridge regression with n data points is $O(n^3)$ in time (Wahba 1990). The divide-and-conquer algorithm (Zhang, Duchi, and Wainwright 2013) reduces this cost by dividing the dataset into $m < n$ disjoint subsets, each of n/m, and averaging the local solutions across these subsets to construct a global predictor. This approach achieves a trade-off between computational cost and estimation error. In contrast, our approach, which incorporates the experience replay method, also averages over subsets but differs fundamentally in how the subsets are constructed. Instead of partitioning the dataset into non-overlapping subsets, we repeatedly draw B random subsamples, each containing k data points. This resampling allows for overlapping subsets and potential duplication of data points, resulting in a total computational cost of $O(Bkq^2 + Bq^3)$ in time. Theorems 1 ensures that the conditions $\lim_{n\to\infty} n/(Bk) \to 0$ and $k = o(n)$ are sufficient for variance reduction. By carefully choosing B and k, our approach achieves both computational savings and variance reduction, offering a practical and efficient alternative to traditional kernel ridge regression, especially for large-scale problems. For instance, setting $B = O(n^{13/8}), k = O(n^{1/8}), q = O(n^{1/8})$, satisfies the conditions of Theorems 1 and reduces the variance. In this setup, the computational cost is further reduced to $O(n^2)$ in time. Additional discussion and examples are provided in Appendix E.

4 Numerical Experiments

4.1 Experiments of Policy Evaluation Using LSTD Algorithm

Firstly, we present the experimental results obtained using LSTD with functions g and f defined in (13). We conduct the experiments in a similar setting as described in Zhu (2024), where the state space $\mathbb{S} = [-\pi, \pi]$, and the state under policy π is driven by the transition distribution $P^\pi(s_{j+1}|s_j)$ following the normal distribution with expectation $se^{\lambda/10}$, variance $\frac{\sigma^2}{2\lambda}(e^{\lambda/5}-1)$, where $\lambda = 0.05$ and $\sigma = 1$. The reward function is set to be $r_*^\pi(s) = 0.1 * [\cos^3(s) - \lambda s(-3\cos^2(s)\sin(s)) - \frac{1}{2}\sigma^2(6\cos(s)\sin^2(s) - 3\cos^3(s))]$ and the discounted factor γ is set to be $e^{-0.1}$. We use periodic bases $\{\phi_n(s)\}_{k=1}^{2I+1} = \frac{1}{\sqrt{\pi}}\{\frac{1}{\sqrt{2}}, \cos(is), \sin(is)\}_{i=1}^{I}$ with $I = 4$. We consider the case $L = 2$, where each trajectory has three data points and the state s_j^l in D_n (11) is sampled at time $j/10$ for $j = 0, \ldots, L$ and $l = 1, \ldots, n$. In each experiment, we draw n independent trajectories $\mathcal{D}_n$ with the initial state s_0^l of each trajectory sampled from a truncated normal distribution over $\mathbb{S}$ with mean 0 and standard deviation 0.1.

We check the performance of the three prediction models on $m = 50$ test points evenly selected in $\mathbb{S}$, denoted by $\mathcal{S}_{test} = \{s_j^*\}_{j=1}^m$ with $s_j^* = -\pi + 2(j-1) * \pi/(m-1)$. The experiment is conducted $M = 50$ times, and the variance of the estimated outcome for each test state s_j^*, where $j = 1, \ldots, m$, is approximated using the sample variance. Three different estimators are used, resulting in approximate variances denoted by $\text{Var}(\tilde{V}(s_j^*))$, $\text{Var}(\hat{V}_U(s_j^*))$ and $\text{Var}(\hat{V}_V(s_j^*))$. To assess the variance reduction property, we compare these three variances across all test states.

Figure 2 compares the variances using standard boxplots that display the quartile breakdown of the differences $\{\text{Var}(\tilde{V}(s_j^*)) - \text{Var}(\hat{V}_U(s_j^*))\}_{j=1}^m$ and $\{\text{Var}(\tilde{V}(s_j^*)) - \text{Var}(\hat{V}_V(s_j^*))\}_{j=1}^m$, with $n \in \{500, 1000, 2500, 5000\}$, $B = \{100, 500, 1000\}$, and $k/n = 0.3$. The results clearly demonstrate that for all of the different parameters, the variance differences across all test data points are consistently greater than 0 for both U- and V-statistics-based experience replay methods, particularly in data-scarce settings. As n increases, the variance differences become small as all estimation methods exhibit reduced variance; nonetheless, the variance reduction remains substantial. To illustrate this, we consider the case where $n = 5000$, $B = 1000$, and $k/n = 0.3$, as shown in Figure 1a. From the figure, we observe that the resampled methods demonstrate a significant improvement in variance in this large n scenario. Additional experiments with varying choices of k/n are provided in Appendix G.1, further confirming the robustness of the approach.

4.2 Experiments of Policy Evaluation Using PDE-Based Algorithm

Secondly, we present the experimental results obtained using the second-order PDE-based algorithm with functions g and f defined in (19) with $\alpha = 2$. Similar to Zhu (2024), we consider an experimental setting where the state dynam-

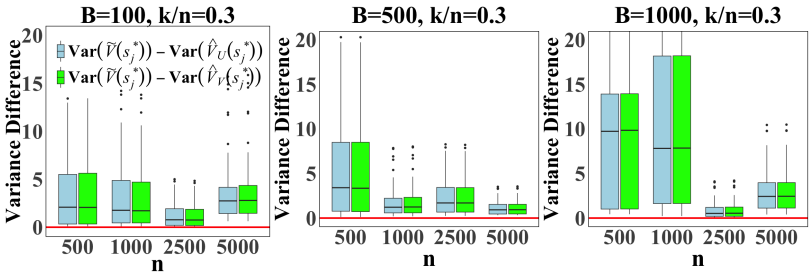


Figure 3: Variance differences among the predicted policy values using the second-order PDE-based algorithm with $m = 50$, $M = 50$, and $k/n = 0.3$, evaluated across various values of n and B. $\tilde{V}(s_j^*)$ represents the results without experience replay, while $\hat{V}_U(s_j^*)$ and $\hat{V}_V(s_j^*)$ represent the results with experience replay. The red line represents the baseline where the variance difference is 0.

ics are governed by the Ornstein–Uhlenbeck (OU) process $ds(t) = \lambda s dt + \sigma dB_t$ with $\lambda = 0.05, \sigma = 1$. The reward function is set to be $r^\pi(s) = \beta\cos^3(s) - \lambda s(-3\cos^2(s)\sin(s)) - 0.5\sigma^2(6\cos(s)\sin^2(s) - 3\cos^3(s))$ with the discounted coefficient $\beta = 0.1$. For the OU process, the transition distribution $P^\pi(s'|s)$ from time t to $t + \Delta t$ follows a normal distribution with mean $se^{\lambda\Delta t}$ and variance $\frac{\sigma^2}{2\lambda}(e^{2\lambda\Delta t} - 1)$. We set $\Delta t = 0.1$, and under this setting, D_n in Section 4.1 follows the same transition distribution, allowing us to use the same simulated trajectory data. Additionally, we employ the same periodic basis functions as described in Section 4.1. The true value function $V^\pi(s)$ then can be exactly obtained from (14), $V^\pi(s) = \cos^3(s)$.

Note that the experiments using LSTD in Section 4.1 can be considered as a way for estimating $V^\pi(s)$ by discretizing it as a MDP. This approach uses the relationships $r_*^\pi(s) = r^\pi(s)\Delta t$ and $\gamma = e^{-\beta\Delta t}$, which hold in the given setting. However, as observed in Figure 1, when the original methods are used, the PDE-based approach generally shows greater accuracy with narrower confidence bands.

We evaluate the performance of the three prediction models using the same way as in Section 4.1. Figure 3 clearly demonstrates that for all of the different parameters, the variance differences across all test data points are consistently greater than 0 for both U- and V-statistics-based experience replay methods. Figure 1b illustrates the large n case where $n = 5000$, $B = 500$, and $k/n = 0.3$.

We present additional experiments with different choices of k/n, along with first-order results in Appendix G.1. With the use of experience replay, the second-order method achieves a greater percentage reduction in variance compared to the LSTD method. Intuitively, the second-order method accounts for two future steps, introducing more stochasticity, which provides greater potential for variance reduction. Moreover, we compare the root mean squared error (RMSE) of the proposed methods with the original method over the m test points across all M experiments for both the LSTD and PDE-based methods in Appendix H.1. The results demonstrate that the combination of experience replay, regardless of the specific resampling method used, not only reduces variance but also tends to achieve smaller prediction errors, further highlighting its effectiveness.

4.3 Experiments of Kernel Ridge Regression

Thirdly, we consider a regression setting where for each $(X, Y) \sim F_Z$, the predictor $X = (X_{(1)}, X_{(2)}) \in \mathbb{R}^2$ is generated with $X_{(1)}, X_{(2)} \sim \text{Unif}(0, 1)$, and the response is given by $Y = e^{10(-(X_{(1)}-0.25)^2-(X_{(2)}-0.25)^2)} + 0.5 \cdot e^{14(-(X_{(1)}-0.7)^2-(X_{(2)}-0.7)^2)} + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 0.25)$ is independent of X. This setting is widely used in the study of kernel ridge regression and generalized regression models (see, Hainmueller and Hazlett 2014; Wood 2003).

For each experiment, we independently draw n data points from F_Z to form the training dataset $\mathcal{D}_n$. We use the `krls` function in `R` to fit the kernel ridge regression model with a Gaussian kernel. The λ is chosen as $n^{-2/3}$. We evaluate the performance of these models on $m = 100$ test points independently drawn from F_Z, denoted by $\mathcal{D}_{test} = \{(x_j, y_j)\}_{j=1}^m$. The experiment is repeated $M = 100$ times, and the variances of the predicted outcomes $\tilde{y}_j, \hat{y}_{j,U}$, and $\hat{y}_{j,V}$ for each test predictor x_j, where $j = 1, \ldots, m$, are approximated using the sample variances, denoted by $\text{Var}(\tilde{y}_j)$, $\text{Var}(\hat{y}_{j,U})$, and $\text{Var}(\hat{y}_{j,V})$. As stated in Dai, Lyu, and Li (2023), the predictions $\tilde{y}_j$, $\hat{y}_{j,U}$, and $\hat{y}_{j,V}$ are approximately equal to $\phi(x_j)^\top\tilde{\theta}_n$, $\phi(x_j)^\top\hat{\theta}_U$, and $\phi(x_j)^\top\hat{\theta}_V$ when q is large. Consequently, $\text{Var}(\tilde{y}_j)$, $\text{Var}(\hat{y}_{j,U})$, and $\text{Var}(\hat{y}_{j,V})$ serve as estimates for $\phi(x_j)^\top\text{Var}(\tilde{\theta}_n)\phi(x_j)$, $\phi(x_j)^\top\text{Var}(\hat{\theta}_U)\phi(x_j)$, and $\phi(x_j)^\top\text{Var}(\hat{\theta}_V)\phi(x_j)$, respectively. Therefore, the reduction in the variance of the estimators of θ can be directly assessed by evaluating the reduction in the variance of these predictions. We compare the variances $\text{Var}(\tilde{y}_j)$, $\text{Var}(\hat{y}_{j,U})$, and $\text{Var}(\hat{y}_{j,V})$ across all test points.

Figure 4 shows the variance differences across test points by plotting the standard quartile breakdown boxplots of $\{\text{Var}(\tilde{y}_j) - \text{Var}(\hat{y}_{j,U})\}_{j=1}^m$ and $\{\text{Var}(\tilde{y}_j) - \text{Var}(\hat{y}_{j,V})\}_{j=1}^m$ with $B = 50$, $n \in \{100, 150, 200, 250\}$, and $k \in \{10, 15, 20\}$. The results confirm that the variance reduction property holds across all settings for both U- and V-statistics-based experience replay methods. Appendix G.2 includes additional experiments with different B values and evaluations on a real-world dataset from the U.S. Census Bureau on Boston housing, further demonstrating effectiveness.

Table 1 presents the time cost reduction achieved by the

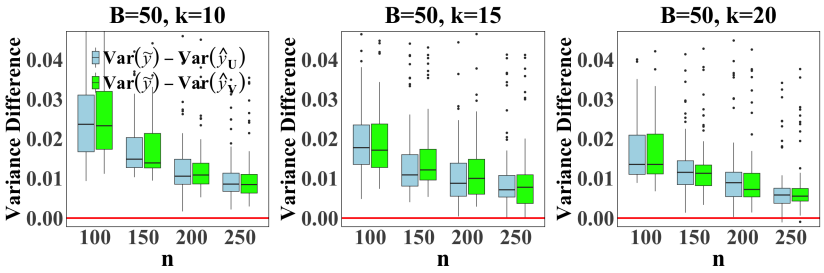


Figure 4: Variance differences in predicted outcomes using kernel ridge regression on the simulated data with $M = 100, m = 100$ and $B = 50$, evaluated across various values of n and k. $\tilde{y}$ represents the results without experience replay, while $\hat{y}_U$ and $\hat{y}_V$ represent the results with experience replay. The red line represents the baseline where the variance difference is 0.

	$k = 10$		$k = 15$		$k = 20$	
n	$t - t_U$	$t - t_V$	$t - t_U$	$t - t_V$	$t - t_U$	$t - t_V$
200	0.369	0.323	0.335	0.272	0.108	0.110
250	3.005	2.905	2.850	2.804	2.954	2.848

Table 1: Time cost reduction achieved by experience replay methods (measured in seconds) with $B = 50$ for different values of k and n.

experience replay methods with $B = 50$, $k \in \{10, 15, 20\}$, and $n \in \{200, 250\}$. Here, t represents the total time cost across all experiments without experience replay, while t_U and t_V represent the total time costs with experience replay based on resampled U- and V-statistics, respectively. Time cost was measured as wall-clock time on a single core without parallelization on a laptop with an Apple M2 Pro and 16 GB of RAM. The results demonstrate that, for a fixed B, the experience replay method reduces the computational cost in time, particularly when k is small and n is large. We also compare the RMSE of the proposed methods with the original method in Appendix H.2. The results indicate that incorporating experience replay, regardless of the specific resampling method used, not only reduces variance and time cost but also decreases prediction errors for all settings, especially in data-scarce scenarios.

While our theoretical results apply to both U- and V-statistics, empirical results show no major differences between them. In practice, V-statistics are often preferable due to their GPU-friendliness, ease of parallelization, and compatibility with modern machine learning frameworks.

5 Conclusion

Experience replay improves stability and efficiency in reinforcement learning, but its theoretical properties are still underexplored. This paper presents a theoretical framework that models experience replay using resampled U- and V-statistics, enabling us to establish variance reduction guarantees across policy evaluation and supervised learning tasks. We applied this framework to two policy evaluation algorithms—the LSTD method and a PDE-based model-free algorithm—demonstrating notable improvements in stability and accuracy, particularly in data-scarce settings. Additionally, we applied the framework to kernel ridge regression, achieving both significant computational savings and variance reduction. Future research could extend experience replay to federated and active learning settings. For example, using replay to improve communication efficiency and model personalization in federated learning, or selecting informative data subsets for replay in active learning, may address distributed data challenges.

Acknowledgments

We would like to thank the area chair, senior program committee, and five anonymous referees for constructive suggestions that improve the paper. Dai's research was supported in part by NIH grant R01DK142026, a Merck Research Award, and a Hellman Fellowship Award. Zhu's research was supported in part by NSF grant DMS-2529107 and a Hellman Fellowship Award.

References

Bradtke, S. J.; and Barto, A. G. 1996. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1): 33–57.

Dai, X.; Lyu, X.; and Li, L. 2023. Kernel knockoffs selection for nonparametric additive models. *Journal of the American Statistical Association*, 118(543): 2158–2170.

Duan, Y.; Wang, M.; and Wainwright, M. J. 2024. Optimal policy evaluation using kernel-based temporal difference methods. *The Annals of Statistics*, 52(5): 1927–1952.

Fedus, W.; Ramachandran, P.; Agarwal, R.; Bengio, Y.; Larochelle, H.; Rowland, M.; and Dabney, W. 2020. Revisiting fundamentals of experience replay. In *International Conference on Machine Learning*, 3061–3071. PMLR.

Frees, E. W. 1989. Infinite order U-statistics. *Scandinavian Journal of Statistics*, 29–45.

Hainmueller, J.; and Hazlett, C. 2014. Kernel regularized least squares: Reducing misspecification bias with a flexible and interpretable machine learning approach. *Political Analysis*, 22(2): 143–168.

Kober, J.; Bagnell, J. A.; and Peters, J. 2013. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11): 1238–1274.

Kong, J.; Pfeiffer, M.; Schildbach, G.; and Borrelli, F. 2015. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, 1094–1099. IEEE.

Lin, L.-J. 1992. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8: 293–321.

Mentch, L.; and Hooker, G. 2016. Quantifying uncertainty in random forests via confidence intervals and hypothesis tests. *Journal of Machine Learning Research*, 17(26): 1–41.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533.

Peng, W.; Coleman, T.; and Mentch, L. 2022. Rates of convergence for random forests via generalized U-statistics. *Electronic Journal of Statistics*, 16(1): 232–292.

Rahimi, A.; and Recht, B. 2007. Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems*, 20.

Schaul, T. 2015. Prioritized Experience Replay. *arXiv preprint arXiv:1511.05952*.

Shawe-Taylor, J.; and Cristianini, N. 2004. *Kernel methods for pattern analysis*. Cambridge University Press.

Shieh, G. S. 1994. Infinite order V-statistics. *Statistics & Probability Letters*, 20(1): 75–80.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587): 484–489.

Song, Y.; Chen, X.; and Kato, K. 2019. Approximating high-dimensional infinite-order U-statistics: Statistical and computational guarantees. *Electronic Journal of Statistics*, 13(2).

Stroock, D. W.; and Varadhan, S. S. 1997. *Multidimensional diffusion processes*, volume 233. Springer Science & Business Media.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Tu, S.; and Recht, B. 2018. Least-squares temporal difference learning for the linear quadratic regulator. In *International Conference on Machine Learning*, 5005–5014. PMLR.

Wahba, G. 1990. *Spline models for observational data*. SIAM.

Wood, S. N. 2003. Thin plate regression splines. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 65(1): 95–114.

Zhang, S.; and Sutton, R. S. 2017. A deeper look at experience replay. *arXiv preprint arXiv:1712.01275*.

Zhang, Y.; Duchi, J.; and Wainwright, M. 2013. Divide and conquer kernel ridge regression. In *Conference on Learning Theory*, 592–617. PMLR.

Zhou, Z.; Mentch, L.; and Hooker, G. 2021. V-statistics and variance estimation. *Journal of Machine Learning Research*, 22(287): 1–48.

Zhu, Y. 2024. PhiBE: A PDE-based Bellman Equation for Continuous Time Policy Evaluation. *arXiv preprint arXiv:2405.12535*.