High-confidence error estimates for learned value functions

Touqir Sajed
Department of Computing Science
University of Alberta, Canada
touqir@ualberta.ca

Welsey Chung
Department of Computing Science
University of Alberta, Canada
wchung@ualberta.ca

Martha White
Department of Computing Science
University of Alberta, Canada
whitem@ualberta.ca

Abstract

Estimating the value function for a fixed policy is a fundamental problem in reinforcement learning. Policy evaluation algorithms—to estimate value functions—continue to be developed, to improve convergence rates, improve stability and handle variability, particularly for off-policy learning. To understand the properties of these algorithms, the experimenter needs high-confidence estimates of the accuracy of the learned value functions. For environments with small, finite state-spaces, like chains, the true value function can be easily computed, to compute accuracy. For large, or continuous state-spaces, however, this is no longer feasible. In this paper, we address the largely open problem of how to obtain these high-confidence estimates, for general state-spaces. We provide a high-confidence bound on an empirical estimate of the value error to the true value error. We use this bound to design an offline sampling algorithm, which stores the required quantities to repeatedly compute value error estimates for any learned value function. We provide experiments investigating the number of samples required by this offline algorithm in simple benchmark reinforcement learning domains, and highlight that there are still many open questions to be solved for this important problem.

1 INTRODUCTION

Policy evaluation is a key step in many reinforcement learning systems. Policy evaluation approximates the value of each state—future sum of rewards—given a policy and either a model of the world or a stream of data produced by an agents choices. In classical policy iteration schemes, the agent continually alternates between improving the policy using the current approximation of the value function, and updating the approximate value function for the new policy. Policy search methods like actor-critic estimate the value function of the current policy to perform gradient updates for the policy.

However, there has been relatively little research into methods for accurately evaluating policy evaluation algorithms when the true values are not available. In most domains where we are interested in performing policy evaluation, it is difficult or impossible to compute the true value function. We may not have access to the transition probabilities or the reward function in every state, making it impossible to obtain the closed form solution of the true value function v^*. Even if we have access to a full model of the environment, we may not be able to represent the value function if the number of states is too large or the state is continuous. Aside from small finite MDPs like gridworlds and random MDPs, where closed-form solutions can be computed [Geist and Scherrer, 2014, White and White, 2016], we often do not have access to v^*. In nearly all our well-known benchmark domains, such as Mountain Car, Puddle World, Cart pole, and Acrobot, we must turn to some other method to evaluate learning progress.

One option that has been considered is to estimate the objective minimized by the algorithms. Several papers [Sutton et al., 2008, Du et al., 2017] have compared the performance of the algorithms in terms of their target objective on a batch of samples, using the approximate linear system for the mean-squared projected Bellman error (MSPBE). One estimator, called RUPEE [White, 2015], is designed to incrementally approximate the MSPBE by keeping a running average across data produced during learning. Some terms, such as the feature covariance matrix, can be estimated easily; however, one component of the MSPBE includes the current weights, and is biased by this moving average approach. More problematically, some algorithms do not converge to the minimum of the MSPBE, such as residual gradient for the mean-squared Bellman error [Baird, 1995] or Emphatic Temporal Difference (ETD) learning [Sutton et al., 2016], which minimize a variant of the MSPBE with a different weighting. This approach, therefore, is limited to comparing algorithms that minimize the same objective.

The more standard approach has been to use rollouts from states to obtain samples of returns. To obtain these rollout estimates, three parameters need to be chosen: the number of states m from which to rollout, the number of rollouts or trajectories t, and the length of each rollout. Given these rollouts, the true values can be estimated from each of the m chosen states, stored offline, and then used for comparison repeatedly during experiments. These evaluation schemes, however, have intuitively chosen parameters, without any guarantees that the distance to the true values, the error, is well-estimated. Early work comparing gradient TD algorithms [Maei et al., 2009] used sampled trajectories—2500 of them—but compared to returns, rather than value estimates. For several empirical studies using benchmark domains, like Mountain Car and Acrobot, there are a variety of choices, including $t = m = 500$ [Gehring et al., 2016]; $m = 2000$, $t = 300$ and 1000 length rollouts [Pan et al., 2017]; and $m = 5000$, $t = 5000$ [Le et al., 2017]. For a continuous physical system, [Dann et al., 2014] used as little as 10 rollouts from a state. Otherwise, other papers have mentioned that extensive rollouts are used[1], but did not describe how [Konidaris et al., 2011, Dabney and Thomas, 2014]. In general, as new policy evaluation algorithms are derived, it is essential to find a solution to this open problem: How can we confidently compare value estimates returned by our algorithms?

In this work, we provide an algorithm that ensures, with high-probability, that the estimated distance has small error in approximating the true distance between the true value function v^* for an arbitrary estimate $\hat{v}$. We focus in the main body of the paper on the clipped mean-absolute percentage value error (CMAPVE) as a representative example of the general strategy. We provide additional results for a variety of other losses in the appendix, to facilitate use for a broader range of error choices. We conclude by demonstrating the rollout parameters chosen for several case studies, highlighting that previous intuitive choices did not effectively direct sampling. We hope for this algorithm to become a standard approach for generating estimates of the true values to facilitate comparison of policy evaluation algorithms by reinforcement learning researchers.

[1]Note that [Boyan and Moore, 1995] used rollouts for a complementary purpose, to train a nonlinear value function, rather than for evaluating policy evaluation algorithms.

2 MEASURES OF LEARNING PERFORMANCE

This paper investigates the problem of comparing algorithms that estimate the discounted sum of future rewards *incrementally* for a fixed policy. In this section, we first introduce the policy evaluation problem and motivate a particular measures of learning performance for policy evaluation algorithms. In the following section, we discuss how to estimate this measure.

We model the agent's interaction with the world as a Markov decision process (MDP), defined by a (potentially uncountable) set of states $\mathcal{S}$, a finite set of actions $\mathcal{A}$, transitions $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, \infty)$, rewards $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ and a scalar discount function $\gamma : \mathcal{S} \to \mathbb{R}$. On each time step t, the agent selects an action according to it's *behaviour policy* $A_t \sim \mu(\cdot|S_t)$, the environment transitions into a new state $S_{t+1} \sim P(\cdot|S_t, A_t)$ and the agent receives a scalar reward $R_{t+1} \overset{\text{def}}{=} R(S_t, A_t, S_{t+1})$. In policy evaluation, the agent's objective is to estimate the expected return

$$v^*(s) = \mathbb{E}[G_t|S_t = s, A_t \sim \pi] \tag{1}$$
$$\text{for return } G_t = \sum_{i=0}^{\infty} \gamma^i R_{t+i+1}$$

where $v^*(s)$ is called the *state-value function* for the *target policy* $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$. From a stream of data, the agent incrementally approximates this value function, $\hat{v}$. For experiments, to report learning curves, we need to measure the accuracy of this estimate every step or at least periodically, such as every 10 steps.

For policy evaluation, when the policy remains fixed, the value error remains the gold standard of evaluation. Ignoring how useful the value function is for policy improvement, our only goal is accuracy with respect to v^*. Assume some weighting $d : \mathcal{S} \to [0, \infty)$, a probability distribution over states. Given access to v^*, it is common to estimate the mean squared value error

$$\begin{aligned} \text{MSVE}(\hat{v}) &\overset{\text{def}}{=} \mathbb{E}[(\hat{v}(S) - v^*(S))^2] \qquad (2) \\ &= \int_{\mathcal{S}} d(s) \left(\hat{v}(S) - v^*(s)\right)^2 ds \end{aligned}$$

or the mean absolute value error

$$\text{MAVE}(\hat{v}) \overset{\text{def}}{=} \mathbb{E}[|\hat{v}(s) - v^*(s)|]. \tag{3}$$

The integral is replaced with a sum if the set of states is finite. Because we consider how to estimate this error for continuous state domains—for which it is more difficult to directly estimate v^*—we preferentially assume the states are continuous.

These losses, however, have several issues, beyond estimating them. The key issue is that the scale of the returns can be quite different across states. This skews the loss and, as we will see, makes it more difficult to get high-accuracy estimates. Consider a cost-to-goal problem, where the agent receives a reward of -1 per step. From one state the value could be -1000 and for another it could be -1. For a prediction of -990 and -11 respectively, the absolute value error for both states would be -10. However, the prediction of -990 for the first state is quite accurate, whereas a prediction of -11 for the second state is highly inaccurate.

One alternative is to estimate a percentage error, or relative error. The mean absolute percentage value error is

$$\text{MAPVE}(\hat{v}) \overset{\text{def}}{=} \mathbb{E}\left[\frac{|\hat{v}(S) - v^*(S)|}{|v^*(S)| + \tau}\right] \tag{4}$$

for some $\tau \geq 0$. The term τ in the denominator ensures the MAPVE does not become excessively high, if true values of states are zero or near zero. For example, for $\tau = 1$, the MAPVE is essentially the MAVE for small $v^*(s)$, which reflects that small absolute differences are meaningful for these smaller numbers. For large $v^*(s)$, the τ has little effect, and the MAPVE becomes a true percentage error, reflecting the fact that we are particularly interested in relative errors for larger $v^*(s)$.

Additionally, the MAPVE can be quite large if $\hat{v}$ is highly inaccurate. When estimating these performance measures, however, it is uninteresting to focus on obtaining high-accuracy estimate of very large MAPVE. Rather, it is sufficient to report that $\hat{v}$ is highly inaccurate, and focus the estimation of the loss on more accurate $\hat{v}$. Towards this goal, we introduce the clipped MAPVE

$$\text{CMAPVE}(\hat{v}) \overset{\text{def}}{=} \mathbb{E}\left[\min\left(c, \frac{|\hat{v}(S) - v^*(S)|}{|v^*(S)| + \tau}\right)\right]$$

for some $c > 0$. This c provides a maximum percentage error. For example, setting $c = 2$ caps error estimates for approximate values that are worse than 200% inaccurate. Such a level of inaccuracy is already high, and when comparing policy evaluation algorithms, we are much more interested in their percentage error—particularly compared to each other—once we are within a reasonable range around the true values. Note that c can be chosen to be the maximum value of the loss, and so the following results remain quite general.

Though many losses could be considered, we put forward the CMAPVE as a proposed standard for policy evaluation. The parameters τ and c can both be appropriately chosen by the experimentalist, for a given MDP. These parameters give sufficient flexibility in highlighting differences between algorithms, while still enabling high-confidence estimates of these errors, which we discuss next. For this reason, we use CMAPVE as the loss in the main body of the text. However, for completeness, we also show how to modify the analysis and algorithms for other losses in the appendix.

3 HIGH-CONFIDENCE ESTIMATES OF VALUE ERROR

Our goal now is to approximate the value error, CMAPVE, with high-confidence, for any value function $\hat{v}$. Instead of approximating the error directly for each $\hat{v}$, the typical approach is to estimate v^* as accurately as possible, for a large set of states $s_1, \ldots, s_m \sim d$. Given these high-accuracy estimates $\bar{v}$, the true expected error can be approximated from this subset of states for any $\hat{v}$.

$$\text{CMAPVE}(\hat{v}) \approx \frac{1}{m}\sum_{i=1}^{m} \min\left(c, \frac{|\hat{v}(s_i) - \bar{v}(s_i)|}{|\bar{v}(s_i)| + \tau}\right)$$

Since the CMAPVE needs to be computed frequently, for many steps during learning potentially across many algorithms, it is important for this estimate of CMAPVE to be efficiently computable. An important requirement, then, is for the number of states m to be as small as possible, so that all the $\bar{v}(s_i)$ can be stored and the summed difference is quick to compute.

One possible approach is to estimate the true value function v^* using a powerful function approximator, offline. A large batch of data could be gathered, and a learning method used to train $\bar{v}$. This large function approximator would not even need to be stored: only $\bar{v}(s_i)$ would need to be saved once this offline procedure was complete. This approach, however, will be biased by the form of the function approximator, which can favor certain policy evaluation algorithms during evaluation. Further, it is difficult to quantify this bias, particularly in a general way agnostic to the type of function approximator an experimentalist might use for their learning setting.

An alternative strategy is to use many sampled rollouts from this subset of states. This strategy is general—requiring only access to samples from the MDP. A much larger number of interactions can be used with the MDP, to compute $\bar{v}$, because this is computed once, offline, to facilitate many further empirical comparisons between algorithms after. For example, one may want to examine the early learning performance of two different policy evaluation algorithms—which may themselves receive only a small number of samples. The cached $\bar{v}$ then enables computing this early learning performance. However, even offline, there are limits to how many samples can be computed feasibly, particularly for computationally costly simulators [Dann et al., 2014].

Therefore, our goal is the following: how can we *efficiently* compute *high-confidence* estimates of CMAPVE, using a *minimal number of offline rollouts*. The choice of a clipped loss actually enables the number of states m to remain small (shown in Lemma 2), enabling efficient computation of CMAPVE. In the next section, we address the second point: how to obtain high-confidence estimates, given access to $\bar{v}$ that approximates v^*. In the following section, we discuss how to obtain these $\bar{v}$.

3.1 OVERVIEW

We first provide an overview of the approach, to make it easier to follow the argument. We additionally include a notation table (Table 1), particularly to help discern the various value functions.

Table 1: Table of Notation

v^*	true values for policy π
$\bar{v}^*$	true values for policy π, when using truncated rollouts to length l
$\bar{v}$	estimated values for policy π using t rollouts, when using truncated rollouts to length l
$\hat{v}$	estimated values for policy π, being evaluated
d	distribution over the states $\mathcal{S}$, $d : \mathcal{S} \to [0, \infty)$
m	number of states $\{s_1, \ldots, s_m\}$, $s_i \sim d$
ℓ_c	clipped error, $\ell_c(v_1, v_2) = \min\left(c, \frac{\lvert v_1 - v_2 \rvert}{\lvert v_2 \rvert + \tau}\right)$
ℓ	true error, $\ell(\hat{v}, v^*) = \mathbb{E}[\ell_c(\hat{v}(S), v^*(S))]$ under d
$\hat{\ell}$	approximate error, $\hat{\ell}(\hat{v}, v^*) = \frac{1}{m}\sum_{i=1}^{m} \ell_c(\hat{v}(s_i), v^*(s_i))$
R_{max}	an upper bound on the maximum absolute value reward, $R_{\text{max}} \geq \sup \lvert R(s, a, s') \rvert$
V_{max}	maximum absolute value for the policy π for any state, e.g., $V_{\text{max}} = \frac{\text{max reward} - \text{min reward}}{1-\gamma}$
K	the number of times the error estimate is queried

First, we consider several value function approximations, for use within the bound, summarized in Table 1. The goal is to determine the accuracy of the estimates of the learned $\hat{v}$ with respect to the true values v^*. We estimate true values $v^*(s_i)$ for s_i using repeated rollouts from s_i; this results in two forms of error. The first is due to truncated rollouts, which for the continuing case would otherwise be infinitely long. The second source of error is due to using an empirical estimate of the true values, by averaging sampled returns. We denote $\bar{v}^*$ as the true values, for truncated returns, and $\bar{v}$ as the sample estimate of $\bar{v}^*$ from m truncated rollouts.

Second, we consider the approximation in computing the loss: the difference between $\hat{v}$ and v^*. We consider the true loss $\ell(\hat{v}, v^*)$ and the approximate loss $\hat{\ell}(\hat{v}, v^*)$, in Table 1. The argument in Theorem 1 revolves around upper bounding the difference between these two losses, in terms of three terms. These terms are bounded in Lemmas 2, 3 and 4. Lemma 2 bounds the error due to sampling only a subset of m. Lemma 3 bounds the error from approximating v^* with truncated rollouts. Lemma 4 bounds the error from dividing by $|\bar{v}(s_i)| + \tau$ instead of $|v^*(s_i)| + \tau$.

Finally, to obtain this general bound, we first assume that we can obtain highly-accurate estimates $\bar{v}$ of $\bar{v}^*$. We state these two assumptions in Assumptions 1 and 2. These estimates could be obtained with a variety of sampling strategies, and so separate it from the main proof. We later develop one algorithm to obtain such estimates, in Section 4.

3.2 MAIN RESULT

We will compute rollout values from a set of sampled states $\{s_i\}_{i=1}^m$. Each rollout consists of a trajectory simulated, or rolled out, some number of steps. The length of this trajectory can itself be random, depending on if an episode terminates or if the trajectory is estimated to be a sufficiently accurate sample of the full, non-truncated return. We first assume that we have access to such trajectories and rollout estimates and in later sections show how to obtain such trajectories and estimates.

Assumption 1. *For any $\epsilon > 0$ and sampled state s_i, the trajectory lengths l_i are specified such that,*

$$|\bar{v}^*(s_i) - v^*(s_i)| \leq \epsilon\left(|v^*(s_i)| + \tau\right) \tag{5}$$

Assumption 2. *Starting from s_i, assume you have trajectories of rewards $\{r_{ijk}\}$ for trajectory index $j \in \{1, \ldots, t\}$ and rollout index $k \in \{0, \ldots, l_{ij} - 1\}$ for a trajectory length l_{ij} that depends on the trajectory. The approximated rollout values*

$$\bar{v}(s_i) \overset{\text{def}}{=} \frac{1}{t}\sum_{j=1}^{t}\sum_{k=0}^{l_{ij}-1} \gamma^k r_{ijk} \tag{6}$$

are an (ϵ, δ, τ) -approximation to the true expected values, where l_{ij} is an instance of the random variable l_i

$$\bar{v}^*(s) \overset{\text{def}}{=} \mathbb{E}\left[\sum_{k=0}^{l_i - 1} \gamma^k R_k\right] \tag{7}$$

i.e, for $0 < \epsilon$, with probability at least $1 - \delta/2$, the following holds for all states

$$|\bar{v}^*(s_i) - \bar{v}(s_i)| \leq \epsilon\left(|\bar{v}^*(s_i)| + \tau\right) \tag{8}$$

Theorem 1. *Let $\{s_1, \ldots, s_m\}$ be states sampled according to d. Assume $\bar{v}(s_i)$ satisfies Assumption 1 and 2 for* *all $i \in \{1, \dots, m\}$. Suppose the empirical loss mean estimates are computed K number of times with different learned value functions $\hat{v}$ each time. Then the approximation error*

$$\hat{\ell}(\hat{v}, \bar{v}) \overset{\text{def}}{=} \frac{1}{m}\sum_{i=1}^{m} \ell_c(\hat{v}(s_i), \bar{v}(s_i)) \tag{9}$$

for all the $\hat{v}$ satisfies, with probability at least $1 - \delta$,

$$\Big|\ell(\hat{v}, v^*) - \hat{\ell}(\hat{v}, \bar{v})\Big| \le (11) + (12) + (13) \tag{10}$$

Proof: We need to bound the errors introduced from having a reduced number of states, a finite set of trajectories to approximate the expected returns for each of those states and truncated rollouts to get estimates of returns. To do so, we first consider the difference under the approximate clipped loss, to the true value function.

$$\Big|\ell(\hat{v}, v^*) - \hat{\ell}(\hat{v}, \bar{v})\Big| \le \Big|\ell(\hat{v}, v^*) - \hat{\ell}(\hat{v}, v^*)\Big| + \Big|\hat{\ell}(\hat{v}, v^*) - \hat{\ell}(\hat{v}, \bar{v})\Big|$$

The first component is bounded in Lemma 2. For the second component, notice that

$$\Big|\hat{\ell}(\hat{v}, v^*) - \hat{\ell}(\hat{v}, \bar{v})\Big| \le \frac{1}{m}\sum_{i=1}^{m} |\ell_c(\hat{v}(s_i), v^*(s_i)) - \ell_c(\hat{v}(s_i), \bar{v}(s_i))|$$

However, these two differences are difficult to compare, because they have different denomiators: the first has $|v^*(s_i)| + \tau$, whereas the second has $|\bar{v}(s_i)| + \tau$. We therefore further separate each component in the sum

$$\begin{aligned}
&\Big|\ell_c(\hat{v}(s_i), v^*(s_i)) - \ell_c(\hat{v}(s_i), \bar{v}(s_i))\Big| \\
&\le \Big|\ell_c(\hat{v}(s_i), v^*(s_i)) - \min\left(c, \frac{|\hat{v}(s_i) - \bar{v}(s_i)|}{|v^*(s_i)| + \tau}\right)\Big| \\
&+ \Big|\min\left(c, \frac{|\hat{v}(s_i) - \bar{v}(s_i)|}{|v^*(s_i)| + \tau}\right) - \ell_c(\hat{v}(s_i), \bar{v}(s_i))\Big|
\end{aligned}$$

The first difference has the same denominator, so

$$\begin{aligned}
&\Big|\ell_c(\hat{v}(s_i), v^*(s_i)) - \min\left(c, \frac{|\hat{v}(s_i) - \bar{v}(s_i)|}{|v^*(s_i)| + \tau}\right)\Big| \\
&= \Big|\min\left(c, \frac{|\hat{v}(s_i) - v^*(s_i)|}{|v^*(s_i)| + \tau}\right) - \min\left(c, \frac{|\hat{v}(s_i) - \bar{v}(s_i)|}{|v^*(s_i)| + \tau}\right)\Big| \\
&= \frac{1}{|v^*(s_i)| + \tau}\Big|\min\left(c(|v^*(s_i)| + \tau), |\hat{v}(s_i) - v^*(s_i)|\right) \\
&- \min\left(c(|v^*(s_i)| + \tau), |\hat{v}(s_i) - \bar{v}(s_i)|\right)\Big| \\
&\le \frac{1}{|v^*(s_i)| + \tau}\min\left(c(|v^*(s_i)| + \tau), |v^*(s_i) - \bar{v}(s_i)|\right) \\
&= \ell_c(\bar{v}(s_i), v^*(s_i))
\end{aligned}$$

The last step follows from the triangle inequality $|\,|x| - |y|\,|_c \le |x - y|_c$ on the clipped loss (see Lemma 7, in Appendix A, for an explicit proof that the triangle inequality holds for the clipped loss).

Therefore, putting it all together, we have

$$\begin{aligned}
&\Big|\ell(\hat{v}, v^*) - \hat{\ell}(\hat{v}, \bar{v})\Big| \\
&\le \Big|\ell(\hat{v}, v^*) - \hat{\ell}(\hat{v}, v^*)\Big| \\
&+ \frac{1}{m}\sum_{i=1}^{m} \ell_c(\bar{v}(s_i), v^*(s_i)) \\
&+ \frac{1}{m}\sum_{i=1}^{m} \Big|\min\left(c, \frac{|\hat{v}(s_i) - \bar{v}(s_i)|}{|v^*(s_i)| + \tau}\right) - \ell_c(\hat{v}(s_i), \bar{v}(s_i))\Big|
\end{aligned}$$

where the first, second and third components are bounded in Lemmas 2, 3 and 4 respectively. Finally, due to the application of Hoeffding's bound (Lemma 2) with error probability of atmost $\delta/2$ and assumption 2 which may not hold with probability atmost $\delta/2$ and the union bound, we conclude that the final bound holds with probability at least $1 - \delta$. ■

Lemma 2 (Dependence on m)**.** *Suppose the empirical loss mean estimates are computed K number of times. Then with probability at least $1 - \delta/2$:*

$$\Big|\ell(\hat{v}, v^*) - \hat{\ell}(\hat{v}, v^*)\Big| \le \sqrt{\frac{\log(4K/\delta)c^2}{2m}} \tag{11}$$

Proof: Since $\hat{\ell}(\hat{v}, v^*) = \frac{1}{m}\sum_{i=1}^{m} \ell_c(\hat{v}(s_i), v^*(s_i))$ is an unbiased estimate of $\ell(\hat{v}, v^*)$, we can use Hoeffding's bound for variables bounded between $[0, c]$. For any of the K times, the concentration probability is as follows:

$$\begin{aligned}
\Pr\left(\Big|\ell(\hat{v}, v^*) - \hat{\ell}(\hat{v}, v^*)\Big| \ge t\right) &\le 2\exp\left(\frac{-2t^2m^2}{\sum_{i=1}^{m} c^2}\right) \\
&= 2\exp\left(\frac{-2mt^2}{c^2}\right) = \frac{\delta}{2K}
\end{aligned}$$

Thus, due to union bound over all the K times, for all those empirical loss mean estimates, the following holds

$$Pr\left(\Big|\ell(\hat{v}, v^*) - \hat{\ell}(\hat{v}, v^*)\Big| \le t\right) \ge 1 - \delta/2.$$

Rearranging the above, to express t in terms of δ,

$$2\exp\left(\frac{-2mt^2}{c^2}\right) = \frac{\delta}{2K} \quad \Longrightarrow \quad t = \sqrt{\frac{\log(4K/\delta)c^2}{2m}}.$$

Therefore, with probability at least $1 - \delta$,

$$\Big|\ell(\hat{v}, v^*) - \hat{\ell}(\hat{v}, v^*)\Big| \le \sqrt{\frac{\log(4K/\delta)c^2}{2m}}.$$

■

Lemma 3 (Dependence on Truncated Rollout Errors)**.** *Under Assumption 2 and 1,*

$$\frac{1}{m}\sum_{i=1}^{m} \ell_c(\bar{v}(s_i), v^*(s_i)) \le 2\epsilon \tag{12}$$

Proof: We can split up this error into sampling error for a finite length rollout and for a finite number of trajectories. We can consider the unclipped error, which is an upper bound on the clipped error.

$$\frac{|v^*(s_i) - \bar{v}(s_i)|}{|v^*(s_i)| + \tau} \leq \frac{|v^*(s_i) - \bar{v}^*(s_i)|}{|v^*(s_i)| + \tau} + \frac{|\bar{v}^*(s_i) - \bar{v}(s_i)|}{|v^*(s_i)| + \tau}$$

These two terms are both bounded by ϵ, by assumption. ■

Lemma 4. *Under Assumption 2,*

$$\left| \min\left(c, \frac{|\hat{v}(s_i) - \bar{v}(s_i)|}{|v^*(s_i)| + \tau}\right) - \min\left(c, \frac{|\hat{v}(s_i) - \bar{v}(s_i)|}{|\bar{v}(s_i)| + \tau}\right) \right|$$
$$\leq c(1 - (1+\epsilon)^{-2}) \tag{13}$$

Proof: We need to bound the difference due to the difference in normalizer. To do so, we simply need to find a constant $\beta > 0$ such that $\min\left(c, \frac{|\hat{v}(s_i) - \bar{v}(s_i)|}{|v^*(s_i)| + \tau}\right) \geq \beta$

The key is to lower bound $|v^*(s_i)| + \tau$, which results in an upper bound on the first term and consequently an upper bound on the difference between the two terms.

$$\begin{aligned}
\frac{|\bar{v}(s_i)| + \tau}{|v^*(s_i)| + \tau} &\leq \frac{|\bar{v}(s_i) - \bar{v}^*(s_i)| + |\bar{v}^*(s_i)| + \tau}{|v^*(s_i)| + \tau} \\
&= \frac{|\bar{v}(s_i) - \bar{v}^*(s_i)|}{|v^*(s_i)| + \tau} + \frac{|\bar{v}^*(s_i)| + \tau}{|v^*(s_i)| + \tau} \\
&\leq \frac{\epsilon(|\bar{v}^*(s_i)| + \tau)}{|v^*(s_i)| + \tau} + \frac{|\bar{v}^*(s_i)| + \tau}{|v^*(s_i)| + \tau} \\
&= \frac{(1+\epsilon)(|\bar{v}^*(s_i)| + \tau)}{|v^*(s_i)| + \tau}
\end{aligned}$$

where the second inequality is due to Assumption 2. Now further

$$\begin{aligned}
\frac{(|\bar{v}^*(s_i)| + \tau)}{|v^*(s_i)| + \tau} &\leq \frac{|\bar{v}^*(s_i) - v^*(s_i)|}{|v^*(s_i)| + \tau} + \frac{|v^*(s_i)| + \tau}{|v^*(s_i)| + \tau} \\
&\leq \frac{\epsilon(|v^*(s_i)| + \tau)}{|v^*(s_i)| + \tau} + 1 \\
&= \epsilon + 1
\end{aligned}$$

giving

$$\frac{|\bar{v}(s_i)| + \tau}{|v^*(s_i)| + \tau} \leq (1+\epsilon)^2 \implies |v^*(s_i)| + \tau \geq \frac{|\bar{v}(s_i)| + \tau}{(1+\epsilon)^2}.$$

So, for $a = (1+\epsilon)^2$ and $b = \frac{|\hat{v}(s_i) - \bar{v}(s_i)|}{|\bar{v}(s_i)| + \tau}$, the term $|\min(c, ab) - \min(c, b)|$ upper bounds the difference. Because $a > 1$ and $b > 0$, this term $|\min(c, ab) - \min(c, b)|$ is maximized when $ab = c$, and $b = c/a$. In the worst case, therefore, $|\min(c, ab) - \min(c, b)| \leq c(1 - a^{-1})$ which finishes the proof. ■

3.3 SATISFYING THE ASSUMPTIONS

The bounds above relied heavily on accurate sample estimates of $v^*(s_i)$. To obtain Assumption 1, we need to rollout trajectories sufficiently far to ensure that truncated sampled returns do not incur too much bias. For problems with discounting, for $\gamma < 1$, the returns can be truncated once γ^l becomes sufficiently small, as the remaining terms in the sum for the return have negligible weight. For episodic problems with no discounting, it is likely that trajectories need to be simulated until termination, since rewards beyond the truncation horizon would not be discounted and so could have considerable weight.

We show how to satisfy Assumption 1, for the discounted setting. Note that for the trivial setting of $R_{\text{max}} = 0$, it is sufficient to use $l = 1$, so we assume $R_{\text{max}} > 0$.

Lemma 5. *For $\gamma < 1$ and $R_{max} > 0$, if*

$$l = \left\lceil \frac{\log(\epsilon\tau(1-\gamma)) - \log(R_{max})}{\log \gamma} \right\rceil \tag{14}$$

then $\bar{v}^*(s)$ satisfies Assumption 1:

$$|\bar{v}^*(s) - v^*(s)| \leq \epsilon\,(|v^*(s)| + \tau) \tag{15}$$

Proof: The first component can be bounded as

$$\begin{aligned}
|\bar{v}^*(s) - v^*(s)| &\leq \left| E\left[\sum_{k=0}^{\infty} \gamma^k R_k\right] - E\left[\sum_{k=0}^{l-1} \gamma^k R_k\right] \right| \\
&\leq R_{\text{max}} \left(\frac{1}{1-\gamma} - \frac{1-\gamma^l}{1-\gamma}\right) \\
&= R_{\text{max}} \frac{\gamma^l}{1-\gamma}
\end{aligned}$$

giving

$$\frac{|\bar{v}^*(s) - v^*(s_i)|}{|v^*(s)| + \tau} \leq R_{\text{max}} \frac{\gamma^l}{\tau(1-\gamma)}.$$

Setting l as in (14) ensures $R_{\text{max}} \frac{\gamma^l}{\tau(1-\gamma)} \leq \epsilon$, completing the proof. ■

For Assumption 2, we need a stopping rule for sampling truncated returns that ensures $\bar{v}$ is within ϵ of the true expected value of the truncated returns, $\bar{v}^*$. The idea is to continue sampling truncated returns, until the confidence interval around the mean estimate shrinks sufficiently to ensure, with high probability, that the values estimates are within ϵ of the true values. Such stopping rules have been designed for non-negative random variables [Domingos and Hulten, 2001, Dagum et al., 2006], and extended to more general random variables [Mnih et al., 2008]. We defer the development of such an algorithm for this setting until the next section.

4 THE ROLLOUT ALGORITHM

We can now design a high-confidence algorithm for estimating the accuracy of a value function. Practically, the most important number to reduce is m, because these values will be stored and used for comparisons on each step. The choice of a clipped loss, however, makes it more manageable to control m. In this section, we focus more on how much the variability in trajectories, and trajectory length, impact the number of required samples.

The general algorithm framework is given in Algorithm 1. The algorithm is straightforward once given an algorithm to sample rollouts from a given state. The rollout algorithm is where development can be directed, to reduce the required number of samples. This rollout algorithm needs to be designed to satisfy Assumptions 1 and 2. We have already shown how to select trajectory lengths to satisfy Assumption 1. Below, we describe how to select m and how to satisfy Assumption 2.

Algorithm 1 Offline computation of $\bar{v}$, to get high-confidence estimates of value error

```
1: ▷ Input $\epsilon, \delta, \tau$
2: ▷ Compute the values $\bar{v}$ once offline and store for repeated use
3: Set $\epsilon_m = \epsilon/2$ and $\bar{\epsilon} = \epsilon/(2(1+c))$
4: $m \leftarrow \frac{\log(4K/\delta)c^2}{2\epsilon_m^2}$
5: for $1, \ldots, m$ do
6:     Sample $s_i \sim d$
7:     $\bar{v}(s_i) \leftarrow$ Algorithm 2 with $\bar{\epsilon}$, $\frac{\delta}{2m}$, $\tau$
```

Specifying the number of sampled states m. For the number of required samples for the outer loop in Algorithm 1, we need enough samples to match the bound in Lemma 2.

$$\epsilon_m = \sqrt{\frac{\log(4K/\delta)c^2}{2m}} \implies m = \frac{\log(4K/\delta)c^2}{2\epsilon_m^2} \qquad (16)$$

m is chosen as $\left\lceil \frac{\log(4K/\delta)c^2}{2\epsilon_m^2} \right\rceil \geq \epsilon_m$ and thus we are being slightly conservative regarding the error to ensure correctness with high probability. We opt for a separate choice of ϵ_m for this part of the bound, because it is completely separate from the other errors. This number ϵ_m could be chosen slightly larger, to reduce the number of required sampled states to compare to, whereas ϵ might need to be smaller depending on the choice of c and τ. Separating them explicitly can significantly reduce the m in the outer loop, both improving time and storage, as well as later comparison time, without impacting the accuracy of the algorithm.

Algorithm 2 High-confidence Monte carlo estimate of the expected return for a state

```
1: ▷ Input $\epsilon, \delta, \tau$, state $s$
2: ▷ Output an $\epsilon, \delta, \tau$-accurate approx. $\bar{v}(s_i)$ of $v^*(s_i)$
3: $\text{LB} \leftarrow 0, \text{UB} \leftarrow \infty$
4: $\widehat{\text{LB}} \leftarrow -\infty, \widehat{\text{UB}} \leftarrow \infty$
5: $\bar{g} \leftarrow 0, M \leftarrow 0$
6: $j \leftarrow 1, h \leftarrow 0, \beta \leftarrow 1.1, p \leftarrow 1.1, \alpha \leftarrow 1, x \leftarrow 1$
7: while $(1+\epsilon)\text{LB} + 2\epsilon\tau < (1-\epsilon)\text{UB}$ or $\text{LB} = 0$ do
8:     $g \leftarrow$ sample return that satisfies Assumption 1 (e.g., see Algorithm 3 in Appendix D)
9:     $\Delta \leftarrow g - \bar{g}$
10:    $\bar{g} \leftarrow \bar{g} + \frac{\Delta}{j}$
11:    $M \leftarrow M + \Delta(g - \bar{g})$
12:    $\sigma \leftarrow \sqrt{M/j}$
13:    ▷ Compute the confidence interval
14:    if $j \geq \lfloor \beta^h \rfloor$ then
15:        $h \leftarrow h + 1$
16:        $\alpha \leftarrow \lfloor \beta^h \rfloor / \lfloor \beta^{h-1} \rfloor$
17:        $x \leftarrow -\alpha \log \frac{\delta(p-1)}{3ph^p}$
18:    $c_j \leftarrow \sigma\sqrt{\frac{2x}{j}} + \frac{3V_{\max}x}{j}$
19:    $\text{LB} \leftarrow \max(\text{LB}, |\bar{g}| - c_j)$
20:    $\text{UB} \leftarrow \min(\text{UB}, |\bar{g}| + c_j)$
21:    $\widehat{\text{LB}} \leftarrow \max(\widehat{\text{LB}}, \bar{g} - c_j)$
22:    $\widehat{\text{UB}} \leftarrow \min(\widehat{\text{UB}}, \bar{g} + c_j)$
23:    if $\frac{\widehat{\text{UB}} - \widehat{\text{LB}}}{2} \leq \epsilon\tau$ then return $\frac{\widehat{\text{UB}} + \widehat{\text{LB}}}{2}$
24:    $j = j + 1$
   return $\frac{\text{sign}(\bar{g})}{2}((1+\epsilon)\text{LB} + (1-\epsilon)\text{UB})$
```

Satisfying Assumption 2. Our goal is to get an (ϵ, δ, τ)-approximation of $\bar{v}(s_i)$, with a feasible number of samples. In many cases, it is difficult to make parametric assumptions about returns in reinforcement learning. A simple strategy is to use a stopping rule for generating returns, based on general concentration inequalities—like Hoeffding's bound—that make few assumptions about the random variables. If we had a bit more information, however, such as the variance of the returns, we could obtain a tighter bound, using Bernstein's inequality and so reduce the number of required samples. We cannot know this variance a priori, but fortunately an empirical Bernstein bound has been developed [Mnih et al., 2008]. Using this bound, Mnih et al. [2008] designed EBGStop, which incrementally estimates variance and significantly reduces the number of samples required to get high-confidence estimates.

EBGStop can be used, without modification, given a mechanism to sample truncated returns that satisfy Assumption 1. However, we generalize the algorithm to allow for our less restrictive condition $|\bar{v}^*(s_i) - \bar{v}(s_i)| \leq \epsilon(|\bar{v}^*(s_i)| + \tau)$, as opposed to the original algorithm which ensured $|\bar{v}^*(s_i) - \bar{v}(s_i)| \leq \epsilon|\bar{v}^*(s_i)|$. When $\tau = 0$ in our algorithm, it reduces to the original; since this is a generalization on that algorithm, we continue to call it EBGStop. This modification is important when $v^*(s_i) = 0$, since this would require $\bar{v}^*(s_i) = v^*(s_i)$ when $\tau = 0$. For $\tau > 0$, once the accuracy is within τ, the algorithm can stop. The Algorithm is summarized in Algorithm 2. The proof follows closely to the proof for EBGStop; we include it in Appendix A. Algorithm 2 uses geometric sampling, like EBGStop, to improve sample efficiency. The idea is to avoid checking the stopping condition after every sample. Instead, for some $\beta > 1$, the condition is checked after β^k samples; the next check occurs at β^{k+1}. This modification improves sample efficiency from a multiplicative factor of $\log(\frac{R}{\epsilon|\mu|})$ to $\log\log(\frac{R}{\epsilon|\mu|})$, where R is the range of the random variables and μ is the mean.

Lemma 6. *Algorithm 2 returns an ϵ, δ, τ-approximation $\bar{v}(s_i)$:*

$$|\bar{v}^*(s_i) - \bar{v}(s_i)| \leq \epsilon(|\bar{v}^*(s_i)| + \tau)$$

Corollary 1. *For any $0 < \epsilon_m$ and $0 < \bar{\epsilon} < 1$, Algorithm 1 returns an ϵ, δ-accurate approximation: with probability at least $1 - \delta$,*

$$\left| \ell(\hat{v}, v^*) - \frac{1}{m}\sum_{i=1}^{m} \ell_c(\hat{v}(s_i), \bar{v}(s_i)) \right| \leq \epsilon$$

where

$$\epsilon = \epsilon_m + 2(1+c)\bar{\epsilon} \tag{17}$$

Algorithm 1 uses this theorem, for a given desired level of accuracy ϵ. To obtain this level of accuracy, $\epsilon_m = \epsilon/2$ and $\bar{\epsilon}$ given to Algorithm 2 is set to ensure $2(1+c)\bar{\epsilon} = \epsilon$.

5 EXPERIMENTS ON BENCHMARK PROBLEMS

We investigate the required number of samples to get with a level of accuracy, for different probability levels. We report this for two continuous-state benchmark problems—Mountain Car and Puddle World—which have previously been used to compare policy evaluation algorithms. Our goal is to (a) demonstrate how this framework can be used to obtain high-confidence estimates of accuracy and (b) provide some insight into how many samples are needed, even for simple reinforcement learning benchmark domains.

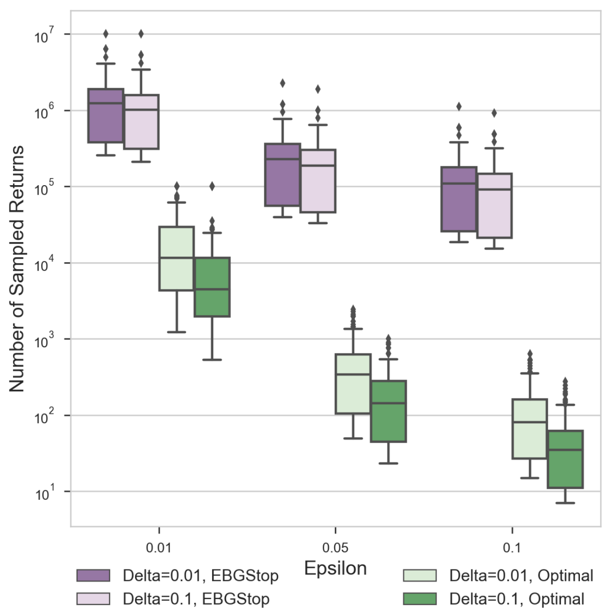
We report the number of returns sampled by Algorithm 2, averaged across several states. The domains, Mountain Car and Puddle World, are as specified in the policy evaluation experiments by Pan et al. [2017]. For Mountain Car, we use the energy pumping policy, with 60% random action selection for the three actions. For Puddle World, we used a uniform random policy for the four actions. They are both episodic tasks, with a maximum absolute value of $V_{\max} = 100$ and $V_{\max} = 8000$ respectively. The variance in Puddle World is particularly high, as it has regions with high-variance, high-magnitude rewards. We sampled $m = 100$ states uniformly across the state-space, to provide some insight into the variability of the number of returns sampled across the state-space. We tested $\epsilon \in \{0.01, 0.05, 0.1\}$ and $\delta \in \{0.01, 0.1\}$, and set $\tau = 1.0$. We focus here on how many returns need to be sampled, rather than the trajectory length, and so do not use c nor explicitly compute clipped errors ℓ_c.

The results indicate that EBGStop requires a large number of samples, particularly in Puddle World. Figure 1b for Mountain Car and Figure 1a both indicate that decreasing ϵ from 0.05 to 0.01, to enable higher-accuracy estimates of value function error, causes an exponential increase in the required number of samples, an increase of 10^3 to 10^4 for Mountain Car and 10^5 to 10^6 for Puddle World. An accuracy level of 0.01, which corresponds to difference of 1% for clipped errors, is a typical choice for policy evaluation experiments, yet requires an inordinate number of samples, particularly in Puddle World.

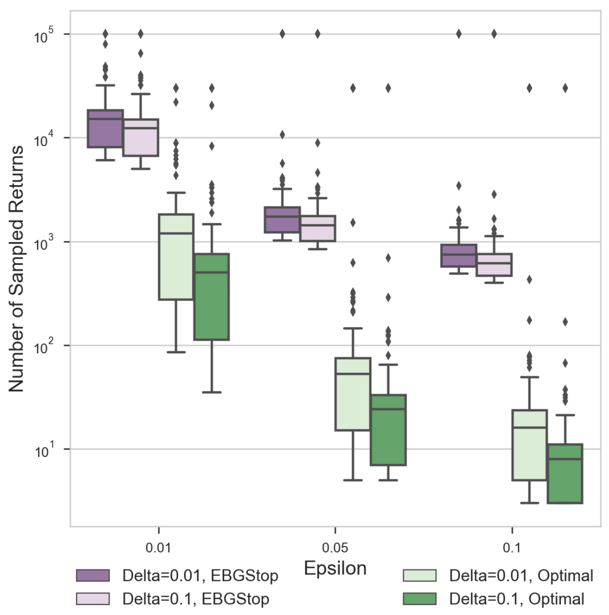
We further investigated lower bounds on the required number of samples. Though EBGStop is a state-of-the-art stopping algorithm, to ensure high-confidence bounds for any distribution with bounded mean and variance, it collects more samples than is actually required. To assess its efficiency gap, we also include an idealistic approach to computing the confidence intervals, using repeated subsamples computed from the simulator. By obtaining many, many estimates of the sample average, using t samples of the truncated return, we can estimate the actual variability of the sample average. We provide additional details in Appendix D. Such a method to compute the confidence interval is not a viable algorithm to reduce the number of samples generated. Rather, the goal here is to report a lower bound on the number of samples required, for comparison and to motivate the amount the sampling algorithm could be improved. The number of samples generated by EBGStop is typically between 10 to 100 times more than the optimal number of samples, which indicates that there is much room to improve sample efficiency.

6 CONCLUSION

In this work, we present the first principled approach to obtain high-confidence error estimates of learned value functions. Our strategy is focused on the setting tackled



(a) Puddle World



(b) Mountain Car

Figure 1: The number of sampled returns to obtain high-confidence estimates $\bar{v}$ of the true values v^*. The y-axis is logarithmic, with many more samples used for smaller ϵ. The box-plot similarly are logarithmic, and so are actually much larger for smaller ϵ than larger ϵ. The accuracy ϵ has a much larger effect on the number of sampled returns that are required, than the probability δ. An additional point of interest is that there are a few states that required significantly more samples for the returns, indicated by the outliers depicted as individual points.

by reinforcement learning empiricists, comparing value function-learning algorithms. In this context, accuracy of value estimates, for multiple algorithms, need to be computed repeatedly, every few steps with increasing data given to the learning algorithms. We provide a general framework for such a setting, where we store estimates of true value functions using samples of truncated returns. The framework for estimating true values for comparison is intentionally generic, to enable any (sample-efficient) stopping algorithm to be used. We propose one solution, which uses empirical Bernstein bounds, to significantly reduce the required number of samples over other concentration inequalities, such as Hoeffding's bound.

This paper highlights several open challenges. As demonstrated in the experiments, there is a large gap between the actual required number of samples and that provided by the algorithm using an empirical Bernstein stopping-rule. For some simulators, this overestimate could result in a prohibitively large number of samples. Although this is a problem more generally faced by the sampling literature, it is particularly exacerbated in reinforcement learning where the variability across states and returns can be high, with large maximum values. An important avenue, then, is to develop more sample-efficient sampling algorithms to make high-confidence error estimates feasible for a broader range of settings in reinforcement learning.

Another open challenge is to address how to sample states $\{s_1, \ldots, s_m\}$. This paper is agnostic to how these states are obtained. However, it is not always straightforward to sample these from a desired distribution. Some choices are simple, such as randomly selecting these across the state space. For other cases, it is more complicated, such as sampling these from the stationary distribution of the behaviour policy, d^μ. The typical strategy is to run μ for a burn-in period, so that afterwards it is more likely for states to be sampled from the stationary distribution. The theoretical effectiveness of this strategy, however, is not yet well-understood. There has been work estimating empirical mixing times [Hsu et al., 2015] and some work bounding the number of samples required for burn-in [Paulin, 2015]. Nonetheless, it remains an important open question on how to adapt these results for the general reinforcement learning setting.

One goal of this paper has been to highlight an open problem that has largely been ignored by reinforcement learning empiricists. We hope for this framework to stimulate further work in high-confidence estimates of value function accuracy.

References

Matthieu Geist and Bruno Scherrer. Off-policy learning with eligibility traces: a survey. *The Journal of Machine Learning Research*, 2014.

Adam M White and Martha White. Investigating practical, linear temporal difference learning. In *International Conference on Autonomous Agents and Multiagent Systems*, 2016.

Adam White. *Developing a predictive approach to knowledge*. PhD thesis, University of Alberta, 2015.

Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *International Conference on Machine Learning*, 1995.

Richard S Sutton, A R Mahmood, and Martha White. An emphatic approach to the problem of off-policy temporal-difference learning. *The Journal of Machine Learning Research*, 2016.

R Sutton, C Szepesvári, A Geramifard, and Michael Bowling. Dyna-style planning with linear function approximation and prioritized sweeping. In *Conference on Uncertainty in Artificial Intelligence*, 2008.

Simon S Du, Jianshu Chen, Lihong Li, Lin Xiao, and Dengyong Zhou. Stochastic Variance Reduction Methods for Policy Evaluation. In *International Conference on Machine Learning*, 2017.

HR Maei, C Szepesvári, S Bhatnagar, D Precup, D Silver, and Richard S Sutton. Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in Neural Information Processing Systems*, 2009.

Clement Gehring, Yangchen Pan, and Martha White. Incremental Truncated LSTD. In *International Joint Conference on Artificial Intelligence*, 2016.

Yangchen Pan, Adam White, and Martha White. Accelerated Gradient Temporal Difference Learning. In *International Conference on Machine Learning*, 2017.

Lei Le, Raksha Kumaraswamy, and Martha White. Learning Sparse Representations in Reinforcement Learning with Sparse Coding. In *International Joint Conference on Artificial Intelligence*, 2017.

Christoph Dann, Gerhard Neumann, and Jan Peters. Policy evaluation with temporal differences: a survey and comparison. *The Journal of Machine Learning Research*, 2014.

J Boyan and A W Moore. Generalization in Reinforcement Learning: Safely Approximating the Value Function. *Advances in Neural Information Processing Systems*, 1995.

George Konidaris, Scott Niekum, and Philip S Thomas. TDgamma: Re-evaluating Complex Backups in Temporal Difference Learning. In *Advances in Neural Information Processing Systems*, 2011.

William Dabney and Philip S Thomas. Natural Temporal Difference Learning. In *AAAI Conference on Artificial Intelligence*, 2014.

Pedro M Domingos and Geoff Hulten. A General Method for Scaling Up Machine Learning Algorithms and its Application to Clustering. In *International Conference on Machine Learning*, 2001.

Paul Dagum, Richard Karp, Michael Luby, and Sheldon Ross. An Optimal Algorithm for Monte Carlo Estimation. *SIAM Journal on Computing*, 2006.

Volodymyr Mnih, Csaba Szepesvari, and Jean-Yves Audibert. Empirical Bernstein stopping. In *the 25th international conference*, 2008.

Daniel Hsu, Aryeh Kontorovich, and Csaba Szepesvari. Mixing Time Estimation in Reversible Markov Chains from a Single Sample Path. In *Advances in Neural Information Processing Systems*, 2015.

Daniel Paulin. Concentration inequalities for Markov chains by Marton couplings and spectral methods. *Electronic Journal of Probability*, 2015.

Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvari. Tuning Bandit Algorithms in Stochastic Environments. In *Algorithmic Learning Theory*. 2007.

BP Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 1962.