Reducing Sampling Error in Batch Temporal Difference Learning

Brahma S. Pavse [1] **Ishan Durugkar** [1] **Josiah P. Hanna** [2 3] **Peter Stone** [1 4]

Abstract

Temporal difference (TD) learning is one of the main foundations of modern reinforcement learning. This paper studies the use of TD(0), a canonical TD algorithm, to estimate the value function of a given policy from a batch of data. In this batch setting, we show that TD(0) may converge to an inaccurate value function because the update following an action is weighted according to the number of times that action occurred in the batch – not the true probability of the action under the given policy. To address this limitation, we introduce *policy sampling error corrected*-TD(0) (PSEC-TD(0)). PSEC-TD(0) first estimates the empirical distribution of actions in each state in the batch and then uses importance sampling to correct for the mismatch between the empirical weighting and the correct weighting for updates following each action. We refine the concept of a certainty-equivalence estimate and argue that PSEC-TD(0) is a more data efficient estimator than TD(0) for a fixed batch of data. Finally, we conduct an empirical evaluation of PSEC-TD(0) on three batch value function learning tasks, with a hyperparameter sensitivity analysis, and show that PSEC-TD(0) produces value function estimates with lower mean squared error than TD(0).

1. Introduction

Reinforcement learning (RL) (Sutton & Barto, 2018) algorithms have been applied to a variety of sequential-decision making problems such as robot manipulation (Kober et al., 2013; Gu et al., 2016) and autonomous driving (Sallab et al., 2017). Many RL algorithms learn an optimal control policy by estimating the *value function*, a function that gives the expected return from each state when following a particular policy (Puterman & Shin, 1978; Bertsekas, 1987; Konda & Tsitsiklis, 2000). These algorithms require accurate value function estimation with finite data. A fundamental approach to value function learning is the temporal difference (TD) algorithm (Sutton, 1988).

In this work, we focus on improving the accuracy of the value function learned by *batch* TD, where TD updates for a value function are computed from a fixed batch of data. We show that batch TD(0) may converge to an inaccurate value function since it ignores the known action probabilities of the policy it is evaluating. For example, consider a single state in which the evaluation policy selects between action a_1 or a_2 with probability 0.5. If, in the finite batch of observed data, a_1 actually happens to occur twice as often as a_2 then TD updates following a_1 will receive twice as much weight as updates following a_2, even though in expectation they should receive the same weight. We describe this finite-sample error in the value function estimate as *policy sampling error*. To correct for policy sampling error we propose to first estimate the maximum likelihood policy from the observed data and then use importance sampling (Precup et al., 2000a) to account for the mismatch between the frequency of sampled actions and their true probability under the evaluation policy. Variants of this technique have been successful in multi-armed bandits (Li et al., 2015; Narita et al., 2018; Xie et al., 2018), policy evaluation (Hanna et al., 2019), and policy gradient learning (Hanna & Stone, 2019). However, we are the first to show that this technique can be used to correct for policy sampling error in value function estimation and the first to show the benefit of importance sampling in on-policy value function estimation. We show that by using the available policy information, our approach is more data efficient than vanilla batch TD(0). We call our new value function learning algorithm batch *policy sampling error corrected-TD(0) (PSEC-TD(0))*.

The contributions of the paper are the following:

1. Show that the fixed point that batch TD(0) converges to for a given policy is inaccurate with respect to the true value function.
2. Introduce the batch PSEC-TD(0) algorithm that reduces the policy sampling error in batch TD(0).
3. Refine the concept of a certainty-equivalence estimate

[1]The University of Texas at Austin [2]School of Informatics, University of Edinburgh [3]To be joining the Computer Sciences Department, University of Wisconsin–Madison [4]Sony AI. Correspondence to: Brahma S. Pavse <brahmasp@cs.utexas.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

for TD(0) (Sutton, 1988) and provide theoretical justification that batch PSEC-TD(0) is more data efficient than batch TD(0).

4. Empirically analyze batch PSEC-TD(0) in the tabular and function approximation setting.

2. Background

This section introduces notation and formally specifies the batch value function learning problem.

2.1. Notation and Definitions

Following the standard MDPNv1 notation (Thomas, 2015), we consider a Markov decision process (MDP) with state space $\mathcal{S}$, action space $\mathcal{A}$, reward function R, transition dynamics function P, and discount factor γ (Puterman, 2014). In any state s, an agent selects stochastic actions according to a policy π, $a \sim \pi(\cdot|s)$. After taking an action a in state s the agent transitions to a new state $s' \sim P(\cdot|s,a)$ and receives reward $R(s,a,s')$. We assume $\mathcal{S}$ and $\mathcal{A}$ to be finite; however, our experiments also consider infinite sized $\mathcal{S}$ and $\mathcal{A}$. We consider the episodic, discounted, and finite horizon setting. The policy and MDP jointly induce a *Markov reward process* (MRP), in which the agent transitions between states s and s' with probability $P(s'|s)$ and receives reward $R(s,s')$. Finally, $\boldsymbol{x}(s): \mathcal{S} \to \mathbb{R}^d$ gives a column feature vector for each state $s \in \mathcal{S}$.

We are concerned with computing the value function, $v^\pi : \mathcal{S} \to \mathbb{R}$, that gives the *value* of any state. The value of a particular state is the expected discounted return, i.e. the expected sum of discounted rewards when following policy π from that state:

$$v^\pi(s) := \mathbf{E}_\pi\left[\sum_{k=0}^{L} \gamma^k R_{t+k+1} \,\middle|\, s_t = s\right], \forall s \in \mathcal{S} \tag{1}$$

where L is the terminal time-step and the expectation is taken over the distribution of future states, actions, and rewards under π and P.

2.2. Batch Value Prediction

This work investigates the problem of approximating v^{π_e} given a batch of data, $\mathcal{D}$, and an evaluation policy, π_e. Let a single episode, τ, be defined as $\tau := (s_0, a_0, r_0, s_1, ..., s_{L_\tau - 1}, a_{L_\tau - 1}, r_{L_\tau - 1})$, where L_τ is the length of the episode τ. The batch of data consists of m episodes, i.e., $\mathcal{D} := \{\tau_i\}_{i=0}^{m-1}$. The policy that generated the batch of data is called the *behavior policy*, π_b. If π_b is the same as π_e for all episodes then learning is said to be done *on-policy*; otherwise it is *off-policy*.

In batch value prediction, a value function learning algorithm uses a fixed batch of data to learn an estimate $\hat{v}^{\pi_e}$ that approximates the true value function, v^{π_e}. In this work, we introduce algorithmic and theoretical concepts with the linear approximation of v^{π_e}:

$$\hat{v}^{\pi_e}(s) := \boldsymbol{w}^T \boldsymbol{x}(s)$$

thus, in the linear case, we seek to find a weight vector $\boldsymbol{w}$, such that $\boldsymbol{w}^T\boldsymbol{x}(s)$ approximates the true value, $v^{\pi_e}(s)$. However, our empirical study also considers the non-linear approximation of v^{π_e}. The error of the predicted value function, $\hat{v}^{\pi_e}$, with respect to the true value function, v^{π_e}, is measured by calculating the mean squared value error between $v^{\pi_e}(s)$ and $\hat{v}^{\pi_e}(s)$ $\forall s \in \mathcal{S}$ weighted by the proportion of time spent in each state under policy π_e, $d_{\pi_e}(s)$. Thus, we seek to find a weight vector $\boldsymbol{w}$ that minimizes:

$$\text{MSVE}(\boldsymbol{w}) := \sum_{s \in \mathcal{S}} d_{\pi_e}(s)\Big(v^{\pi_e}(s) - \boldsymbol{w}^T\boldsymbol{x}(s)\Big)^2 \tag{2}$$

In this work, we compare data efficiency between two algorithms, X and Y, as follows:

Definition 1. *Data Efficiency. A prediction algorithm X is more data efficient than algorithm Y if estimates from X have, on average, lower MSVE than estimates from Y for a given batch size.*

2.3. Batch Linear TD(0)

A fundamental algorithm for value prediction is the single-step temporal difference learning algorithm, TD(0). Algorithm 1 gives pseudo-code for the batch linear TD(0) algorithm described by Sutton (1988).

Algorithm 1 Batch Linear TD(0) to estimate v^{π_e}

Input: policy to evaluate π_e, behavior policy π_b, batch $\mathcal{D}$, linear value function, $\hat{v} : \mathcal{S} \times \mathbb{R}^d \to \mathbb{R}$, step-size $\alpha > 0$, convergence threshold $\Delta > 0$
Initialize: weight vector $\boldsymbol{w}_0$ arbitrarily (e.g.: $\boldsymbol{w}_0 := \mathbf{0}$), aggregation vector $\boldsymbol{u} := \mathbf{0}$, batch process counter, $i = 0$
while $|\boldsymbol{w}_{i+1} - \boldsymbol{w}_i| \geq \mathbf{1} \cdot \Delta$ **do**
    for each episode, $\tau \in \mathcal{D}$ **do**
        for each transition, $(s, a, r, s') \in \tau$ **do**
            $\hat{y} \leftarrow r + \gamma \boldsymbol{w}_i^T \boldsymbol{x}(s')$
            $\rho \leftarrow \frac{\pi_e(a|s)}{\pi_b(a|s)}$ {for on-policy, $\pi_b = \pi_e$}
            $\boldsymbol{u} \leftarrow \boldsymbol{u} + \left[\rho\hat{y} - \boldsymbol{w}_i^T\boldsymbol{x}(s)\right]\boldsymbol{x}(s)$
        end for
    end for
    $\boldsymbol{w}_{i+1} \leftarrow \boldsymbol{w}_i + \alpha \boldsymbol{u}$ {batch update}
    $\boldsymbol{u} \leftarrow \mathbf{0}$ {clear aggregation}
    $i \leftarrow i + 1$
end while

Sutton (1988) proved that batch linear TD(0) converges to a fixed point in the on-policy case i.e. when $\pi_e = \pi_b$. An off-policy batch TD(0) algorithm uses importance sampling ratios to ensure that the expected update is the same as it would be if actions were taken with π_e instead of π_b (Precup et al., 2000a). Unlike on-policy TD(0), off-policy TD(0) is *not* guaranteed to converge (Baird, 1995).

3. Convergence of Batch Linear TD(0)

In this section, we discuss the convergence of batch linear TD(0) to a fixed-point, the *certainty equivalence estimate* (CEE) for the underlying Markov reward process (MRP). We refine this concept to better reflect our objective of evaluating a policy in an MDP and then prove that batch TD(0) converges to an equivalent fixed point that ignores knowledge of the known evaluation policy, π_e, leading to inaccuracy in the value function estimate. This result motivates our proposed algorithm.

First, we introduce additional notation and assumptions. In this section, we assume that we are in the on-policy setting ($\pi_b = \pi_e$). Let $\widehat{\mathcal{S}}$ be the set of states and $\widehat{\mathcal{A}}$ be the set of actions that appear in $\mathcal{D}$ and let $\bar{R}(s)$ be the mean reward received when transitioning from state s in the batch $\mathcal{D}$. Finally, if the notation includes a hat ($\hat{}$), it is the maximum-likelihood estimate (MLE) according to $\mathcal{D}$. For example, $\hat{\pi}$ is the MLE of π_b. Sutton (1988) proved that batch linear TD(0) converges to the *CEE*. That is, it converges to the exact value function of the maximum likelihood MRP according to the observed batch. This exact value function can be calculated using dynamic programming (Bellman, 2003; Bertsekas, 1987) with the MLE MRP transition function. We call this value function estimate the *Markov reward process certainty equivalence estimate* (MRP-CEE).

Definition 2. *Markov Reward Process Certainty Equivalence Estimate (MRP-CEE) Value Function. The MRP-CEE is the value function $\hat{v}_{\texttt{MRP}}$ that, $\forall s, s' \in \widehat{\mathcal{S}}$, satisfies:*

$$\hat{v}_{\texttt{MRP}}(s) = \bar{R}(s) + \gamma \sum_{k \in \widehat{\mathcal{S}}} \widehat{P}(s'|s)\hat{v}_{\texttt{MRP}}(s'). \tag{3}$$

Having now defined the MRP-CEE value function, we prove that batch TD(0) converges to the MRP-CEE value function. This fact was first proven by Sutton (1988) (see Theorem 3 of Sutton (1988)), however the original proof only considers rewards upon termination and no discounting. The extension to rewards per-step and discounting is straightforward, but to the best of our knowledge has not appeared in the literature before. Following Sutton's proof (Sutton, 1988), we first prove the extension before extending the proof to an MDP, where the data inefficiency of TD(0) becomes clear. Proof details are in Appendix C.

Theorem 1 (Batch Linear TD(0) Convergence)**.** *For any batch whose observation vectors $\{\boldsymbol{x}(s)|s \in \widehat{\mathcal{S}}\}$ are linearly independent, there exists an $\epsilon > 0$ such that, for all positive $\alpha < \epsilon$ and for any initial weight vector, the predictions for linear TD(0) converge under repeated presentations of the batch with weight updates after each complete presentation to the fixed-point (3).*

In RL, the transitions of an MRP are a function of the behavior policy *and* transition dynamics distributions. That is $\forall s, s' \in \widehat{\mathcal{S}}$:

$$\widehat{P}(s'|s) = \sum_{a \in \widehat{\mathcal{A}}} \hat{\pi}(a|s)\widehat{P}(s'|s,a),$$
$$\bar{R}(s) = \sum_{a \in \widehat{\mathcal{A}}} \hat{\pi}(a|s)\bar{R}(s,a)$$

where $\bar{R}(s,a)$ is the mean reward observed in state s on taking action a. We define a new certainty-equivalence estimate that separates these two factors. We call this new value function estimate the *Markov decision process certainty equivalent estimate* (MDP-CEE).

Definition 3. *Markov Decision Process Certainty Equivalence Estimate (MDP-CEE) Value Function. The MDP-CEE is the value function, $\hat{v}^{\hat{\pi}}_{\texttt{MDP}}$, that, $\forall s, s' \in \widehat{\mathcal{S}}$, satisfies:*

$$\hat{v}^{\hat{\pi}}_{\texttt{MDP}}(s) = \sum_{a \in \widehat{\mathcal{A}}} \hat{\pi}(a|s) \left(\bar{R}(s,a) + \gamma \sum_{s' \in \widehat{\mathcal{S}}} \widehat{P}(s'|s,a)\hat{v}^{\hat{\pi}}_{\texttt{MDP}}(s') \right) \tag{4}$$

Given the definitions of $\widehat{P}$ and $\bar{R}$, the MRP-CEE (Definition 2) and MDP-CEE (Definition 3) are equivalent. Theorem 2 gives the convergence of batch TD(0) to the MDP-CEE value function. Proof details are in Appendix D.

Theorem 2 (Batch Linear TD(0) Convergence)**.** *For any batch whose observation vectors $\{\boldsymbol{x}(s)|s \in \widehat{\mathcal{S}}\}$ are linearly independent, there exists an $\epsilon > 0$ such that, for all positive $\alpha < \epsilon$ and for any initial weight vector, the predictions for linear TD(0) converge under repeated presentations of the batch with weight updates after each complete presentation to the fixed-point (4).*

The MDP-CEE value function highlights two sources of estimation error in the value function estimate: $P \neq \widehat{P}$ and/or $\pi_e \neq \hat{\pi}$. We describe the former as *transition sampling error* and the latter as *policy sampling error*. Transition sampling error may be unavoidable in a model-free setting since we do not know P. However, we do know π_e and can use this knowledge to potentially correct policy sampling error. In the next section, we present an algorithm that uses the knowledge of π_e to correct for policy sampling error and obtain a more accurate value function estimate.

4. Batch Linear PSEC-TD(0)

In this section, we introduce the batch policy sampling error corrected-TD(0) (PSEC-TD(0)) algorithm that corrects for the policy sampling error in batch TD learning. From Theorem 2, batch TD(0) converges to the value function for the maximum likelihood policy, $\hat{\pi}$, instead of π_e. Under this view, PSEC-TD(0) treats policy sampling error as an off-policy learning problem and uses importance sampling (Precup et al., 2000a) to correct the weighting of TD(0) updates from $\hat{\pi}$ to π_e. Even though importance sampling is usually associated with off-policy learning, this approach is applicable in the on- *and* off-policy cases.

In addition to $\mathcal{D}$ and π_e, we assume we are given a set of policies, Π. Batch PSEC-TD(0) first computes the maximum likelihood estimate of the behavior policy:

$$\hat{\pi} := \operatorname*{argmax}_{\pi' \in \Pi} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{L_{\tau-1}} \log \pi'(a_t^\tau | s_t^\tau)$$

This estimation can be done in a number of ways. For example, in the tabular setting we could use the empirical count of actions in each state. This count-based approach is often intractable, and hence, in many problems of interest we must rely on function approximation. When using function approximation, the policy estimate can be obtained by minimizing a negative log-likelihood loss function. Once $\hat{\pi}$ is computed, the batch PSEC-TD algorithm is the same as Algorithm 1 with $\hat{\pi}$ replacing π_b in the importance sampling ratio. That is, for transition (s, a, r, s') in $\mathcal{D}$, the contribution to the weight update is $\boldsymbol{u} \leftarrow \boldsymbol{u} + \left[\hat{\rho}\hat{y} - \boldsymbol{w}_i^T \boldsymbol{x}(s)\right] \boldsymbol{x}(s)$, where $\hat{\rho} := \frac{\pi_e(a|s)}{\hat{\pi}(a|s)}$ is the PSEC weight (refer to Line 8 in Algorithm 1). Thus, PSEC makes an importance sampling correction from the *empirical* to the *evaluation* policy distribution.

4.1. Convergence of Batch Linear PSEC-TD(0)

Section 3 showed that batch TD(0) converges to two equivalent certainty-equivalence estimates. We now define a new certainty-equivalent estimate (CEE) to which our new batch PSEC-TD(0) algorithm converges. Intuitively, the MDP-CEE estimate (Definition 3) is the exact value function for the *MLE of the behavior policy,* $\hat{\pi}$, in the MLE of the MDP environment; our new algorithm converges to the exact value function for π_e in the MLE of the MDP environment, making it more data efficient than batch TD(0) once the batch size is large enough.

We define this new CEE as the *PSEC Markov Decision Process Certainty Equivalence Estimate* (PSEC-MDP-CEE) Value Function.

Definition 4. *PSEC Markov Decision Process Certainty Equivalence Estimate (PSEC-MDP-CEE) Value Function. The PSEC-MDP-CEE is the value function, $\hat{v}^{\pi_e}_{\mathtt{PSEC-MDP}}$, that, $\forall s, s' \in \widehat{\mathcal{S}}$, satisfies:*

$$\begin{aligned}\hat{v}^{\pi_e}_{\mathtt{PSEC-MDP}}(s) = \sum_{a \in \widehat{\mathcal{A}}} \pi_e(a|s)[\bar{R}(s,a) \\ + \gamma \sum_{k \in \widehat{\mathcal{S}}} \widehat{P}(s'|s,a)\hat{v}^{\pi_e}_{\mathtt{PSEC-MDP}}(s'))]\end{aligned} \tag{5}$$

Theorem 3 states that batch PSEC-TD(0) converges to the new PSEC-MDP-CEE value function (Equation 5). Proof details are in Appendix E.

Theorem 3 (Batch Linear PSEC-TD(0) Convergence)**.** *For any batch whose observation vectors $\{\boldsymbol{x}(s) | s \in \widehat{\mathcal{S}}\}$ are linearly independent, there exists an $\epsilon > 0$ such that, for all positive $\alpha < \epsilon$ and for any initial weight vector, the predictions for linear PSEC-TD(0) converge under repeated presentations of the batch with weight updates after each complete presentation to the fixed-point (5).*

We remark that convergence has only been shown for the on-policy setting. While PSEC-TD(0) can be applied in the off-policy setting, it may, like other semi-gradient TD methods, diverge when off-policy updates are made with function approximation (Baird, 1995). It is possible that combining PSEC-TD(0) with Emphatic TD (Mahmood et al., 2015) or Gradient-TD (Sutton et al., 2009) may result in provably convergent behavior with off-policy updates, however, that study is outside the scope of this work.

4.2. Extending PSEC to other TD Variants

In general, PSEC can improve any value function learning algorithm that computes the TD-error, δ, or equivalent errors. As an example, we consider the off-policy least-squares TD (LSTD) algorithm (Bradtke & Barto, 1996; Ghiassian et al., 2018), which analytically computes the exact parameters that minimize the TD-error in a batch of data using the following steps:

$$\begin{aligned}A &= \sum_{(s,a,s') \in \mathcal{D}} \left[\hat{\rho}\boldsymbol{x}(s)(\boldsymbol{x}(s) - \gamma\boldsymbol{x}(s'))^T\right] \\ \mathbf{b} &= \sum_{(s,a,s') \in \mathcal{D}} R(s,a,s')\boldsymbol{x}(s) \\ \boldsymbol{w} &= A^{-1}\mathbf{b},\end{aligned}$$

where $\hat{\rho}$ is the PSEC weight. Even though we primarily consider TD(0) in this work, the extension to LSTD demonstrates that PSEC-TD can be extended to other value function learning algorithms.

5. Empirical Study

In this section, we empirically study PSEC-TD to answer the following questions:

1. Does batch PSEC-TD(0) lower MSVE compared to batch TD(0)?
2. Does batch linear PSEC-TD(0) empirically converge to its certainty-equivalence solution?
3. Does PSEC yield benefit when applied to LSTD?
4. What factors does PSEC's data efficiency depend on in the function approximation setting?

We briefly describe the RL domains used in our experiments.

- **Gridworld:** In this domain, an agent navigates a 4×4 grid to reach a corner. The *state and action spaces are* *discrete* and we use a tabular representation for $\hat{v}^{\pi_e}$. PSEC-TD(0) uses count-based estimation for $\hat{\pi}$. The ground truth value function is computed with dynamic programming and the MSVE computation uniformly weights the error in each state. In Section 5.1, we consider a *deterministic* gridworld, where there is no transition dynamics sampling error.

- **CartPole:** In this domain, an agent controls a cart to balance a pole upright. The *state space is continuous and action space is discrete*. We only consider the on-policy setting. The evaluation policy is a neural network trained using REINFORCE (Williams, 1992). It has 2 hidden layers with 16 neurons. We evaluate PSEC with varying linear and neural network representations for the value function. $\hat{\pi}$ maps the raw state features to a softmax distribution over the actions with varying linear and neural network architectures. Since the true value function is unknown, we follow Pan et al. (2016) and use Monte Carlo rollouts from a fixed number of states sampled from episodes following the evaluation policy to approximate the ground-truth state-values of those states. We then compute the MSVE between the learned values and the average Monte Carlo return from these sampled states.

- **InvertedPendulum:** This domain is similar to CartPole, and the objective is the same – to balance a pole upright. However, the *state and action spaces are both continuous*. We only consider the on-policy setting. The evaluation policy is a neural network trained by PPO (Schulman et al., 2017). The network has 2 hidden layers with 64 neurons each. We evaluate PSEC with varying linear and neural network representations for the value function. The $\hat{\pi}$ estimate consists of two components: 1) a linear or neural network mapping from raw state features to the mean vector of a Gaussian distribution, and 2) parameters representing the log standard deviation of each element of the output vector. As in CartPole, we compute Monte Carlo rollouts for sampled states.

In all experiments, the value function learning algorithm iterates over the whole batch of data until convergence, after which the MSVE of the final value function is computed. Some experiments include a parameter sweep over the hyperparameters, which can be found in Appendix G.

5.1. Tabular Setting

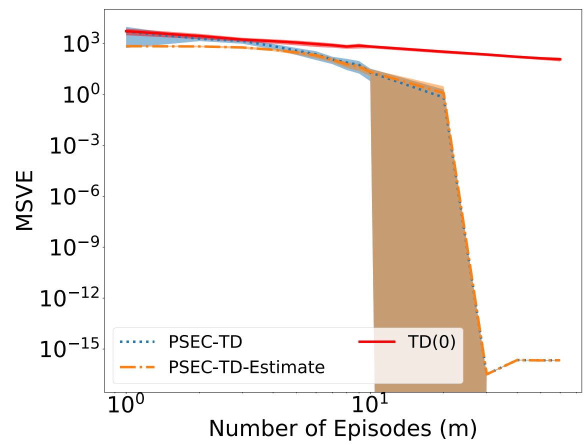
In this set of experiments, we consider two variants of PSEC-TD that differ in the placement of the PSEC weight:

- PSEC-TD-Estimate: Multiplies $\hat{\rho}$ by the new estimate: $\hat{y} = R + \gamma \boldsymbol{w}^T \boldsymbol{x}(s')$.
- PSEC-TD: Multiplies $\hat{\rho}$ by the TD error: $\delta = (R + \gamma \boldsymbol{w}^T \boldsymbol{x}(s')) - \boldsymbol{w}^T \boldsymbol{x}(s)$.

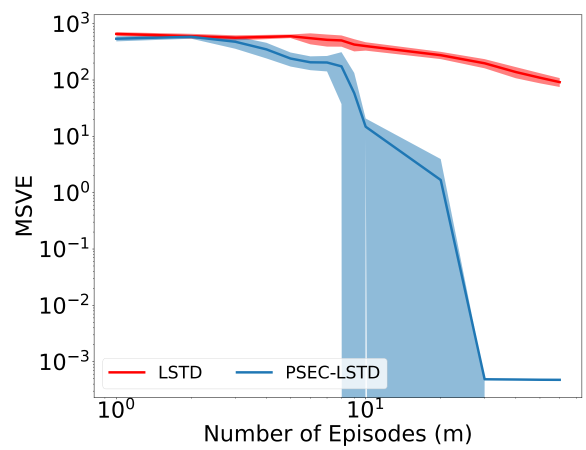
For off-policy TD(0), these placements of $\hat{\rho}$ are equivalent in expectation although the method using the TD-error has been reported to perform better in practice (Ghiassian et al., 2018). In this section, we focus on the on-policy results. Appendix F.1 includes off-policy results.

5.1.1. Data Efficiency

Figure 1 answers our first and third empirical questions, and shows that PSEC lowers MSVE compared to batch TD(0), and a variant of TD(0), LSTD(0). The gap between PSEC and its TD counterpart increases dramatically with more data; we discuss this observation in Section 5.1.2.



(a) On-policy (PSEC-)TD(0)

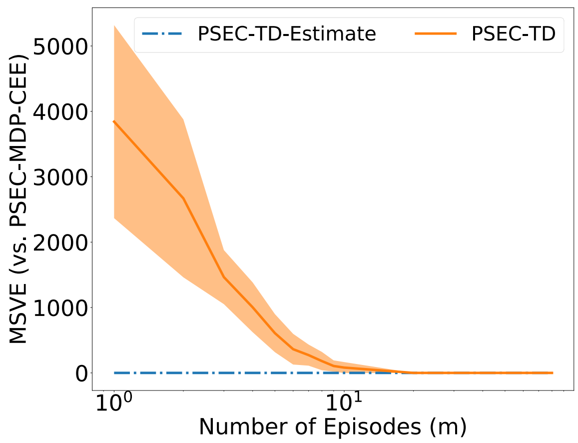


(b) On-Policy (PSEC-)LSTD(0)

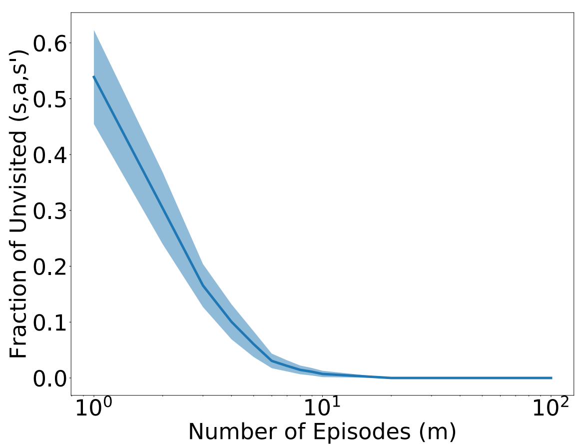
Figure 1. Deterministic Gridworld experiments. Both axes are log-scaled. Errors are computed over 200 trials with 95% confidence intervals. Asymmetric confidence intervals are due to log-scaling. Figure 1(a) and Figure 1(b) compare the data efficiency of PSEC-TD(0) and PSEC-LSTD(0) with their respective TD equivalents. Lower MSVE is better.

5.1.2. Convergence to the PSEC-Certainty-Equivalence

To address our second empirical question, we empirically verify that both variants of batch linear PSEC, PSEC-TD and PSEC-TD-Estimate, converge to the dynamic programming computed PSEC-MDP-CEE value function (5) in Gridworld. According to Theorem 3, batch linear PSEC-TD-Estimate converges to the fixed-point (5) for all batch sizes.



(a)



(b)

Figure 2. Additional Gridworld experiments. Errors are computed over 50 trials with 95% confidence intervals. Figure 2(a) shows MSVE achieved by variants of linear batch PSEC-TD(0), PSEC-TD and PSEC-TD-Estimate, with respect to the PSEC-MDP-CEE (5). Figure 2(b) shows the fraction of unvisited (s, a, s') tuples.

We also empirically confirm that the other variant of PSEC, PSEC-TD converges to the same fixed-point (5) when the following condition holds true: only when all non-zero probability actions for each state in the batch have been sampled at least once. We note that when this condition is false, PSEC-TD-Estimate treats the value of taking that action as 0. For example, if a state, s, appears in the batch and an action, a, that could take the agent to state s' does not appear in the batch, then PSEC-TD-Estimate treats the new estimate $R + \gamma \boldsymbol{w}^T \boldsymbol{x}(s')$ as 0, which is also done by the dynamic programming computation (5). We note that PSEC-TD converges to the fixed-point (5) only when this condition is true since the PSEC weight requires a fully supported probability distribution when applied to the TD-error estimate. From Figure 2(a) and Figure 2(b), we can see that this condition holds at batch size of 10 episodes. We also note that PSEC-TD(0) corrects policy sampling error for each (s, a, s') transition. Thus, when all such transitions are visited, PSEC fully corrects for all policy sampling error, which occurs at batch size of 10 episodes in this *deterministic* gridworld.

5.2. Function Approximation Setting

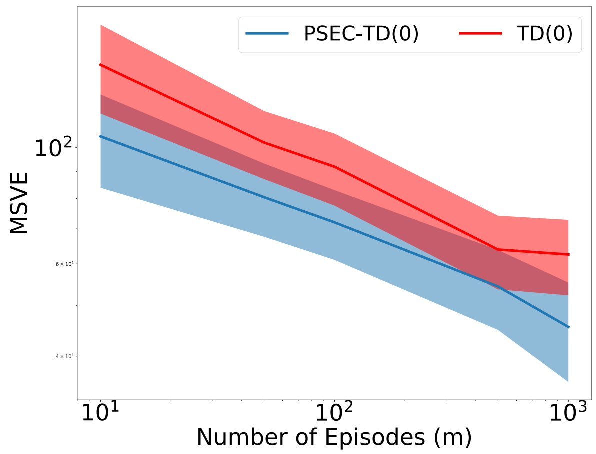
In this set of experiments, we answer our first and fourth empirical questions concerning function approximation in PSEC. Our experiments focus on applying only the second variant of PSEC, PSEC-TD, since we found that PSEC-TD-Estimate diverges. The results shown below are for the on-policy case. In addition to results of PSEC as a function of data size, we conduct experiments on a fixed batch size to better understand how components of the PSEC training process impact performance. Finally, we give a practical recommendation for use of batch PSEC-TD(0).

In these experiments, we have three function approximators: one for the value function; one to estimate the behavior policy; and the pre-learned behavior policy itself. When any are referred to as “fixed", it means its architecture is unchanged. Due to space constraints, we only show a subset of results from CartPole and Inverted Pendulum; however, a fuller set of experiments can be found in Appendix F.2 and F.3. Note that in all PSEC training settings, PSEC performs gradient steps using the full batch of data, uses a separate batch of data as the validation data, and terminates training according to early stopping. Statistical significance is determined by Welch's test (Welch, 1947) with a significance level of 0.05. For hyperparameter details refer to Appendix F.2 and F.3.

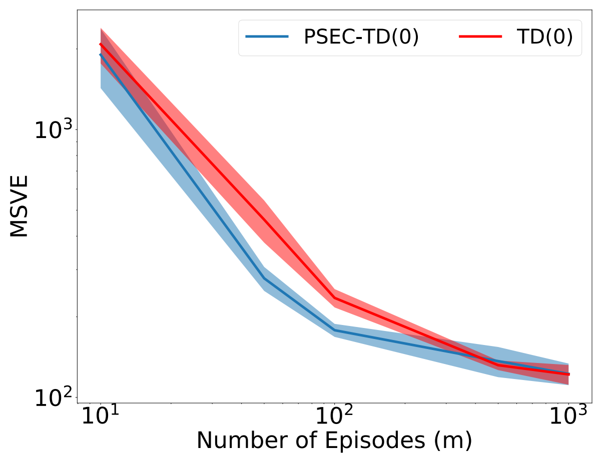
5.2.1. Data Efficiency

In CartPole, PSEC produced statistically significant improvement over TD in all batch sizes except 500. In InvertedPendulum, like in Gridworld, the improvement was marginal for smaller batch sizes, but produced statistically significant improvement with larger batch sizes. As data gets larger, we observe that both methods perform similarly for two reasons: 1) the PSEC weight approaches 1, which effectively becomes TD(0) and 2) saturation in value function representation capacity, which we discuss in Section

5.2.2. Note that while a thorough parameter sweep can achieve better performance, it is computationally expensive. The results shown here are with sweeps over only the value function model class and PSEC learning rate.



(a) CartPole

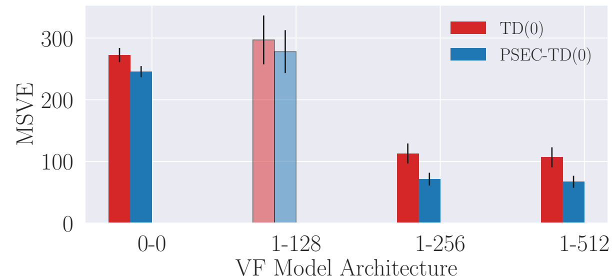


(b) InvertedPendulum

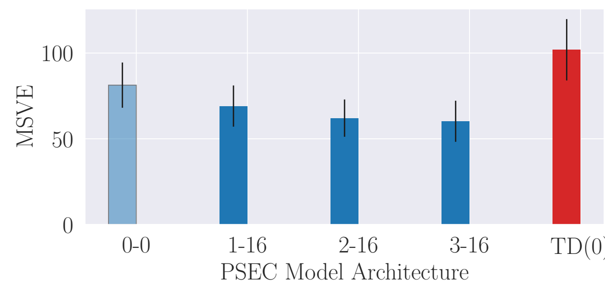
Figure 3. Comparing data efficiency of PSEC and TD on different batch sizes. Results for Figure 3(a) and Figure 3(b) are averaged over 400 and 250 trials resp. with shaded region of 95% confidence. Both axes are log-scaled. Lower MSVE is better.

5.2.2. Architecture Model Selection

Figure 4(a) illustrates the impact of different value function classes on the data efficiency of TD and PSEC, while holding the PSEC model and behavior policy architectures fixed, on CartPole. We generally found that more expressive value function representations resulted in better data efficiency by both algorithms. We also found that the gap between PSEC and TD increased as the VF representation became more expressive. We hypothesize that even though PSEC finds a more accurate fixed point than TD in the space of all value functions, the shown difference between the two algorithms is dependent on the space of representable value functions – a more representable function class can capture the difference between the two algorithms better. The lighter shades mean that any difference between PSEC and TD was statistically insignificant.



(a)



(b)

Figure 4. Figure 4(a) and Figure 4(b) compare data efficiency of PSEC, with varying VF model architectures, and PSEC, with varying model arch, respectively against TD on CartPole. Both use a batch size of 10 episodes, and results shown are averaged over 300 trials with error bars of 95% confidence. Darker shades represent statistically significant results. The label on the x axis shown is (# hidden layers - # neurons). Lower MSVE is better.

Figure 4(b) compares the data efficiency of PSEC against TD with varying PSEC neural network model architectures, while the value function and behavior policy architectures are fixed, on CartPole. In general, we found that more expressive network models produced better PSEC weights since they were able to better capture the MLE of the policy from the data. Unlike the NN PSEC policies, the linear function PSEC policy did not produce a statistically significant improvement over TD.

5.2.3. Sensitivity Studies

Due to space limitations, we defer the empirical analysis of other effects to Appendices F.2 and F.3. Figure 7 and Figure 12 indicate that a small learning rate for the PSEC model is preferred. Figure 9 and 14 indicate that some overfitting by the PSEC model is tolerable, and perhaps, preferable, but extreme overfitting can degrade performance.

Practical Recommendation Based on our experiments, we recommend the following: 1) an expressive value function that can represent the more accurate fixed-point of PSEC-TD, 2) a PSEC model class that can represent the true behaviour policy but with awareness that extreme overfitting may hamper performance, and 3) a small learning rate.

6. Related Work

In this section, we discuss the literature on importance sampling with an estimated behavior policy and reducing sampling error in reinforcement learning.

The approach in this work has been motivated by prior work showing that importance sampling with an estimated behavior policy can lower variance when estimating an expected value in RL. Hanna et al. (2019) introduce a family of methods called regression importance sampling methods (RIS) and show that they have lower variance than importance sampling with the true behavior policy. Hanna & Stone (2019) show that a similar technique led to more sample-efficient policy gradient learning. These works are related to work in the multi-armed bandit (Li et al., 2015; Narita et al., 2018; Xie et al., 2018), causal inference (Hirano et al., 2003; Rosenbaum, 1987), and Monte Carlo integration (Henmi et al., 2007; Delyon & Portier, 2016) literature. In contrast, our work focuses on *value function learning*, where the focus is on learning the expected return at every state visited by the agent instead of across a set of actions (multi-armed bandit) or for some start states that are a subset of all the states the agent visits.

PSEC-TD(0) corrects policy sampling error through importance sampling with an estimated behavior policy. Other works avoid policy sampling error entirely by computing analytic expectations. Expected SARSA (van Seijen et al., 2009), learns action-values by analytically computing the expected return of the next state during bootstrapping as opposed to using the value of the sampled next action. The Tree-backup algorithm (Precup et al., 2000b) extends Expected SARSA to a multi-step algorithm. $Q(\sigma)$ (Asis et al., 2017) unifies SARSA (Sutton, 1996; Rummery & Niranjan, 1994), Expected SARSA, and Tree-backups, to find a balance between sampling and analytic expectation computation. Our work is distinct from these in that we focus on learning state values which may be preferable for prediction as well as a variety of actor-critic approaches (Konda & Tsitsiklis, 2000; Mnih et al., 2016). To the best of our knowledge, no other approach exists for correcting policy sampling error when learning state values.

7. Summary and Discussion

In batch value function approximation, we observed that TD(0) may converge to an inaccurate estimate of the value function due to policy sampling error. We proposed batch PSEC-TD(0) as a method to correct this error and showed that it leads to a more data efficient estimator than batch TD(0). In this paper, we theoretically analyzed PSEC-TD and empirically evaluated it in the tabular and function approximation settings. Our empirical study validated that PSEC converges to a more accurate fixed point than TD, and studied how the numerous components in the PSEC training setup impact its data efficiency with respect to TD.

Despite the data efficiency benefits that batch PSEC-TD(0) introduced, there are limitations. First, it requires knowledge of the evaluation policy, which on-policy TD(0) does not. This comparative disadvantage is only for the on-policy setting as both TD(0) and PSEC-TD(0) require knowledge of the evaluation policy for the off-policy setting. Additionally, PSEC-TD(0), in the off-policy case, has the advantage of not requiring knowledge of the behavior policy π_b. Second, the policy estimation step required by PSEC-TD(0) could potentially be computationally expensive. For instance, requiring the computation and storage of $\mathcal{O}(|\mathcal{S}||\mathcal{A}|)$ parameters in the tabular setting.

There are several directions for future work. First, our work focused on batch TD(0). We expect that a variant of PSEC can improve value function learning with n-step TD and TD(λ). Second, with an improved value function learning algorithm, it would be interesting to see if an agent can learn better control policies. Third, it would be interesting to theoretically and empirically study PSEC when learning the state-action values. Finally, automatically finding the optimal training setting for PSEC in the function approximation setting is another important direction for future work.

Acknowledgements

We thank Darshan Thaker, Elad Liebman, Reuth Mirsky, Sanmit Narvekar, Scott Niekum, Sid Desai, Yunshu Du, and the anonymous reviewers for reviewing our work and for their helpful comments. This work has taken place in the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. LARG research is supported in part by grants from the National Science Foundation (CPS-1739964, IIS-1724157, NRI-1925082), the Office of Naval Research (N00014-18-2243), Future of Life Institute (RFP2-000), Army Research Office (W911NF-19-2-0333), DARPA, Lockheed Martin, General Motors, and Bosch. The views and conclusions contained in this document are those of the authors alone. Peter Stone serves as the Executive Director of Sony AI America and receives financial compensation for this work.

The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

References

Asis, K. D., Hernandez-Garcia, J. F., Holland, G. Z., and Sutton, R. S. Multi-step reinforcement learning: A unifying algorithm, 2017.

Baird, L. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Machine Learning (ICML)*. 1995.

Bellman, R. E. *Dynamic Programming*. Dover Publications, Inc., USA, 2003. ISBN 0486428095.

Bertsekas, D. P. *Dynamic Programming: Deterministic and Stochastic Models*. Prentice-Hall, Inc., USA, 1987. ISBN 0132215810.

Bradtke, S. and Barto, A. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22: 33–57, 03 1996. doi: 10.1007/BF00114723.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *ArXiv*, abs/1606.01540, 2016.

Delyon, B. and Portier, F. Integral approximation by kernel smoothing. *Bernoulli*, 22(4):2177?2208, Nov 2016. ISSN 1350-7265. doi: 10.3150/15-bej725. URL http://dx.doi.org/10.3150/15-BEJ725.

Gerschgorin, S. Uber die abgrenzung der eigenwerte einer matrix. *Izvestija Akademii Nauk SSSR, Serija Matematika*, 7(3):749–754, 1931.

Ghiassian, S., Patterson, A., White, M., Sutton, R. S., and White, A. Online off-policy prediction, 2018.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M. (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL http://proceedings.mlr.press/v9/glorot10a.html.

Gu, S., Holly, E., Lillicrap, T. P., and Levine, S. Deep reinforcement learning for robotic manipulation. *CoRR*, abs/1610.00633, 2016. URL http://arxiv.org/abs/1610.00633.

Hanna, J. and Stone, P. Reducing sampling error in the monte carlo policy gradient estimator. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2019.

Hanna, J., Niekum, S., and Stone, P. Importance sampling policy evaluation with an estimated behavior policy. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, June 2019.

Henmi, M., Yoshida, R., and Eguchi, S. Importance sampling via the estimated sampler. *Biometrika*, 94(4):985–991, 2007. URL https://EconPapers.repec.org/RePEc:oup:biomet:v:94:y:2007:i:4:p:985-991.

Hirano, K., Imbens, G. W., and Ridder, G. Efficient estimation of average treatment effects using the estimated propensity score. *Econometrica*, 71(4):1161–1189, 2003. doi: 10.1111/1468-0262.00442. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/1468-0262.00442.

Kemeny, J. G., Snell, J. L., et al. *Finite markov chains*, volume 356. van Nostrand Princeton, NJ, 1960.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

Kober, J., Bagnell, J., and Peters, J. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32:1238–1274, 09 2013. doi: 10.1177/0278364913495721.

Konda, V. R. and Tsitsiklis, J. N. Actor-critic algorithms. In Solla, S. A., Leen, T. K., and Müller, K. (eds.), *Advances in Neural Information Processing Systems 12*, pp. 1008–1014. MIT Press, 2000. URL http://papers.nips.cc/paper/1786-actor-critic-algorithms.pdf.

Li, L., Munos, R., and Szepesvári, C. Toward minimax off-policy value estimation. In *AISTATS*, 2015.

Mahmood, A. R., Yu, H., White, M., and Sutton, R. S. Emphatic temporal-difference learning. *arXiv preprint arXiv:1507.01569*, 2015.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

Narita, Y., Yasui, S., and Yata, K. Efficient counterfactual learning from bandit feedback. *CoRR*, abs/1809.03084, 2018. URL http://arxiv.org/abs/1809.03084.

Pan, Y., White, A. M., and White, M. Accelerated gradient temporal difference learning. *CoRR*, abs/1611.09328, 2016. URL http://arxiv.org/abs/1611.09328.

Precup, D., Sutton, R. S., and Singh, S. Eligibility traces for off-policy policy evaluation. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*, pp. 759–766, 2000a.

Precup, D., Sutton, R. S., and Singh, S. P. Eligibility traces for off-policy policy evaluation. In *International Conference on Machine Learning (ICML)*, 2000b.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Puterman, M. L. and Shin, M. C. Modified policy iteration algorithms for discounted markov decision problems. *Management Science*, 24(11):1127–1137, 1978. ISSN 00251909, 15265501. URL http://www.jstor.org/stable/2630487.

Rosenbaum, P. R. Model-based direct adjustment. *Journal of the American Statistical Association*, 82(398):387–394, 1987. ISSN 01621459. URL http://www.jstor.org/stable/2289440.

Rummery, G. A. and Niranjan, M. On-line q-learning using connectionist systems. Technical report, 1994.

Sallab, A. E., Abdou, M., Perot, E., and Yogamani, S. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017(19):70–76, 2017.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

Sutton, R. S. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E. (eds.), *Advances in Neural Information Processing Systems 8*, pp. 1038–1044. MIT Press, 1996.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C., and Wiewiora, E. 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, 2009.

Thomas, P. S. A notation for markov decision processes. *CoRR*, abs/1512.09075, 2015. URL http://arxiv.org/abs/1512.09075.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

van Seijen, H., van Hasselt, H., Whiteson, S., and Wiering, M. A theoretical and empirical analysis of expected sarsa. In *2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pp. 177–184, March 2009. doi: 10.1109/ADPRL.2009.4927542.

Welch, B. L. The Generalization Of Student's Problem When Several Different Population Variances Are Involved. *Biometrika*, 34(1-2):28–35, 01 1947. ISSN 0006-3444. doi: 10.1093/biomet/34.1-2.28. URL https://doi.org/10.1093/biomet/34.1-2.28.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3?4):229?256, May 1992. ISSN 0885-6125. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Xie, Y., Liu, B., Liu, Q., Wang, Z., Zhou, Y., and Peng, J. Off-policy evaluation and learning from logged bandit feedback: Error reduction via surrogate policy. *CoRR*, abs/1808.00232, 2018. URL http://arxiv.org/abs/1808.00232.