
Comparing Direct and Indirect Temporal-Difference Methods for Estimating the Variance of the Return

Craig Sherstan¹, Dylan R. Ashley², Brendan Bennett², Kenny Young,
Adam White, Martha White, Richard S. Sutton

Reinforcement Learning and Artificial Intelligence Laboratory
Department of Computing Science, University of Alberta, Edmonton, Canada

1. Corresponding author: sherstan@ualberta.ca

2. Co-authors contributed equally.

Abstract

Temporal-difference (TD) learning methods are widely used in reinforcement learning to estimate the expected return for each state, without a model, because of their significant advantages in computational and data efficiency. For many applications involving risk mitigation, it would also be useful to estimate the *variance* of the return by TD methods. In this paper, we describe a way of doing this that is substantially simpler than those proposed by Tamar, Di Castro, and Mannor in 2012, or those proposed by White and White in 2016. We show that two TD learners operating in series can learn expectation and variance estimates. The trick is to use the square of the TD error of the expectation learner as the reward of the variance learner, and the square of the expectation learner’s discount rate as the discount rate of the variance learner. With these two modifications, the variance learning problem becomes a conventional TD learning problem to which standard theoretical results can be applied. Our formal results are limited to the table lookup case, for which our method is still novel, but the extension to function approximation is immediate, and we provide some empirical results for the linear function approximation case. Our experimental results show that our direct method behaves just as well as a comparable indirect method, but is generally more robust.

1 INTRODUCTION

Conventionally, in reinforcement learning (RL) the agent estimates the expected value of the return—the discounted sum of future rewards—as an intermediate step

to finding an optimal policy. The agent estimates the value function by averaging the returns observed from each state in a trajectory of experiences. To estimate this value function online—while the trajectory is still unfolding—we update the agent’s value estimates towards the expected return. Algorithms that estimate the expected value of the return in this way are called temporal-difference (TD) learning methods. However, it is reasonable to consider estimating other functions of the return beyond the first moment. For example, Belle-mare et al. (2017) estimated the distribution of returns explicitly. In this paper, we focus on estimating the variance of the return using TD methods.

The variance of the return can be used to design algorithms which account for risk in decision making. The main approach is to formulate the agent’s objective as maximizing reward, while minimizing the variance of the return (Sato et al., 2001; Prashanth and Ghavamzadeh, 2013; Tamar et al., 2012).

An estimate of the variance of the return can also be useful for adapting the parameters of a learning system automatically, thus avoiding time-consuming, human-driven meta parameter tuning. Sakaguchi and Takano (2004) used the variance estimate explicitly in the decision making policy to set the temperature variable in the Boltzmann action selection rule. Using variance in this way can automatically adjust the amount of exploration, allowing the learning system to adapt to new circumstances online. Conventionally, this temperature would either be set to a constant or decayed according to a fixed schedule. In either circumstance, the performance can be quiet poor in non-stationary domains, and a human expert is required to select the constant value or fixed schedule. Similarly, White and White (2016) estimated the variance of the return to automatically adapt the trace-decay parameter, λ , used in learning updates of TD algorithms (see Section 2 for an explanation of the role of λ). Not only does this approach avoid the need to tune λ by hand, but it can result in faster learning.

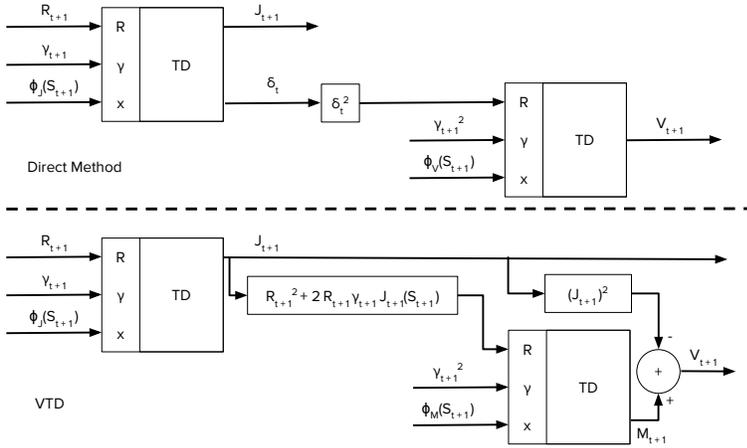


Figure 1: Each TD node takes as input a reward R , a discounting function γ , and features ϕ . For the direct method (**top**) the squared TD error of the first-stage value estimator is used as the meta-reward for the second-stage V estimator. For VTD (**bottom**), a more complex computation is used for the meta-reward and an extra stage of computation is required.

The variance V of the return can be estimated either directly or indirectly. Indirect estimation involves computing an estimate of variance from estimates of the first and second moments. Sobel (1982) was the first to formulate Bellman operators for the second moment and showed how this could be used to compute variance indirectly. This is the approach used by Tamar et al. (2016), Tamar and Mannor (2013), and Prashanth and Ghavamzadeh (2013). White and White (2016) introduced several extensions to this indirect method including estimation of the λ -return (Sutton and Barto, 1998), support for off-policy learning (Sutton, Maei, et al., 2009; Maei, 2011), and state-dependent discounting (Sutton, Modayil, et al., 2011; White, 2017). Their method, which they refer to as VTD, serves as the indirect estimation algorithm used in this paper. We note that an alternative method, which we do not investigate here, could be to estimate the distribution of returns as done by Bellemare et al. (2017) and compute the variance from this estimated distribution.

Variance may also be estimated directly. Tamar et al. (2012) gave a direct algorithm but restricted it to estimating cost-to-go returns in a strictly episodic manner, i.e., estimates are only updated after an entire trajectory has been captured. We introduce a new algorithm for directly estimating the variance of the return incrementally using TD methods. Our algorithm uses two TD learners, one for estimating value and the other for estimating the variance of the return. These estimators operate in series with the squared TD error of the value learner serving as the reward of the variance learner and the squared discount rate of the value learner serving as the discount rate of the variance learner. Like VTD (White and White, 2016), our algorithm supports estimating the variance of the λ -return, state-dependent discounting, estimating the variance of the on-policy return from off-policy samples, and estimating the variance of the off-policy return from on-policy samples (Section 3.2 motivates these extensions). We call our new algorithm Direct Variance TD

(DVTD). We recognize that the algorithm of Sato et al. (2001) can be seen as the simplest instance of our algorithm, using the on-policy setting with fixed discounting and no traces¹. Sakaguchi and Takano (2004) also used this simplified algorithm, but treated the discount of the variance estimator as a free parameter.

We introduce a Bellman operator for the variance of the return, and further prove that, even for a value function that does not satisfy the Bellman operator for the expected return, the error in this recursive formulation is proportional to the error in the value function estimate. Interestingly, the Bellman operator for the second moment requires an unbiased estimate of the return (White and White, 2016). Since our Bellman operator for the variance avoids this term, it has a simpler update. As shown in Figure 1, Both DVTD and VTD can be seen as a network of two TD estimators running sequentially. Note, that we restrict our formal derivations and subsequent analysis to the table lookup setting.

Our primary goal is to understand the empirical properties of the direct and indirect approaches. In general, we found that DVTD is just as good as VTD and in many cases better. We observe that DVTD behaves better in the early stages of learning before the value function has converged. Furthermore, we observe that the variance of the estimates of V can be higher for VTD under several circumstances: (1) when there is a mismatch in step-sizes between the value estimator and the V estimator, (2) when traces are used with the value estimator, (3) when estimating V of the off-policy return, and (4) when there is error in the value estimate. Finally, we observe significantly better performance of DVTD in a linear function approximation setting. Overall, we conclude that the direct approach to estimating V , DVTD, is both simpler and better behaved than VTD.

¹Dimitrakakis (2006) used a related TD method, which estimates the squared TD error

2 THE MDP SETTING

We model the agent’s interaction with the environment as a finite Markov decision process (MDP) consisting of a finite set of states \mathcal{S} , a finite set of actions, \mathcal{A} , and a transition model $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ defining the probability $p(s'|s, a)$ of transitioning from state s to s' when taking action a . In the policy evaluation setting considered in this paper, the agent follows a fixed policy $\pi(a|s) \in [0, 1]$ that provides the probability of taking action a in state s . At each timestep the agent receives a random reward R_{t+1} , dependent only on S_t, A_t, S_{t+1} . The return is the discounted sum of future rewards

$$\begin{aligned} G_t &= R_{t+1} + \gamma_{t+1}R_{t+2} + \gamma_{t+1}\gamma_{t+2}R_{t+3} + \dots \\ &= R_{t+1} + \gamma_{t+1}G_{t+1}. \end{aligned} \quad (1)$$

where $\gamma \in [0, 1]$ specifies the degree to which future rewards are discounted. Note that we define discounting as state-dependent such that $\gamma_{t+1} \equiv \gamma(S_{t+1})$. This allows us to combine the specification of continuing and episodic tasks. Further implications of this are discussed in Section 3.2.

The value of a state, $j(s)$, is defined as the expected return from state s under a particular policy π :

$$j(s) = \mathbb{E}_\pi[G_t | S_t = s]. \quad (2)$$

We use j to indicate the true value function and J the estimate. The TD-error is the difference between the one-step approximation and the current estimate:

$$\delta_t = R_{t+1} + \gamma_{t+1}J_t(S_{t+1}) - J_t(S_t). \quad (3)$$

This can then be used to update the value estimator using a TD method, such as TD(0) as follows:

$$J(s)_{t+1} = J(s)_t + \alpha\delta_t \quad (4)$$

3 ESTIMATING THE VARIANCE OF THE RETURN

For clarity of presentation, we first discuss the simplest version of both the direct and indirect methods and present the full algorithms in Section 3.2.

The direct TD method uses both a value estimator and a variance estimator. The *value estimator* provides an estimate of the expected return. The *variance estimator*, on the other hand, uses the value estimator to provide an estimate of the variance of the return. Since we use TD methods for both the value and variance estimators we need to adopt additional notation; variables with a bar are used by either the second moment or variance estimator. Otherwise, they are used by the value estimator.

The key to using both the indirect and direct methods as TD methods is to provide a discounting function, $\bar{\gamma}$, and a meta-reward, \bar{R} . In the following, we present a simplified TD(0) version of both algorithms.

Simplified Direct Variance Algorithm

$$\begin{aligned} \bar{\gamma}_{t+1} &\leftarrow \gamma_{t+1}^2 \\ \bar{R}_{t+1} &\leftarrow \delta_t^2 \\ \bar{\delta}_t &\leftarrow \bar{R}_{t+1} + \bar{\gamma}_{t+1}V_t(s') - V_t(s) \\ V_{t+1}(s) &\leftarrow V_t(s) + \bar{\alpha}\bar{\delta}_t \end{aligned} \quad (5)$$

Simplified Second Moment Algorithm

$$\begin{aligned} \bar{\gamma}_{t+1} &\leftarrow \gamma_{t+1}^2 \\ \bar{R}_{t+1} &\leftarrow R_{t+1}^2 + 2\gamma_{t+1}R_{t+1}J_{t+1}(s') \\ \bar{\delta}_t &\leftarrow \bar{R}_{t+1} + \bar{\gamma}_{t+1}M_t(s') - M_t(s) \\ M_{t+1}(s) &\leftarrow M_t(s) + \bar{\alpha}\bar{\delta}_t \\ V_{t+1}(s) &\leftarrow M_{t+1}(s) - J_{t+1}(s)^2 \end{aligned} \quad (6)$$

3.1 DERIVATION OF THE DIRECT METHOD

We now derive the direct method for estimating the variance of the return. Again, for clarity, we only consider the simple case described in Section 3 (See Appendix B for a derivation of the more general extended algorithm).

The derivation of the direct method follows from characterizing the Bellman operator for the variance of the return: Theorem 1 gives a Bellman equation for the variance v . It has the form of a TD target with meta-reward $\bar{R}_t = \delta_t^2$ and discounting function $\bar{\gamma}_{t+1} = \gamma_{t+1}^2$. Therefore, we can estimate V using TD methods. The Bellman operators for the variance are general, in that they allow for either the episodic or continuing setting, by using variable γ . By directly estimating variance, we avoid a second term in the cumulant that is present in approaches that estimate the second moment (Tamar and Mannor, 2013; Tamar et al., 2016; White and White, 2016).

To have a well-defined solution to the fixed point, we need the discount to be less than one for some transition (White, 2017; Yu, 2015). This corresponds to assuming that the policy is proper, for the cost-to-go setting (Tamar et al., 2016).

Assumption 1. *The policy reaches a state s where $\gamma(s) < 1$ in a finite number of steps.*

Theorem 1. *For any $s \in \mathcal{S}$,*

$$\begin{aligned} j(s) &= \mathbb{E}[R_{t+1} + \gamma_{t+1}j(S_{t+1}) | S_t = s] \\ v(s) &= \mathbb{E}[\delta_t^2 + \gamma_{t+1}^2v(S_{t+1}) | S_t = s] \end{aligned} \quad (7)$$

Proof. First we expand $G_t - j(S_t)$, from which we re-

cover a series with the form of a return.

$$\begin{aligned} G_t - j(S_t) &= R_{t+1} + \gamma_{t+1}G_{t+1} - j(S_t) \\ &= R_{t+1} + \gamma_{t+1}j(S_{t+1}) - j(S_t) + \gamma_{t+1}(G_{t+1} - j(S_{t+1})) \\ &= \delta_t + \gamma_{t+1}(G_{t+1} - j(S_{t+1})) \end{aligned} \quad (8)$$

The variance of G_t is therefore

$$\begin{aligned} v(s) &= \mathbb{E} \left[(G_t - \mathbb{E}[G_t | S_t = s])^2 | S_t = s \right] \\ &= \mathbb{E} \left[(G_t - j(s))^2 | S_t = s \right] \\ &= \mathbb{E} \left[(\delta_t + \gamma_{t+1}(G_{t+1} - j(S_{t+1})))^2 | S_t = s \right] \\ &= \mathbb{E} [\delta_t^2 | S_t = s] \\ &\quad + \mathbb{E} [\gamma_{t+1}^2 (G_{t+1} - j(S_{t+1}))^2 | S_t = s] \\ &\quad + 2\mathbb{E} [\gamma_{t+1}\delta_t(G_{t+1} - j(S_{t+1})) | S_t = s] \end{aligned} \quad (9)$$

Equation (7) follows from Lemma 1 in Appendix B which shows $\mathbb{E}[\gamma_{t+1}\delta_t(G_{t+1} - j(S_{t+1})) | S_t = s] = 0$. Similar to Lemma 1, using the law of total expectation, $\mathbb{E}[\gamma_{t+1}^2(G_{t+1} - j(S_{t+1}))^2 | S_t = s] = \mathbb{E}[\gamma_{t+1}^2 v(S_{t+1}) | S_t = s]$. \square

We provide an initial characterization of error in the variance estimate obtained under this recursion, when an approximate value function rather than the true value function is used. As we show in the below theorem, the resulting error in the variance estimator is proportional to the squared error in the value estimate, and discounted accumulated errors into the future. If the approximation error is small, we expect this accumulated error to be small, particularly as the accumulation errors are signed and so can cancel, and because they are discounted. However, more needs to be done to understand the impact of this accumulated error.

Theorem 2. *For approximate value function J with variance estimate $V(s) = \mathbb{E}[\delta_t^2 + \gamma_{t+1}^2 V(S_{t+1}) | S_t = s]$, if there exists $\epsilon : \mathcal{S} \rightarrow [0, \infty)$ bounding squared value estimation error $(J(s) - j(s))^2 \leq \epsilon(s)$ and accumulation error $|\mathbb{E}[\gamma_{t+1}\delta_t(j(S_{t+1}) - J(S_{t+1})) + \gamma_{t+1}^2\gamma_{t+2}\delta_{t+1}(j(S_{t+2}) - J(S_{t+2})) + \dots | S_t = s]| \leq \epsilon(s)$, then*

$$|v(s) - \mathbb{E}[\delta_t^2 + \gamma_{t+1}^2 V(S_{t+1}) | S_t = s]| \leq 3\epsilon(s)$$

Proof. We can re-express the true variance in terms of the approximation J , as

$$\begin{aligned} v(s) &= \mathbb{E} [(G_t - j(s) + J(s) - J(s))^2 | S_t = s] \\ &= \mathbb{E} [(G_t - J(s))^2 | S_t = s] + (J(s) - j(s))^2 \\ &\quad + 2\mathbb{E}[G_t - J(s) | S_t = s] (J(s) - j(s)) \end{aligned} \quad (10)$$

This last term simplifies to

$$\begin{aligned} \mathbb{E}[G_t - J(s) | S_t = s] &= \mathbb{E}[G_t - j(s) | S_t = s] + j(s) - J(s) \\ &= j(s) - J(s) \end{aligned} \quad (11)$$

giving $(J(s) - j(s))^2 + 2(j(s) - J(s))(J(s) - j(s)) = -(J(s) - j(s))^2$. We can use the same recursive form as (9), but with J , giving

$$\begin{aligned} \mathbb{E}[(G_t - J(s))^2 | S_t = s] &= \mathbb{E}[\delta_t^2 + \gamma_{t+1}^2 V(S_{t+1}) | S_t = s] \\ &\quad + 2\mathbb{E}[\gamma_{t+1}\delta_t(G_{t+1} - J(S_{t+1})) | S_t = s] \\ &\quad + 2\mathbb{E}[\gamma_{t+1}^2\gamma_{t+2}\delta_{t+1}(G_{t+2} - J(S_{t+2})) | S_t = s] + \dots \end{aligned} \quad (12)$$

where the terms involving $\delta_t(G_{t+1} - J(S_{t+1}))$ accumulate. Notice that

$$\begin{aligned} &|\mathbb{E}[\gamma_{t+1}\delta_t(G_{t+1} - J(S_{t+1})) | S_t = s]| \\ &= |\mathbb{E}[\gamma_{t+1}\delta_t(G_{t+1} - j(S_{t+1})) | S_t = s] \\ &\quad + \mathbb{E}[\gamma_{t+1}\delta_t(j(S_{t+1}) - J(S_{t+1})) | S_t = s]| \\ &= |\mathbb{E}[\gamma_{t+1}\delta_t(j(S_{t+1}) - J(S_{t+1})) | S_t = s]| \end{aligned}$$

where the second equality follows from Lemma 1. By the same argument as in Lemma 1, this will also hold true for all the other terms in (12). By assumption, the sum of all these covariance terms between j and J are bounded by $\epsilon(s)$. Putting this together, we get

$$\begin{aligned} |v(s) - V(s)| &= |v(s) - \mathbb{E}[\delta_t^2 + \gamma_{t+1}^2 V(S_{t+1}) | S_t = s]| \\ &\leq 2\epsilon(s) + (J(s) - j(s))^2 \leq 3\epsilon(s) \end{aligned} \quad \square$$

3.2 THE EXTENDED DIRECT METHOD

Here, we extend the direct method to support estimating the λ -return, state-dependent γ , eligibility traces and off-policy estimation, just as White and White, 2016 did with VTD (derivation provided in Appendix B). We first explain each of these extensions before providing our full direct algorithm and VTD.

The λ -return is defined as

$$G_t^\lambda = R_{t+1} + \gamma_{t+1}(1 - \lambda_{t+1})J_t(S_{t+1}) + \gamma_{t+1}\lambda_{t+1}G_{t+1}^\lambda$$

and provides a bias-variance trade-off by incorporating J , which is a potentially lower-variance but biased estimate of the return. This trade-off is determined by a state-dependent trace-decay parameter, $\lambda_t \equiv \lambda(S_t) \in [0, 1]$. When $J_t(S_{t+1})$ is equal to the expected return from $S_{t+1} = s$, then $\mathbb{E}_\pi[(1 - \lambda_{t+1})J_t(S_{t+1}) + \gamma_{t+1}\lambda_{t+1}G_{t+1}^\lambda | S_{t+1} = s] = \mathbb{E}_\pi[G_{t+1}^\lambda | S_{t+1} = s]$, and so the λ -return is unbiased. Beneficially the expected value $J_t(S_{t+1})$ is lower-variance than the sample G_{t+1}^λ . If J_t is inaccurate, however, some bias is introduced. Therefore, when $\lambda = 0$, the λ -return is lower-variance but can be biased. When $\lambda = 1$, the λ -return equals the Monte Carlo return (Equation (1)); in this case, the update target exhibits more variance, but no bias. In the tabular setting evaluated in this paper, λ does not affect

Table 1: Algorithm Notation

J	estimated value function of the target policy π .
M	estimate of the second moment.
V	estimate of the variance.
R, \bar{R}	meta-reward used by the J and (M, V) estimators.
λ	bias-variance parameter of the target λ -return.
$\kappa, \bar{\kappa}$	trace-decay parameter of the J and (M, V) estimators.
$\gamma, \bar{\gamma}$	discounting function used by J and (M, V) estimators.
$\delta_t, \bar{\delta}_t$	TD error of the J and (M, V) estimators at time t .
$\bar{\rho}$	importance sampling ratio for estimating the variance of the target return from off-policy samples.
η	importance sampling ratio used to estimate the variance of the off-policy return.

the fixed point solution of the value estimate, only the rate at which learning occurs. It does, however, affect the observed variance of the return, which we estimate. The λ -return is implemented using traces as in the following TD(λ) algorithm, shown with accumulating traces:

$$\begin{aligned}
 e_t(s) &\leftarrow \begin{cases} \gamma_t \lambda_t e_{t-1}(s) + 1 & s = S_t \\ \gamma_t \lambda_t e_{t-1}(s) & \forall s \in \mathcal{S}, s \neq S_t \end{cases} \\
 J_{t+1}(S_t) &\leftarrow J_t(S_t) + \alpha \delta_t e_t(S_t)
 \end{aligned} \tag{13}$$

For notational purposes, we define the trace parameter for the value and secondary estimators as κ and $\bar{\kappa}$ respectively. Both of these parameters are independent of the λ -return for which we estimate the variance. That is, we are entirely free to estimate the variance of the λ -return for any value of λ independently of the use of any traces in either the value or secondary estimator.

State-Dependent γ . While most RL methods focus on fixed discounting values, it is straightforward to use state-based discounting (Sutton, Modayil, et al., 2011), where $\gamma_t \equiv \gamma(S_t)$ (White (2017) go further by defining transition based discounting). This generalization enables a wider variety of returns to be considered. First, it allows a convenient means of describing both episodic and continuing tasks and provides an algorithmic mechanism for terminating an episode without defining a recurrent terminal state explicitly. Further, it allows for event-based terminations (Sutton, Modayil, et al., 2011). It also enables soft terminations which may prove useful when training an agent with sub-goals (White, 2017). The use of state-dependent discounting functions is relatively new and remains to be extensively explored.

Off-policy learning. Value estimates are made with respect to a target policy, π . If the behavior policy, $\mu = \pi$ then we say that samples are collected on-policy, otherwise, the samples are collected off-policy. An off-policy learning approach is to weight each update by the importance sampling ratio: $\rho_t = \frac{\pi(S_t, A_t)}{\mu(S_t, A_t)}$. There are two different scenarios to be considered when estimating the

variance of the return in the off-policy setting. The first is estimating the variance of the on-policy return of the target policy while following a different behavior policy. The second has the goal of estimating the variance of the off-policy return itself. The off-policy λ -return is:

$$G_t^{\lambda; \rho} = \rho_t (R_{t+1} + \gamma_{t+1} (1 - \lambda_{t+1}) j_t(S_{t+1}) + \gamma_{t+1} \lambda_{t+1} G_{t+1}^{\lambda; \rho}). \tag{14}$$

where the multiplication by the potentially large importance sampling ratios can significantly increase variance.

It is important to note you would only ever estimate one or the other of these settings with a given estimator. Let η be the weighting for the value estimator, and $\bar{\rho}$ the weighting for the variance estimator. If estimating the variance of the target return from off-policy samples, the first scenario, $\eta_t = 1 \forall t$ and $\bar{\rho}_t = \rho_t$. If estimating the variance of the off-policy return $\bar{\rho}_t = 1 \forall t$ and $\eta_t = \rho_t$.

3.2.1 The Extended Algorithms

To estimate V , our method uses both value and variance estimators. The *value estimator* provides an estimate of the expected return. The *variance estimator*, on the other hand, uses the value estimator to provide an estimate of the variance of the return. Our method, DVTD, and the indirect method, VTD, can be seen as simply defining a meta-reward and a discounting function and can thus be learned with any known TD method, such as TD with accumulating traces as shown in Equation 13. Table 1 summarizes our notation.

Direct Variance Algorithm - DVTD

$$\begin{aligned}
 \bar{R}_{t+1} &\leftarrow (\eta_t \delta_t + (\eta_t - 1) J_{t+1}(s))^2 \\
 \bar{\gamma}_{t+1} &\leftarrow \gamma_{t+1}^2 \lambda_{t+1}^2 \eta_t^2 \\
 \bar{\delta}_t &\leftarrow \bar{R}_{t+1} + \bar{\gamma}_{t+1} V_t(s') - V_t(s) \\
 \bar{e}_t(s) &\leftarrow \begin{cases} \bar{\rho}_t (\bar{\gamma}_t \bar{\kappa}_t \bar{e}_{t-1}(s) + 1) & s = S_t \\ \bar{\rho}_t (\bar{\gamma}_t \bar{\kappa}_t \bar{e}_{t-1}(s)) & \forall s \in \mathcal{S}, s \neq S_t \end{cases} \\
 V_{t+1}(s) &\leftarrow V_t(s) + \bar{\alpha} \bar{\delta}_t \bar{e}_t(s)
 \end{aligned} \tag{15}$$

We also present the full VTD algorithm below (again, shown with accumulating traces). Note that this algorithm does not impose that the variance be non-negative.

Second Moment Algorithm - VTD

$$\begin{aligned}
\bar{G}_t &\leftarrow R_{t+1} + \gamma_{t+1}(1 - \lambda_{t+1})J_{t+1}(s') \\
\bar{R}_{t+1} &\leftarrow \eta_t^2 \bar{G}_t^2 + 2\eta_t^2 \gamma_{t+1} \lambda_{t+1} \bar{G}_t J_{t+1}(s') \\
\bar{\gamma}_{t+1} &\leftarrow \eta_t^2 \gamma_{t+1}^2 \lambda_{t+1}^2 \\
\bar{\delta}_t &\leftarrow \bar{R}_{t+1} + \bar{\gamma}_{t+1} M_t(s') - M_t(s) \\
\bar{e}_t(s) &\leftarrow \begin{cases} \bar{\rho}_t(\bar{\gamma}_t \bar{\kappa}_t \bar{e}_{t-1}(s) + 1) & s = S_t \\ \bar{\rho}_t(\bar{\gamma}_t \bar{\kappa}_t \bar{e}_{t-1}(s)) & \forall s \in \mathcal{S}, s \neq S_t \end{cases} \\
M_{t+1}(s) &\leftarrow M_t(s) + \bar{\alpha} \bar{\delta}_t \bar{e}_t(s) \\
V_{t+1}(s) &\leftarrow M_{t+1}(s) - J_{t+1}(s)^2
\end{aligned} \tag{16}$$

4 EXPERIMENTS

The primary purpose of these experiments is to demonstrate that both algorithms can approximate the true expected V under various conditions in the tabular setting. We consider two domains. The first is a deterministic chain, which is useful for basic evaluation and gives results which are easy to interpret (Figure 2). The second is a randomly generated MDP, with different discount and trace-decay parameters in each state (Figure 3). For all experiments Algorithm 13 is used as the value estimator. Unless otherwise stated, traces are not used ($\kappa = \bar{\kappa} = 0$) and estimates were initialized to zero. For each experimental setting the average of 30 separate experiments is presented with standard deviation shown as shaded regions. True values were determined by Monte Carlo estimation and are shown as dashed lines in the figures.

We look at the effects of relative step-size between the value estimator and the variance estimators in Section 4.1. Then, in Section 4.2 we use the random MDP to show that both algorithms can estimate the variance with state-dependent γ and λ . In Section 4.3 we evaluate the two algorithms' responses to errors in the value estimate. Section 4.4 looks at the effect of using traces in the estimation method. We then examine the off-policy setting in Section 4.5. Finally, Section 4.6 provides experimental results in a linear function approximation setting.

4.1 THE EFFECT OF STEP-SIZE

We use the chain MDP to investigate the impact of step-size choice. In Figure 4(a) all step-sizes are the same ($\alpha = \bar{\alpha} = 0.001$) and here both algorithms behave similarly. For Figure 4(b) the step-size of the value estimate, ($\alpha = 0.01$), is greater than that of the secondary estimators, ($\bar{\alpha} = 0.001$). Now DVTD smoothly approaches the

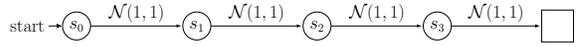


Figure 2: **Chain MDP** with 4 non-terminal states and 1 terminal state. From non-terminal states there is a single action with a deterministic transition to the right. On each transition, rewards are drawn from a normal distribution with mean and variance of 1.0. Evaluation was performed for $\lambda = 0.9$, which was chosen because it is not at either extreme and because 0.9 is a commonly used value for many RL experimental domains.

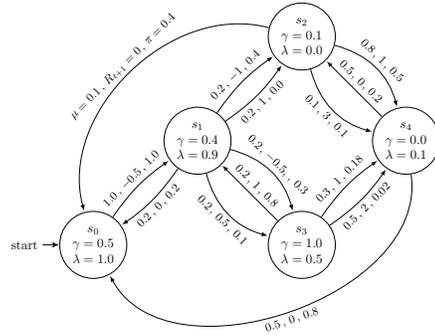


Figure 3: **Random MDP**, with a stochastic policy and state-based γ and λ . The state-dependent values of γ and λ are chosen to provide a range of values, with at least one state acting as a terminal state where $\gamma = 0$. On-policy action probabilities are indicated by μ and off-policy ones by π .

correct value, while VTD first dips well below zero. This is expected as the estimates are initialized to zero and the variance is calculated as $V(s) = M(s) - J(s)^2$. If the second moment lags behind the value estimate, then the variance will be negative. In Figure 4(c) the step-size for the secondary estimators is larger than for the value estimator ($0.001 = \alpha < \bar{\alpha} = 0.01$). While both methods overshoot the target in this example, VTD has greater overshoot. For both cases of unequal step-size, we see higher variance in the estimates for VTD.

Figure 5 explores this further. Here the value estimator is initialized to the true values and updates are turned off ($\alpha = 0$). The secondary estimators are initialized to zero and learn with $\bar{\alpha} = 0.001$, chosen simply to match the step-sizes used in the previous experiments. Despite being given the true values the VTD algorithm produces higher variance in its estimates, suggesting that VTD is dependent on the value estimator tracking.

This sensitivity to step-size is shown in Figure 6. All estimates are initialized to their true values. For each ratio, we computed the average variance of the 30 runs of 2000

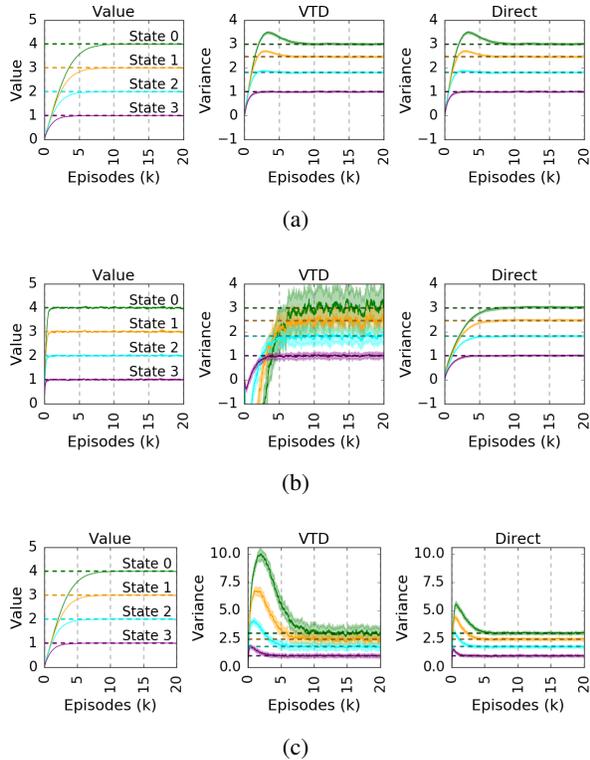


Figure 4: **Chain MDP** ($\lambda = 0.9$). Varying the ratio of step-size between value and variance estimators. **a)** Step-sizes equal. $\alpha = \bar{\alpha} = 0.001$. **b)** Variance step-size smaller. $\alpha = 0.01, \bar{\alpha} = 0.001$. **c)** Variance step-size larger. $\alpha = 0.001, \bar{\alpha} = 0.01$. We see greater variance in the estimates and greater over/undershoot for VTD when step-sizes are not equal.

episodes. We can see that DVTD is largely insensitive to step-size ratio, but that VTD has higher mean squared error (MSE) except when the step-sizes are equal. This result holds for the other experimental settings of this paper, including the random MDP, but further results are omitted for brevity.

Would there ever be a situation where different step-sizes between value and secondary estimators is justified? The automatic tuning of parameters, such as step-size, is an important area of research, seeking to make learning algorithms more efficient, robust and easier to deploy. Methods which automatically set the step-sizes may produce different values specific to the performance of each estimator. One such algorithm is ADADELTA, which adapts the step-size based on the TD error of the estimator (Zeiler, 2012). Figure 7 shows that using a separate ADADELTA ($\rho = 0.99, \epsilon = 1e-6$) step-size calculation for each estimator results in higher variance for VTD as expected, given that the value estimator and VTD produce different TD errors.

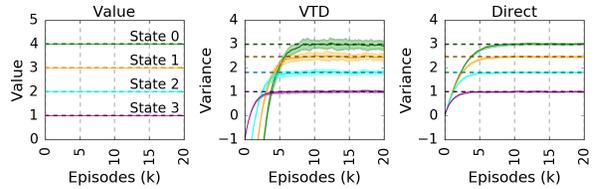


Figure 5: **Chain MDP** ($\lambda = 0.9$). Value estimate held fixed at the true values ($\alpha = 0, \bar{\alpha} = 0.001$). Notice the increased estimate variance for VTD, especially State 0.

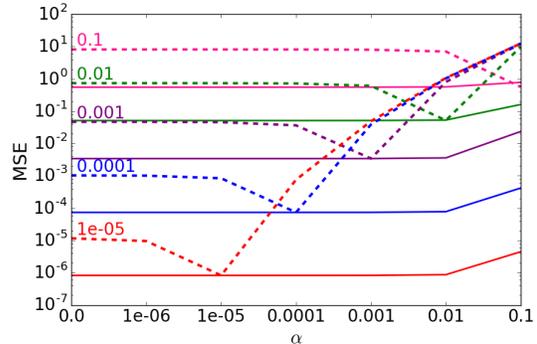


Figure 6: **Chain MDP** ($\lambda = 0.9$). The MSE summed over all states as a function of ratios between the value step-size α (shown along the x-axis) and the variance step-size $\bar{\alpha}$ (shown along the 5 series). The direct algorithm is indicated by the solid lines, and VTD is indicated by the dashed. The MSE of the VTD algorithm is higher than the direct algorithm, except when the step-size is the same for all estimators, $\alpha = \bar{\alpha}$ or for very small $\bar{\alpha}$.

4.2 STATE-DEPENDENT γ AND λ .

One of the contributions of VTD was the generalization to support state-based γ and λ . Here we evaluate the random MDP from Figure 3 (in the on-policy setting, using μ), which was designed for this scenario and which has a stochastic policy, is continuing, and has multiple possible actions from each state. Both algorithms achieved similar results (see Appendix A).

4.3 VARIABLE ERROR IN THE VALUE ESTIMATES

The derivation of our DVTD assumes access to the true value function. The experiments of the previous sections demonstrate that both methods are robust under this assumption, in the sense that the value function was estimated from data and used to estimate V . It remains unclear, however, how well these methods perform when the value estimates converge to biased solutions.

To examine this, we again use the random MDP shown

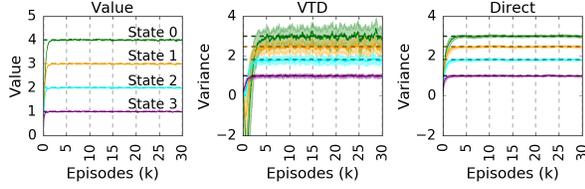


Figure 7: **Chain MDP** ($\lambda = 0.9$). Results using ADADELTA algorithm to automatically and independently set the step-sizes α and $\bar{\alpha}$. The step-sizes produced are given in Appendix D.

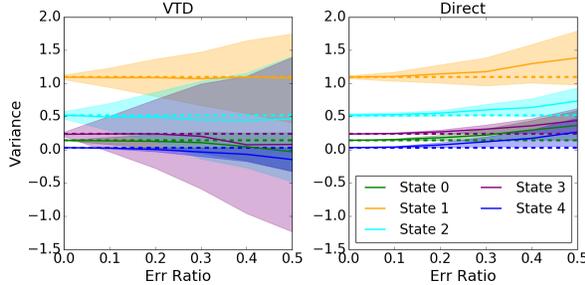


Figure 8: **Random MDP**. For each run, the value estimate of each state is offset by a random noise drawn from a uniform distribution whose size is a function of an error ratio and the maximum true value in the MDP. Standard deviation of the estimates is shown by shading.

by Figure 3. True values for the value functions and variance estimates are calculated from Monte Carlo simulation of 10,000,000 timesteps. For each run of the experiment each state of the value estimator was initialized to the true value plus an error ($J(s)_0 = j(s) + \epsilon(s)$) drawn from a uniform distribution: $\epsilon(s) \in [-\zeta, \zeta]$, where $\zeta = \max_s(|v(s)|) * \text{err ratio}$ (the maximum value in this domain is 1.55082409). The value estimate was held constant throughout the run ($\alpha = 0.0$). The experiment consisted of 120 runs of 80,000 timesteps. To look at the steady-state response of the algorithms we use only the last 10,000 timesteps in our calculations. Figure 8 plots the average variance estimate for each state with the average standard deviation of the estimates as the shaded regions. Sweeps over step-size were conducted, $\bar{\alpha} \in [0.05, 0.04, 0.03, 0.02, 0.01, 0.007, 0.005, 0.003, 0.001]$, and the MSE evaluated for each state. Each data point is for the step-size with the lowest MSE for that error ratio and state. While the average estimate is closer to the true values for VTD, the variance of the estimates is much larger. Further, the average estimates for VTD are either unchanged or move negative, while those of the direct algorithm tend toward positive bias.

For Figure 9 the MSE is summed over all states. Again, for each error ratio the MSE was compared over the same

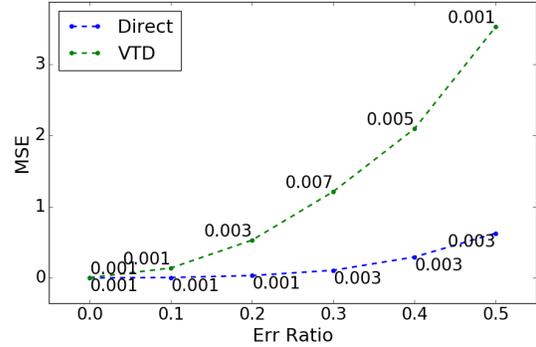


Figure 9: **Random MDP**. The MSE computed for the last 10,000 timesteps of 120 runs summed over all states using the step-size with the lowest overall MSE at each error ratio. For each point the step-size used ($\alpha = \bar{\alpha}$) is displayed.

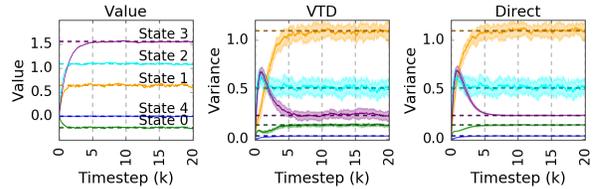


Figure 10: **Random MDP**. Using traces (TD(λ), $\alpha = \bar{\alpha} = 0.01$). Traces only used in value estimator ($\kappa = 1.0, \bar{\kappa} = 0.0$). Notice the slight increase in the variance of the VTD estimates for State 0 and 3.

step-sizes as before and, for each point, the smallest MSE is plotted. These results suggest the direct algorithm is less affected by error in J .

4.4 USING TRACES

We briefly look at the behavior of the random MDP when traces are used. We found no difference when traces are only used in the secondary estimator and not in the value estimator ($\kappa = 0.0, \bar{\kappa} = 1.0$. See Appendix A, Figure 14). Figure 10 considers the opposite scenario, where traces are only used in the value estimator ($\kappa = 1.0, \bar{\kappa} = 0.0$). Here we do see a difference. Particularly the VTD method shows more variance in its estimates for State 0 and 3.

4.5 OFF-POLICY LEARNING

We evaluate two different off-policy scenarios on the random MDP. First, we estimate V under the target policy from off-policy samples. That is, we estimate the V that would be observed if we followed the target policy, i.e., $\eta = 1, \bar{\rho} = \rho$. Both methods achieved similar results in

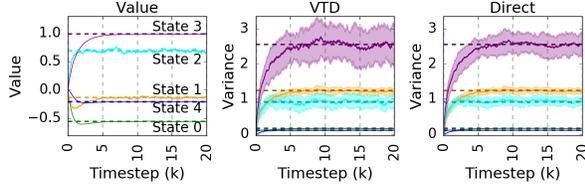


Figure 11: **Random MDP** estimating the variance of the off-policy return ($\alpha = \bar{\alpha} = 0.01, \bar{\rho} = 1, \eta = \rho$).

this setting (Figure 15). In the second off-policy setting, we estimate the variance of the off-policy return (Equation 14). Here $\bar{\rho} = 1$ and $\eta = \rho$. Figure 11 shows that both algorithms successfully estimate the return in this setting. However, despite having the same step-size as the value estimator, VTD produces higher variance in its estimates, as is most clearly seen in State 3.

4.6 FUNCTION APPROXIMATION

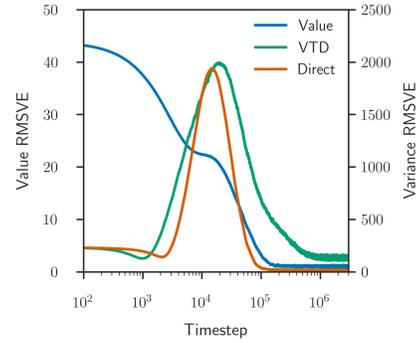
While this paper has focused on the tabular case, where each state is represented uniquely, here we include a first empirical result in the function approximation setting. We evaluate both methods on the random walk shown in Figure 12(a). This domain was previously used by Tamar et al. (2016) for indirectly estimating the variance of the return with LSTD(λ). We use transition based γ (White, 2017) to remove the terminal state and translate the task into a continuing task. Further, we alter the state representation to make it more amenable to TD(λ). For a state s_i we used $\phi_J(i) = [1, (i + 1)/30]^T$ as features for the value learner and $\phi_M(i) = \phi_V(i) = [1, (i + 1)/30, (i + 1)^2/30^2]^T$ as features for the secondary learner. We set $\kappa = \bar{\kappa} = 0.95$ and performed sweeps over step-sizes of $\{2^i, i \in \{-15, -12, \dots, -1, 0\}\}$. We first found the best step-size for the value learner and then found the best step-size for VTD. Using the same step-size for VTD and DVTD, we obtain the results shown in Figure 12(b). Here we see DVTD drastically outperforms VTD. Further details are available in Appendix C.

5 DISCUSSION

Both DVTD and VTD effectively estimate the variance across a range of settings, but DVTD is simpler and more robust. This simplicity alone makes DVTD preferable. The higher variance in estimates produced by VTD is likely due to the larger target which VTD uses in its learning updates: $\mathbb{E}[X^2] \geq \mathbb{E}[(X - \mathbb{E}[X])^2]$; we show more explicitly how this affects the updates of VTD in Appendix E. We expect the differences between the two approaches to be most pronounced for domains with larger returns than those demonstrated here. Consider



(a)



(b)

Figure 12: **Random Walk.** **a)** Random walk with rewards of -1 for every transition to a non-terminal state. Note that there is no discounting in this domain. **b)** Results under linear function approximation averaged over 100 runs. Shading indicates standard error (negligible).

the task of a helicopter hovering formalized as a reinforcement learning task (Kim et al., 2004). In the most well-known variants of this problem the agent receives a massive negative reward for crashing the helicopter (e.g., minus one million). In such problems the magnitude and variance of the return is large. Here, estimating the second moment may not be feasible from a statistical point of view, whereas the target of our direct variance estimate should be better behaved. By focusing on simple MDPs we were able to carefully evaluate the properties of these algorithms while keeping them isolated from additional effects like state-aliasing due to function approximation. Further studies in more complex settings, such as function approximation, are left to future work.

Acknowledgements

Funding was provided by the Natural Sciences and Engineering Research Council of Canada, Alberta Innovates, and Google DeepMind. Thanks to Hossein Aboutaleb who notified us of an error in the original proof of Theorem 2.

References

- Bellemare, M. G., Dabney, W., & Munos, R. (2017). A distributional perspective on reinforcement learning. In *International Conference on Machine Learning (ICML)*. Sydney, Australia.
- Dimitrakakis, C. (2006). *Ensembles for sequence learning* (Doctoral dissertation, EPFL).
- Kim, H. J., Jordan, M. I., Sastry, S., & Ng, A. Y. (2004). Autonomous helicopter flight via reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 799–806).
- Maei, H. R. (2011). *Gradient temporal-difference learning algorithms* (Doctoral dissertation, University of Alberta).
- Prashanth, L. & Ghavamzadeh, M. (2013). Actor-critic algorithms for risk-sensitive mdps. In *Advances in Neural Information Processing Systems (NIPS)* (pp. 252–260).
- Sakaguchi, Y. & Takano, M. (2004). Reliability of internal prediction/estimation and its application. I. Adaptive action selection reflecting reliability of value function. *Neural Networks*, 17(7), 935–952.
- Sato, M., Kimura, H., & Kobayashi, S. (2001). TD algorithm for the variance of return and mean-variance reinforcement learning. *Transactions of the Japanese Society for Artificial Intelligence*, 16(3), 353–362.
- Sobel, M. J. (1982). The variance of discounted markov decision processes. *Journal of Applied Probability*, 19(4), 794–802.
- Sutton, R. S. & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.
- Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C., & Wiewiora, E. (2009). Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *International Conference on Machine Learning (ICML)* (pp. 993–1000).
- Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., & Precup, D. (2011). Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (Vol. 2, pp. 761–768).
- Tamar, A., Di Castro, D., & Mannor, S. (2012). Policy gradients with variance related risk criteria. In *International Conference on Machine Learning (ICML)* (pp. 387–396).
- Tamar, A., Di Castro, D., & Mannor, S. (2016). Learning the variance of the reward-to-go. *Journal of Machine Learning Research*, 17(13), 1–36.
- Tamar, A. & Mannor, S. (2013). Variance adjusted actor critic algorithms. arXiv: 1310.3697
- White, M. (2017). Unifying task specification in reinforcement learning. In *International Conference on Machine Learning (ICML)* (pp. 3742–3750).
- White, M. & White, A. (2016). A greedy approach to adapting the trace parameter for temporal difference learning. In *International Conference on Autonomous Agents & Multiagent Systems (AAMAS)* (pp. 557–565).
- Yu, H. (2015). On convergence of emphatic temporal-difference learning. In *Conference on Learning Theory (COLT)* (pp. 1724–1751).
- Zeiler, M. D. (2012). ADADELTA: An adaptive learning rate method. arXiv: 1212.5701