A Greedy Approach to Adapting the Trace Parameter for Temporal Difference Learning

Martha White
Department of Computer Science
Indiana University
Bloomington, IN 47405, USA
martha@indiana.edu

Adam White
Department of Computer Science
Indiana University
Bloomington, IN 47405, USA
adamw@indiana.edu

ABSTRACT

One of the main obstacles to broad application of reinforcement learning methods is the parameter sensitivity of our core learning algorithms. In many large-scale applications, online computation and function approximation represent key strategies in scaling up reinforcement learning algorithms. In this setting, we have effective and reasonably well understood algorithms for adapting the learning-rate parameter, online during learning. Such meta-learning approaches can improve robustness of learning and enable specialization to current task, improving learning speed. For temporal-difference learning algorithms which we study here, there is yet another parameter, λ, that similarly impacts learning speed and stability in practice. Unfortunately, unlike the learning-rate parameter, λ parametrizes the objective function that temporal-difference methods optimize. Different choices of λ produce different fixed-point solutions, and thus adapting λ online and characterizing the optimization is substantially more complex than adapting the learning-rate parameter. There are no meta-learning method for λ that can achieve (1) incremental updating, (2) compatibility with function approximation, and (3) maintain stability of learning under both on and off-policy sampling. In this paper we contribute a novel objective function for optimizing λ as a function of state rather than time. We derive a new incremental, linear complexity λ-adaption algorithm that does not require offline batch updating or access to a model of the world, and present a suite of experiments illustrating the practicality of our new algorithm in three different settings. Taken together, our contributions represent a concrete step towards black-box application of temporal-difference learning methods in real world problems.

Keywords

Reinforcement learning; temporal difference learning; off-policy learning

1. INTRODUCTION

In reinforcement learning, the training data is produced by an adaptive learning agent's interaction with its environment, which makes tuning the parameters of the learning process both challenging and essential for good performance. In the online setting we study here, the agent-environment interaction produces an unending stream of temporally correlated data. In this setting there is no testing-training split, and thus the agent's learning process must be robust and adapt to new situations not considered by the human designer. Robustness is often critically related to the values of a small set parameters that control the learning process (e.g., the step-size parameter). In real-world applications, however, we cannot expect to test a large range of theses parameter values, in all the situations the agent may face, to ensure good performance—common practice in empirical studies. Unfortunately, safe values of these parameters are usually problem dependent. For example, in off-policy learning (e.g., learning from demonstrations), large importance sampling ratios can destabilize provably convergent gradient temporal difference learning methods, when the parameters are not set in a very particular way ($\lambda = 0$) [33]. In such situations, we turn to meta-learning algorithms that can adapt the parameters of the agent continuously, based on the stream of experience and some notion of the agent's own learning progress. These meta-learning approaches can potentially improve robustness, and also help the agent specialize to the current task, and thus improve learning speed.

Temporal difference learning methods make use of two important parameters: the step-size parameter and the trace-decay parameter. The step-size parameter is the same as those used in stochastic gradient descent, and there are algorithms available for adjusting this parameter online, in reinforcement learning [2]. For the trace decay parameter, on the other hand, we have no generally applicable meta-learning algorithms that are compatible with function approximation, incremental processing, and off-policy sampling.

The difficulty in adapting the trace decay parameter, λ, mainly arises from the fact that it has seemingly multiple roles and also influences the fixed-point solution. This parameter was introduced in Samuel's checker player [16], and later described as interpolation parameter between offline TD(0) and Monte-Carlo sampling (TD($\lambda = 1$) by Sutton[21]. It has been empirically demonstrated that values of λ between zero and one often perform the best in practice [21, 23, 32]. This trace parameter can also be viewed as a bias-variance trade-off parameter: λ closer to one is less biased but likely to have higher variance, where λ closer to zero is more biased, but likely has lower variance. However, it has also been described as a credit-assignment parameter [19], as a method to encode probability of transitions [25], a way to incorporate the agent's confidence in its value func-

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), J. Thangarajah, K. Tuyls, C. Jonker, S. Marsella (eds.), May 9–13, 2016, Singapore.*

tion estimates [23, 29], and as an averaging of n-step returns [23]. Selecting λ is further complicated by the fact that λ is a part of the problem definition: the solution to the Bellman fixed point equation is dependent on the choice of λ (unlike the step-size parameter).

There are few approaches for setting λ, and most existing work is limited to special cases. For instance, several approaches have analyzed setting λ for variants of TD that were introduced to simplify the analysis, including phased TD [5] and TD$^*(\lambda)$ [17]. Though both provide valuable insights into the role of λ, the analysis does not easily extend to conventional TD algorithms. Sutton and Singh [25] investigated tuning both the learning rate parameter and λ, and proposed two meta-learning algorithms. The first assumes the problem can be modeled by an acyclic MDP, and the other requires access to the transition model of the MDP. Singh and Dayan [18] and Kearns and Singh [5] contributed extensive simulation studies of the interaction between λ and other agent parameters on a chain MDP, but again relied on access to the model and offline computation. The most recent study [3] explores a Bayesian variant of TD learning, but requires a batch of samples and can only be used off-line. Finally, Konidaris et al. [6] introduce TD$_\gamma$ as a method to remove the λ parameter altogether. Their approach, however, has not been extended to the off-policy setting and their full algorithm is too computationally expensive for incremental estimation, while their incremental variant introduces a sensitive meta-parameter. Although this long-history of prior work has helped develop our intuitions about λ, the available solutions are still far from the use cases outlined above.

This paper introduces an new objective based on locally optimizing bias-variance, which we use to develop an efficient, incremental algorithm for learning state-based λ. We use a forward-backward analysis [23] to derive an incremental algorithm to estimate the variance of the return. Using this estimate, we obtain a closed-form estimate of λ on each time-step. Finally, we empirically demonstrate the generality of the approach with a suite of on-policy and off-policy experiments. Our results show that our new algorithm, λ-greedy, is consistently amongst the best performing, adapting as the problem changes, whereas any fixed approach works well in some settings and poorly in anothers.

2. BACKGROUND

We model the agent's interaction with an unknown environment as a discrete time Markov Decision Process (MDP). A MDP is characterized by a finite set of states $\mathcal{S}$, set of actions $\mathcal{A}$, a reward function $r : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$, and generalized state-based discount $\gamma : \mathcal{S} \in [0,1]$, which encodes the level of discounting per-state (e.g., a common setting is a constant discount for all states). On each of a discrete number of timesteps, $t = 1, 2, 3, \ldots$, the agent observes the current state S_t, selects an action A_t, according to its target policy $\pi : \mathcal{S} \times \mathcal{A} \to [0,1]$, and the environment transitions to a new state S_{t+1} and emits a reward R_{t+1}. The state transitions are governed by the transition function $P : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to [0,1]$, where $P(S_t, A_t, S_{t+1})$ denotes the probability of transitioning from S_t to S_{t+1}, due to action A_t. At timestep t, the future rewards are summarized by the Monte Carlo (MC) return $G_t \in \mathbb{R}$ defined by the infinite discounted sum

$$\begin{aligned} G_t &\overset{\text{def}}{=} R_{t+1} + \gamma_{t+1}R_{t+2} + \gamma_{t+1}\gamma_{t+2}R_{t+3} + \ldots \qquad \triangleright\, \gamma_t = \gamma(S_t) \\ &= R_{t+1} + \gamma_{t+1}G_{t+1}. \end{aligned}$$

The agent's objective is to estimate the expected return or value function, $v^\pi : \mathcal{S} \to \mathbb{R}$, defined as $v^\pi(s) \overset{\text{def}}{=} \mathbb{E}[G_t | S_t = s, A_t \sim \pi]$. We estimate the value function using the standard framework of linear function approximation. We assume the state of the environment at time t can be characterized by a fixed-length feature vector $\mathbf{x}_t \in \mathbb{R}^n$, where $n \ll |\mathcal{S}|$; implicitly, $\mathbf{x}_t$ is a function of the random variable S_t. The agent uses a linear estimate of the value of S_t: the inner product of $\mathbf{x}_t$ and a modifiable set of weights $\mathbf{w} \in \mathbb{R}^n$, $\hat{v}(S_t, \mathbf{w}) \overset{\text{def}}{=} \mathbf{x}_t^\top \mathbf{w}$, with mean-squared error (MSE) $= \sum_{s\in\mathcal{S}} d(s)(v(s) - \hat{v}(s, \mathbf{w}))^2$, where $d : \mathcal{S} \to [0,1]$ encodes the distribution over states induced by the agent's behavior in the MDP.

Instead of estimating the expected value of G_t, we can estimate a λ-return that is expected to have lower variance

$$G_t^\lambda \overset{\text{def}}{=} R_{t+1} + \gamma_{t+1}[(1 - \lambda_{t+1})\mathbf{x}_{t+1}^\top \mathbf{w} + \lambda_{t+1}G_{t+1}^\lambda],$$

where the *trace decay function* $\lambda : \mathcal{S} \to [0,1]$ specifies the trace parameter as a function of state. The trace parameter $\lambda_{t+1} = \lambda(s_{t+1})$ averages the estimate of the return, $\mathbf{x}_{t+1}^\top \mathbf{w}$, and the λ-return starting on the next step, G_{t+1}^λ. When $\lambda = 1$, G_t^λ becomes the MC return G_t, and the value function can be estimated by averaging rollouts from each state. When $\lambda = 0$, G_t^λ becomes equal to the *one-step λ-return*, $R_{t+1} + \gamma_{t+1}\mathbf{x}_{t+1}^\top \mathbf{w}$, and the value function can be estimated by the linear TD(0) algorithm. The λ-return when $\lambda \in (0, 1)$ is often easier to estimate than MC, and yields more accurate predictions than using the one-step return. The intuition, is that the for large λ, the estimate is high-variance due to averaging possibly long trajectories of noisy rewards, but less bias because the initial biased estimates of the value function participate less in the computation of the return. In the case of low λ, the estimate has lower-variance because fewer potentially noisy rewards participate in G_t^λ, but there is more bias due to the increase role of the initial value function estimates. We further discuss the intuition for this parameter in the next section.

The generalization to state-based γ and λ have not yet been widely considered, though the concept was introduced more than a decade ago [22, 27] and the generalization shown to be useful [22, 10, 14, 26]. The Bellman operator can be generalized to include state-based γ and λ (see [26, Equation 29]), where the choice of λ per-state influences the fixed point. Time-based λ, on the other hand, would not result in a well-defined fixed point. Therefore, to ensure a well-defined fixed point, we will design an objective and algorithm to learn a state-based λ.

This paper considers both on- and off-policy policy evaluation. In the more conventional on-policy learning setting, we estimate $v^\pi(s)$ based on samples generated while selecting actions according to the target policy π. In the off-policy case, we estimate $v^\pi(s)$ based on samples generated while selecting actions according to the behavior policy $\mu : \mathcal{S} \times \mathcal{A} \to [0,1]$, and $\pi \neq \mu$. In order to learn $\hat{v}$ in both these settings we use the GTD(λ) algorithm [9] specified by the following update equations:

$$\rho_t \overset{\text{def}}{=} \frac{\pi(S_t, A_t)}{\mu(S_t, A_t)} \qquad \triangleright \text{ importance sampling ratio}$$

$$\mathbf{e}_t \overset{\text{def}}{=} \rho_t(\gamma_t\lambda_t\mathbf{e}_{t-1} + \mathbf{x}_t) \qquad \triangleright \text{ eligibility trace}$$

$$\delta_t \overset{\text{def}}{=} R_{t+1} + \gamma_t\mathbf{x}_{t+1}^\top\mathbf{w}_t - \mathbf{x}_t^\top\mathbf{w}_t \qquad \triangleright \text{ TD error}$$

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \alpha(\delta_t \mathbf{e}_t - \gamma_t(1-\lambda_{t+1})\mathbf{e}_t^\top \mathbf{h}_t \mathbf{x}_{t+1})$$
$$\mathbf{h}_{t+1} \leftarrow \mathbf{h}_t + \alpha_{\mathbf{h}}(\delta_t \mathbf{e}_t - \mathbf{x}_t^\top \mathbf{h}_t)\mathbf{x}_t \qquad \triangleright \text{auxiliary weights}$$

with step-sizes $\alpha, \alpha_{\mathbf{h}} \in \mathbb{R}^+$ and an arbitrary initial $\mathbf{w}_0, \mathbf{h}_0$ (e.g., the zero vector). The importance sampling ratio $\rho_t \in \mathbb{R}^+$ facilitates learning about rewards as if they were generated by following π, instead of μ. This ratio can be very large if $\mu(S_t, A_t)$ is small, which can compound and destabilize learning.

3. OBJECTIVE FOR TRACE ADAPTATION

To obtain an objective for selecting λ, we need to clarify its role. Although λ was not introduced with the goal of trading off bias and variance [21], several algorithms and significant theory have developed its role as such [5, 17]. Other roles have been suggested; however, as we discuss below, each of them can still be thought of as a bias-variance trade-off.

The λ parameter has been described as a credit assignment parameter, which allows TD(λ) to perform multi-step updates on each time step. On each update, λ_t controls the amount of credit assigned to previous transitions, using the eligibility trace $\mathbf{e}$. For λ_t close to 1, TD(λ) assigns more credit for the current reward to previous transitions, resulting in updates to many states along the current trajectory. Conversely, for $\lambda_t = 0$, the eligibility trace is cleared and no credit is assigned back in time, performing a single-step TD(0) update. In fact, this intuition can still be thought of as a bias-variance trade-off. In terms of credit assignment, we ideally always want to send maximal credit $\lambda = 1$, but decayed by γ, for the current reward, which is also unbiased. In practice, however, this often leads to high variance, and thus we mitigate the variance by choosing λ less than one and speed learning overall, but introduce bias.

Another interpretation is that λ should be set to reflect confidence in value function estimates [29, 23]. If your confidence in the value estimate of state s is high, then $\lambda(s)$ should be close to 0, meaning we trust the estimates provided by $\hat{v}$. If your confidence is low, suspecting that $\hat{v}$ may be inaccurate, then $\lambda(s)$ should be close to 1, meaning we trust observed rewards more. For example in states that are indistinguishable with function approximation (i.e., aliased states), we should not trust the $\hat{v}$ as much. This intuition similarly translates to bias-variance. If $\hat{v}$ is accurate, then decreasing $\lambda(s)$ does not incur (much) bias, but can significantly decrease the variance since $\hat{v}$ gives the correct value. If $\hat{v}$ is inaccurate, then the increased bias is not worth the reduced variance, so $\lambda(s)$ should be closer to 1 to use actual (potentially high-variance) samples.

Finally, a less commonly discussed interpretation is that λ acts as parameter that simulates a form of experience replay (or model-based simulation of trajectories). One can imagine that sending back information in eligibility traces is like simulating experience from a model, where the model could be a set of trajectories, as in experience replay [8]. If $\lambda = 1$, the traces are longer and each update gets more trajectory information, or experience replay. If a trajectory from a point, however, was unlikely (e.g., a rare transition), we may not want to use that information. Such an approach was taken by Sutton and Singh [25], where λ was set to the transition probabilities. Even in this model-based interpretation, the goal in setting λ becomes one of mitigating variance, without incurring too much bias.

Optimizing this bias-variance trade-off, however, is difficult because λ affects the return we are approximating. Jointly optimizing for λ across all time-steps is generally not feasible. One strategy is to take a batch approach, where the optimal λ is determined after seeing all the data [3]. Our goal, however, is to develop approaches for the online setting, where future states, actions, rewards and the influence of λ have yet to be observed.

We propose to take a greedy approach: on each time step select λ_{t+1} to optimize the bias-variance trade-off *for only this step*. This greedy objective corresponds to minimizing the mean-squared error between the unbiased $\lambda = 1$ return G_t and the estimate $\hat{G}_t$ with $\lambda_{t+1} \in [0,1]$ with $\lambda = 1$ into the future after $t+1$

$$\hat{G}_t \overset{\text{def}}{=} \mathbb{E}[\ \rho_t(R_{t+1} + \gamma_{t+1}[(1-\lambda_{t+1})\mathbf{x}_{t+1}^\top \mathbf{w}_t + \lambda_{t+1} \underbrace{G_{t+1}}_{\text{Monte Carlo}}])\ ].$$

Notice that $\hat{G}_t$ interpolates between the current value estimate and the unbiased $\lambda = 1$ MC return, and so is not recursive. Picking $\lambda_{t+1} = 1$ gives an unbiased estimate, since then we would be estimating $\hat{G}_t = G_t$. We greedily decide how λ_{t+1} should be set on this step to locally optimize the mean-squared error (i.e., bias-variance). This greedy decision is made given both $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$, which are both available when choosing λ_{t+1}. To simplify notation in this section, we assume that $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$ are both given in the below expectations.

To minimize the mean-squared error in terms of λ_{t+1}

$$\text{MSE}(\lambda_{t+1}) \overset{\text{def}}{=} \mathbb{E}[(G_t - \hat{G}_t)^2]$$

we will consider the two terms that compose the mean-squared error: the squared bias term and the variance term.

$$\text{Bias}(\lambda_{t+1}) \overset{\text{def}}{=} \mathbb{E}[G_t] - \mathbb{E}[\hat{G}_t]$$
$$\text{Variance}(\lambda_{t+1}) \overset{\text{def}}{=} \text{Var}[\hat{G}_t]$$
$$\text{MSE}(\lambda_{t+1}) = \text{Bias}(\lambda_{t+1})^2 + \text{Variance}(\lambda_{t+1}).$$

Let us begin by rewriting the bias. Since we are given $\mathbf{x}_t$, ρ_t, $\mathbf{x}_{t+1}$ and γ_{t+1} when choosing λ_{t+1},

$$\begin{aligned}\mathbb{E}[\hat{G}_t] &= \rho_t \mathbb{E}\left[R_{t+1} + \gamma_{t+1}(1-\lambda_{t+1})\mathbf{x}_{t+1}^\top \mathbf{w}_t + \lambda_{t+1} G_{t+1}\right] \\ &= \rho_t \mathbb{E}[R_{t+1}] + \rho_t \gamma_{t+1}\left((1-\lambda_{t+1})\mathbf{x}_{t+1}^\top \mathbf{w}_t + \lambda_{t+1}\mathbb{E}[G_{t+1}]\right)\end{aligned}$$

For convenience, define

$$\text{err}(\mathbf{w}, \mathbf{x}_{t+1}) \overset{\text{def}}{=} \mathbb{E}[G_{t+1}] - \mathbf{x}_{t+1}^\top \mathbf{w} \qquad (1)$$

as the difference between the $\lambda = 1$ return and the current approximate value from state $\mathbf{x}_{t+1}$ using weights $\mathbf{w}_t$. Using this definition, we can rewrite

$$\begin{aligned}&(1-\lambda_{t+1})\mathbf{x}_{t+1}^\top \mathbf{w}_t + \lambda_{t+1}\mathbb{E}[G_{t+1}] \\ &= (1-\lambda_{t+1})(\mathbb{E}[G_{t+1}] - \text{err}(\mathbf{w}_t, \mathbf{x}_{t+1})) + \lambda_{t+1}\mathbb{E}[G_{t+1}] \\ &= \mathbb{E}[G_{t+1}] - (1-\lambda_{t+1})\text{err}(\mathbf{w}_t, \mathbf{x}_{t+1})\end{aligned}$$

giving

$$\begin{aligned}\mathbb{E}[\hat{G}_t] &= \rho_t\left(\mathbb{E}[R_{t+1}] + \gamma_{t+1}\mathbb{E}[G_{t+1}]\right) \\ &\qquad - \rho_t \gamma_{t+1}(1-\lambda_{t+1})\text{err}(\mathbf{w}_t, \mathbf{x}_{t+1}) \\ &= \mathbb{E}[G_t] - \rho_t \gamma_{t+1}(1-\lambda_{t+1})\text{err}(\mathbf{w}_t, \mathbf{x}_{t+1})\end{aligned}$$

$$\implies \text{Bias}^2(\lambda_{t+1}) = \left(\mathbb{E}[G_t] - \mathbb{E}[\hat{G}_t]\right)^2$$
$$= \rho_t^2\gamma_{t+1}^2(1-\lambda_{t+1})^2\text{err}^2(\mathbf{w}_t, \mathbf{x}_{t+1}).$$

For the variance term, we will assume that the noise in the reward R_{t+1} given $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$ is independent of the other dynamics [12], with variance $\sigma_r(\mathbf{x}_t, \mathbf{x}_{t+1})$. Again since we are given $\mathbf{x}_t$, ρ_t, $\mathbf{x}_{t+1}$ and γ_{t+1}

$$\text{Var}[\hat{G}_t]$$
$$= \rho_t^2\text{Var}[R_{t+1} + \underbrace{\gamma_{t+1}(1-\lambda_{t+1})\mathbf{x}_{t+1}^\top\mathbf{w}_t}_{\text{constant given } \rho_t, \mathbf{x}_{t+1}, \gamma_{t+1}} + \gamma_{t+1}\lambda_{t+1}G_{t+1}]$$
$$= \rho_t^2 \underbrace{\text{Var}[R_{t+1}] + \rho_t^2\gamma_{t+1}^2\lambda_{t+1}^2\text{Var}[G_{t+1}]}_{\text{independent given } \mathbf{x}_t, \mathbf{x}_{t+1}}$$
$$= \rho_t^2\sigma_r(\mathbf{x}_t, \mathbf{x}_{t+1}) + \rho_t^2\gamma_{t+1}^2\lambda_{t+1}^2\text{Var}[G_{t+1}]$$
$$\implies \text{Variance}(\lambda_{t+1}) = \rho_t^2\gamma_{t+1}^2\lambda_{t+1}^2\text{Var}[G_{t+1}] + \rho_t^2\sigma_r(\mathbf{x}_t, \mathbf{x}_{t+1})$$

Finally, we can drop the constant $\rho_t^2\sigma_r(\mathbf{x}_t, \mathbf{x}_{t+1})$ in the objective, and drop the $\rho_t^2\gamma_{t+1}^2$ in both the bias and variance terms as it only scales the objective, giving the optimization

$$\min_{\lambda_{t+1}\in[0,1]} \text{Bias}^2(\lambda_{t+1}) + \text{Variance}(\lambda_{t+1})$$
$$\equiv \min_{\lambda_{t+1}\in[0,1]} (1-\lambda_{t+1})^2\text{err}_{t+1}^2(\mathbf{w}_t) + \lambda_{t+1}^2\text{Var}[G_{t+1}].$$

We can take the gradient of this optimization to find a closed form solution

$$-2(1-\lambda_{t+1})\text{err}_{t+1}^2(\mathbf{w}_t) + 2\lambda_{t+1}\text{Var}[G_{t+1}] = 0$$
$$\implies \left(\text{Var}[G_{t+1}] + \text{err}_{t+1}^2(\mathbf{w}_t)\right)\lambda_{t+1} - \text{err}_{t+1}^2(\mathbf{w}_t) = 0$$
$$\implies \lambda_{t+1} = \frac{\text{err}_{t+1}^2(\mathbf{w}_t)}{\text{Var}[G_{t+1}] + \text{err}_{t+1}^2(\mathbf{w}_t)} \quad (2)$$

which is always feasible, unless both the variance and error are zero (in which case, any choice of λ_{t+1} is equivalent). Though the importance sampling ratio ρ_t does not affect the choice of λ on the current time step, it can have a dramatic effect on $\text{Var}[G_{t+1}]$ into the future via the eligibility trace. For example, when the target and behavior policy are strongly mis-matched, ρ_t can be large, which multiplies into the eligibility trace $\mathbf{e}_t$. If several steps have large ρ_t, then $\mathbf{e}_t$ can get very large. In this case, the equation in (2) would select a small λ_{t+1}, significantly decreasing variance.

4. TRACE ADAPTATION ALGORITHM

To approximate the solution to our proposed optimization, we need a way to approximate the error and the variance terms in equation (2). To estimate the error, we need an estimate of the expected return from each state, $\mathbb{E}[G_t]$. To estimate the variance, we need to obtain an estimate of $\mathbb{E}[G_t^2]$, and then can use $\text{Var}[G_t] = \mathbb{E}[G_t^2] - \mathbb{E}[G_t]^2$. The estimation of the expected return is in fact the problem tackled by this paper, and one could use a TD algorithm, learning weight vector $\mathbf{w}^{\text{err}}$ to obtain approximation $\mathbf{x}_t^\top\mathbf{w}^{\text{err}}$ to $\mathbb{E}[G_t]$. This approach may seem problematic, as this sub-step appears to be solving the same problem we originally aimed to solve. However, as in many meta-parameter optimization approaches, this approximation can be inaccurate and still adequately guide selection of λ. We discuss this further in the experimental results section.

Similarly, we would like to estimate $\mathbb{E}[G_t^2]$ with $\mathbf{w}^{\text{sq}}\mathbf{x}_t^\top$ by learning $\mathbf{w}^{\text{sq}}$; estimating the variance or the second moment of the return, however, has not been extensively studied. Sobel [20] provides a Bellman equation for the variance of the λ-return, when $\lambda = 0$. There is also an extensive literature on risk-averse MDP learning, where the variance of the return is often used as a measure [15, 13, 1, 28]; however, an explicit way to estimate the variance of the return for $\lambda > 0$ is not given. There has also been some work on estimating the variance of the *value function* [12, 35], for general λ; though related, this is different than estimating the variance of the λ-*return*.

In the next section, we provide a derivation for a new algorithm called variance temporal difference learning (VTD), to approximate the second moment of the return for any state-based λ. The general VTD updates are given at the end of Section 5.2. For λ-greedy, we use VTD to estimate the variance, with the complete algorithm summarized in Algorithm 1. We opt for simple meta-parameter settings, so that no additional parameters are introduced. We use the same step-size α that is used for the main weights to update $\mathbf{w}^{\text{err}}$ and $\mathbf{w}^{\text{sq}}$. In addition, we set the weights $\mathbf{w}^{\text{err}}$ and $\mathbf{w}^{\text{sq}}$ to reflect *a priori* estimates of error and variance. As a reasonable rule-of-thumb, $\mathbf{w}^{\text{err}}$ should be set larger than $\mathbf{w}^{\text{sq}}$, to reflect that initial value estimates are inaccurate. This results in

Algorithm 1: Policy evaluation with λ-greedy

```
$\mathbf{w}_t \leftarrow \mathbf{0}$ // Main weights
$\mathbf{h}_t \leftarrow \mathbf{0}$ // Auxiliary weights for off-policy learning
$\mathbf{e}_t \leftarrow \mathbf{0}$ // Main trace parameters
$\mathbf{x}_t \leftarrow$ the initial observation
$\mathbf{w}^{\text{err}} \leftarrow \frac{\text{Rmax}}{1-\gamma} \times \mathbf{1}, \mathbf{w}^{\text{sq}} \leftarrow 0.0 \times \mathbf{1}$ // Aux. weights for $\lambda$
$\bar{\mathbf{e}}_t \leftarrow \mathbf{0}, \bar{\mathbf{z}}_t \leftarrow \mathbf{0}$ // Aux. traces
repeat
  Take action accord. to $\pi$, observe $\mathbf{x}_{t+1}$, reward $r_{t+1}$
  $\rho_t = \pi(s_t, a_t)/\mu(s_t, a_t)$    // In on-policy, $\rho_t = 1$
  $\lambda_{t+1} \leftarrow \lambda$-greedy$(\mathbf{w}^{\text{err}}, \mathbf{w}^{\text{sq}}, \mathbf{w}_t, \mathbf{x}_t, \mathbf{x}_{t+1}, r_{t+1}, \rho_t)$
  // Now can use any algorithm, e.g., GTD
  $\delta_t = r_{t+1} + \gamma_{t+1}w_t^\top\mathbf{x}_{t+1} - \mathbf{w}_t^\top\mathbf{x}_t$
  $\mathbf{e}_t = \rho_t(\gamma_t\lambda_t\mathbf{e}_{t-1} + \mathbf{x}_t)$
  $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha(\delta_t\mathbf{e}_t - \gamma_{t+1}(1-\lambda_{t+1})\mathbf{x}_{t+1})(\mathbf{e}_t^\top\mathbf{h}_t))$
  $\mathbf{h}_{t+1} = \mathbf{h}_t + \alpha_{\mathbf{h}}(\delta_t\mathbf{e}_t - (\mathbf{h}_t^\top\mathbf{x}_t)\mathbf{x}_t)$
until agent done interaction with environment
```

Algorithm 2: λ-greedy$(\mathbf{w}^{\text{err}}, \mathbf{w}^{\text{sq}}, \mathbf{w}_t, \mathbf{x}_t, \mathbf{x}_{t+1}, r_{t+1}, \rho_t)$

```
// Use GTD to update $\mathbf{w}^{\text{err}}$
$\bar{g}_{t+1} \leftarrow \mathbf{x}_{t+1}^\top\mathbf{w}^{\text{err}}$
$\delta_t \leftarrow r_{t+1} + \gamma_{t+1}\bar{g}_{t+1} - \mathbf{x}_t^\top\mathbf{w}^{\text{err}}$
$\bar{\mathbf{e}}_t = \rho_t(\gamma_t\bar{\mathbf{e}}_{t-1} + \mathbf{x}_t)$
$\mathbf{w}^{\text{err}} = \mathbf{w}^{\text{err}} + \alpha\delta_t\bar{\mathbf{e}}_t$
// Use VTD to update $\mathbf{w}^{\text{sq}}$
$\bar{r}_{t+1} \leftarrow \rho_t^2 r_{t+1}^2 + \rho_t^2\gamma_{t+1}r_{t+1}\bar{g}_{t+1}$
$\bar{\gamma}_{t+1} \leftarrow \rho_t^2\gamma_{t+1}^2$
$\bar{\delta}_t \leftarrow \bar{r}_{t+1} + \bar{\gamma}_{t+1}\mathbf{x}_{t+1}^\top\mathbf{w}^{\text{sq}} - \mathbf{x}_t^\top\mathbf{w}^{\text{sq}}$
$\bar{\mathbf{z}}_t = \bar{\gamma}_t\bar{\mathbf{z}}_{t-1} + \mathbf{x}_t$
$\mathbf{w}^{\text{sq}} = \mathbf{w}^{\text{sq}} + \alpha\bar{\delta}_t\bar{\mathbf{z}}_t$
// Compute $\lambda$ estimate
errsq $= (\bar{g}_{t+1} - \mathbf{x}_{t+1}^\top\mathbf{w}_t)^2$
varg $= \max(0, \mathbf{x}_{t+1}^\top\mathbf{w}^{\text{sq}} - (\bar{g}_{t+1})^2)$
$\lambda_{t+1} = $ errsq/(varg + errsq)
return $\lambda_{t+1}$
```

an estimate of variance $\text{Var}[G_{t+1}] \approx \mathbf{x}_{t+1}^\top \mathbf{w}^{\text{sq}} - (\mathbf{x}_{t+1}^\top \mathbf{w}^{\text{err}})^2$ that is capped at zero until $\mathbf{x}_{t+1}^\top \mathbf{w}^{\text{sq}}$ becomes larger than $(\mathbf{x}_{t+1}^\top \mathbf{w}^{\text{err}})^2$.

5. APPROXIMATING THE SECOND MOMENT OF THE RETURN

In this section, we derive the general VTD algorithm to approximate the second moment of the λ-return. Though we will set $\lambda_{t+1} = 1$ in our algorithm, we nonetheless provide the more general algorithm as the only model-free variance estimation approach for general λ-returns.

The key novelty is in determining a Bellman operator for the squared return, which then defines a fixed-point objective, called the Var-MSPBE. With this Bellman operator and recursive form for the squared return, we derive a gradient TD algorithm, called VTD, for estimating the second moment. To avoid confusion with parameters for the main algorithm, as a general rule throughout the document, the additional parameters used to estimate the second moment have a bar. For example, γ_{t+1} is the discount for the main problem, and $\bar{\gamma}_{t+1}$ is the discount for the second moment.

5.1 Bellman operator for squared return

The recursive form for the squared-return is

$$\begin{aligned}(G_t^\lambda)^2 &= \rho_t^2(\bar{G}_t^2 + 2\gamma_{t+1}\lambda_{t+1}\bar{G}_t G_t^\lambda + \gamma_{t+1}^2\lambda_{t+1}^2(G_{t+1}^\lambda)^2) \\ &= \bar{r}_{t+1} + \bar{\gamma}_{t+1}(G_{t+1}^\lambda)^2\end{aligned}$$

where for a given $\lambda : \mathcal{S} \to [0,1]$ and $\mathbf{w}$,

$$\begin{aligned}\bar{G}_t &\overset{\text{def}}{=} R_{t+1} + \gamma_{t+1}(1-\lambda_{t+1})\mathbf{x}_{t+1}^\top \mathbf{w} \\ \bar{r}_{t+1} &\overset{\text{def}}{=} \rho_t^2 \bar{G}_t^2 + 2\rho_t^2 \gamma_{t+1}\lambda_{t+1}\bar{G}_t G_{t+1}^\lambda \\ \bar{\gamma} &\overset{\text{def}}{=} \rho_t^2 \gamma_{t+1}^2 \lambda_{t+1}^2.\end{aligned}$$

The $\mathbf{w}$ are the weights for the λ-return, and not the weights $\mathbf{w}^{\text{sq}}$ we will learn for approximating the second moment. For further generality, we introduce a meta-parameter $\bar{\lambda}_t$

$$\bar{V}_t^{\bar{\lambda}} \overset{\text{def}}{=} \bar{r}_{t+1} + \bar{\gamma}_{t+1}\left((1-\bar{\lambda}_{t+1})\mathbf{x}_{t+1}^\top \mathbf{w}^{\text{sq}} + \bar{\lambda}_{t+1}\bar{V}_{t+1}^{\bar{\lambda}}\right)$$

to get a $\bar{\lambda}$-squared-return where for $\bar{\lambda}_{t+1} = 1$, $\bar{V}_t^{\bar{\lambda}} = (G_t^\lambda)^2$. This meta-parameter $\bar{\lambda}$ plays the same role for estimating $(G_t^\lambda)^2$ as λ for estimating G_t.

We can define a generalized Bellman operator $\bar{\mathbf{T}}$ for the squared-return, using this above recursive form. The goal is to obtain the fixed point $\bar{\mathbf{T}}\bar{\mathbf{v}} = \bar{\mathbf{v}}$, where a fixed point exists *if the operator is a contraction*. For the first moment, the Bellman operator is known to be a contraction [30]. This result, however, does not immediately extend here because, thought $\bar{r}_{t+1}$ is a valid finite reward, $\bar{\gamma}_{t+1}$ does not satisfy $\bar{\gamma}_{t+1} \le 1$, because ρ_t^2 can be large.

We can nonetheless define such a Bellman operator for the $\bar{\lambda}$-squared-return and determine if a fixed point exists. Interestingly, $\bar{\gamma}_{t+1}$ can in fact be larger than 1, and we can still obtain a contraction. To define the Bellman operator, we use a recent generalization that enables the discount to be defined as a function of (s, a, s') [34], rather than just as a function of s'. We first define $\bar{\mathbf{v}}$, the expected $\bar{\lambda}$-squared-return

$$\bar{\mathbf{v}} \overset{\text{def}}{=} \sum_{t=0}^{\infty} (\bar{\mathbf{P}}^{\pi,\gamma})^t \bar{\mathbf{r}},$$

where

$$\begin{aligned}\bar{\mathbf{P}}^{\pi,\gamma}(s,s') &\overset{\text{def}}{=} \sum_{s,a,s'} P(s,a,s')\mu(s,a)\rho(s,a)^2\gamma(s')^2\lambda(s')^2 \quad (3) \\ \bar{\mathbf{r}} &\overset{\text{def}}{=} \sum_{s,a,s'} P(s,a,s')\mu(s,a)\bar{r}(s,s').\end{aligned}$$

Using similar equations to the generalized Bellman operator [34], we can define

$$\bar{\mathbf{T}}\bar{\mathbf{v}} = \sum_{t=0}^{\infty} (\bar{\mathbf{P}}^{\pi,\gamma}\mathbf{\Lambda})^t \left(\bar{\mathbf{r}} + \bar{\mathbf{P}}^{\pi,\gamma}(\mathbf{I} - \mathbf{\Lambda})\bar{\mathbf{v}}\right)$$

where $\mathbf{\Lambda} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ is a matrix with $\lambda(s)$ on the diagonal, for all $s \in \mathcal{S}$. The infinite sum is convergent if the maximum singular value of $\bar{\mathbf{P}}^{\pi,\gamma}\mathbf{\Lambda}$ is less than 1, giving solution $\sum_{t=0}^{\infty}(\bar{\mathbf{P}}^{\pi,\gamma}\mathbf{\Lambda})^t = (\mathbf{I} - \bar{\mathbf{P}}^{\pi,\gamma}\mathbf{\Lambda})^{-1}$. Otherwise, however, the value is infinite and one can see that in fact the variance of the return is infinite!

We can naturally investigate when the second moment of the return is guaranteed to be finite. This condition on $\bar{\mathbf{P}}^{\pi,\gamma}\mathbf{\Lambda}$ should facilitate identifying theoretical conditions on the target and behavior policies that enable finite variance of the return. This theoretical characterization is outside of the scope of this work, but we can reason about different settings that provide a well-defined, finite fixed point. First, clearly setting $\lambda_{t+1} = 0$ for every state ensures a finite second moment, given a finite $\bar{\mathbf{v}}$, regardless of policy mis-match. For the on-policy setting, where $\rho_t = 1$, $\bar{\gamma}_{t+1} \le \gamma_{t+1}$ and so a well-defined fixed point exists, under standard assumptions (see [34]). For the off-policy setting, if $\bar{\gamma}_{t+1} = \lambda_{t+1}^2\gamma_{t+1}^2\rho_t^2 < 1$, this is similarly the case. Otherwise, a solution may still exist, by ensuring that the maximum singular value of $\bar{\mathbf{P}}^{\pi,\gamma}$ is less than one; we hypothesize that this property is unlikely if there is a large mis-match between the target and behavior policy, causing many large ρ_t. An important future avenue is to understand the required similarity between π and μ to enable finite variance of the return, for any given λ. Interestingly, the λ-greedy algorithm should adapt to such infinite variance settings, where (2) will set $\lambda_{t+1} = 0$.

5.2 VTD derivation

In this section, we propose Var-MSPBE, the mean-squared projected Bellman error (MSPBE) objective for the $\bar{\lambda}$-squared-return, and derive VTD to optimize this objective. Given the definition of the generalized Bellman operator $\bar{\mathbf{T}}$, the derivation parallels GTD(λ) for the first moment [9]. The main difference is in obtaining unbiased estimates of parts of the objective; we will therefore focus the results on this novel aspect, summarized in the below two theorems and corollary.

Define the error of the estimate $\mathbf{x}_t^\top \mathbf{w}^{\text{sq}}$ to the future $\bar{\lambda}$-squared-return

$$\bar{\delta}_t^{\bar{\lambda}} \overset{\text{def}}{=} \bar{V}_t^{\bar{\lambda}} - \mathbf{x}_t^\top \mathbf{w}^{\text{sq}}$$

and, as in previous work [24, 9], we define the MSPBE that corresponds to $\bar{\mathbf{T}}$

$$\text{Var-MSPBE}(\mathbf{w}^{\text{sq}}) \overset{\text{def}}{=} \mathbb{E}[\bar{\delta}_t^{\bar{\lambda}}\mathbf{x}_t]^\top \mathbb{E}[\mathbf{x}_t\mathbf{x}_t]^{-1}\mathbb{E}[\bar{\delta}_t^{\bar{\lambda}}\mathbf{x}_t].$$

To obtain the gradient of the objective, we prove that we can obtain an unbiased sample of $\bar{\delta}_t^{\bar{\lambda}}\mathbf{x}_t$ (a forward view) using a trace of the past (a backward view). The equivalence is simpler if we assume that we have access to an estimate of the first moment of the λ-return. For our setting, we do in fact have such an estimate, because we simultaneously learn $\mathbf{w}^{\text{err}}$. We include the more general expectation equivalence in Theorem 2, with all proofs in the appendix.

THEOREM 1. *For a given unbiased estimate $\bar{g}_{t+1}$ of $\mathbb{E}[G^{\lambda}_{t+1}|S_{t+1}]$, define*

$$\bar{\delta}_t \overset{def}{=} (\rho_t^2 \bar{G}_t^2 + 2\rho_t^2 \gamma_{t+1}\lambda_{t+1}\bar{G}_t \bar{g}_{t+1}) + \bar{\gamma}_{t+1}\mathbf{x}_{t+1}^\top \mathbf{w}_t^{sq} - \mathbf{x}_t^\top \mathbf{w}_t^{sq}$$

$$\bar{\mathbf{z}}_t \overset{def}{=} \mathbf{x}_t + \bar{\gamma}_{t+1}\bar{\mathbf{z}}_{t-1}$$

Then $$\mathbb{E}[\bar{\delta}_t^{\bar{\lambda}} \mathbf{x}_t] = \mathbb{E}[\bar{\delta}_t \bar{\mathbf{z}}_t]$$

THEOREM 2.

$$\mathbb{E}[\bar{r}_{t+1}\bar{\mathbf{z}}_t] = E[\rho_{t+1}^2 \bar{G}_t^2 \bar{\mathbf{z}}_t] + 2E[\rho_{t+1}^2 \bar{G}_t(\bar{\mathbf{a}}_t + \bar{\mathbf{B}}_t \mathbf{w}_t)]$$

where

$$\bar{\mathbf{a}}_t \overset{def}{=} \rho_t \gamma_t \lambda_t (R_t \bar{\mathbf{z}}_{t-1} + \bar{\mathbf{a}}_{t-1})$$

$$\bar{\mathbf{B}}_t \overset{def}{=} \rho_t \gamma_t \lambda_t (\gamma_t (1-\lambda_t)\bar{\mathbf{z}}_{t-1}\mathbf{x}_t^\top + \bar{\mathbf{B}}_{t-1})$$

COROLLARY 1. *For $\lambda_t = 1$ for all t,*

$$\mathbb{E}[\bar{r}_{t+1}\bar{\mathbf{z}}_t] = E[\rho_t^2 R_{t+1}^2 \bar{\mathbf{z}}_t] + 2E[R_{t+1}\bar{\mathbf{a}}_t]$$

where

$$\bar{\mathbf{z}}_t = \mathbf{x}_t + \rho_{t-1}^2 \gamma_t^2 \bar{\mathbf{z}}_{t-1}$$

$$\bar{\mathbf{a}}_t = \rho_t \gamma_t (R_t \bar{\mathbf{z}}_{t-1} + \bar{\mathbf{a}}_{t-1}).$$

To derive the VTD algorithm, we take the gradient of the Var-MSPBE. As this again parallels GTD(λ), we include the derivation in the appendix for completeness and provide only the final result here.

$$\begin{aligned}&-\tfrac{1}{2}\nabla\text{Var-MSPBE}(\mathbf{w}^{\text{sq}})\\ &= \mathbb{E}[\bar{\delta}_t^{\bar{\lambda}}\mathbf{x}_t] - \mathbb{E}[\bar{\gamma}_{t+1}(1-\bar{\lambda}_{t+1})\mathbf{x}_{t+1}\bar{\mathbf{z}}_t^\top]\mathbb{E}[\mathbf{x}_t\mathbf{x}_t]^{-1}\mathbb{E}[\bar{\delta}_t^{\bar{\lambda}}\mathbf{x}_t].\end{aligned}$$

As with previous gradient TD algorithms, we will learn an auxiliary set of weights $\mathbf{w}^{\text{sq}}$ to estimate a part of this objective: $\mathbb{E}[\mathbf{x}_t\mathbf{x}_t]^{-1}\mathbb{E}[\bar{\delta}_t^{\bar{\lambda}}\mathbf{x}_t]$. To obtain such an estimate, notice that $\mathbf{h}^{sq} = \mathbb{E}[\mathbf{x}_t\mathbf{x}_t]^{-1}\mathbb{E}[\bar{\delta}_t^{\bar{\lambda}}\mathbf{x}_t]$ corresponds to an LMS solution, where the goal is to obtain $\mathbf{x}_t^\top \mathbf{h}^{sq}$ that estimates $\mathbb{E}[\bar{\delta}_t^{\bar{\lambda}}|\mathbf{x}_t]$. Therefore, we can use an LMS update for $\mathbf{h}^{sq}$, giving the final set of update equations for VTD:

$$\begin{aligned}\bar{g}_{t+1} &\leftarrow \mathbf{x}_{t+1}^\top \mathbf{w}^{\text{err}}\\ \bar{r}_{t+1} &\leftarrow \rho_t^2 \bar{G}_t^2 + 2\rho_t^2\gamma_{t+1}\lambda_{t+1}\bar{G}_t\bar{g}_{t+1}\\ \bar{\gamma}_{t+1} &\leftarrow \rho_t^2\gamma_{t+1}^2\lambda_{t+1}^2\\ \bar{\delta}_t &\leftarrow \bar{r}_{t+1} + \bar{\gamma}_{t+1}\mathbf{x}_{t+1}^\top\mathbf{w}_t^{\text{sq}} - \mathbf{x}_t^\top\mathbf{w}_t^{\text{sq}}\\ \mathbf{w}_{t+1}^{\text{sq}} &\leftarrow \mathbf{w}_t^{\text{sq}} + \bar{\alpha}_h\bar{\delta}_t\bar{\mathbf{z}}_t - \bar{\alpha}_h\bar{\gamma}_{t+1}(1-\bar{\lambda}_{t+1})\mathbf{x}_{t+1}\bar{\mathbf{z}}_t^\top\mathbf{h}_t^{sq}\\ \mathbf{h}_{t+1}^{sq} &\leftarrow \mathbf{h}_t^{sq} + \bar{\alpha}_h\bar{\delta}_t\bar{\mathbf{z}}_t - \bar{\alpha}_h\mathbf{x}_t^\top\mathbf{h}_t^{sq}\mathbf{x}_t.\end{aligned}$$

For λ-greedy, we set $\bar{\lambda}_{t+1} = 1$, causing the term with the auxiliary weights to be multiplied by $1-\bar{\lambda}_{t+1}$, and so removing the need to approximate $\mathbf{h}^{sq}$.

6. RELATED WORK

There has been a significant effort to empirically investigate λ, typically using batch off-line computing and model-based techniques. Sutton and Singh [25] investigated tuning both α and λ. They proposed three algorithms, the first two assume the underlying MDP has no cycles, and the third makes use of an estimate of the transition probabilities and is thus of most interest in tabular domains. Singh and Dayan [18] provided analytical expression for bias and variance, given the model. They suggest that there is a largest feasible step-size α, below which bias converges to zero and variance converges to a non-zero value, and above which bias and/or variance may diverge. Downey and Sanner [3] used a Bayesian variant of TD learning, requiring a batch of samples and off-line computation, but did provide an empirical demonstration off optimally setting λ after obtaining all the samples. Kearns and Singh[5] compute a bias-variance error bound for a modification of TD called phased TD. In each discrete phase the algorithm is given n trajectories from each state. Because we have n trajectories in each state the effective learning rate is $1/n$ removing the complexities of sample averaging in the conventional online TD-update. The error bounds are useful for, among other things, computing a new λ value for each phase which outperforms any fixed λ value, empirically demonstrating the utility of changing λ.

There has also been a significant effort to theoretically characterizing λ. Most notably, the work of Schapire and Warmuth [17] contributed a finite sample analysis of incremental TD-style algorithms. They analyze a variant of TD called TD$^\star(\lambda)$, which although still linear and incremental, computes value estimates quite differently. The resulting finite sample bound is particularly interesting, as it does not rely on model assumptions, using only access to a sequence of feature vectors, rewards and returns. Unfortunately, the bound cannot be analytically minimized to produce an optimal λ value. They did simulate their bound, further verifying the intuition that λ should be larger if the best linear predictor is inaccurate, small if accurate and an intermediate value otherwise. Li [7] later derived similar bounds for another gradient descent algorithm, called residual gradient. This algorithm, however, does not utilize eligibility traces and converges to a different solution than TD methods when function approximation is used [24].

Another approach involves removing the λ parameter altogether, in an effort to improve robustness. Konidaris et al. [6] introduced a new TD method called TD$_\gamma$. Their work defines a plausible set of assumptions implicitly made when constructing the λ-returns, and then relaxes one of those assumptions. They derive an exact (but computationally expensive) algorithm, TD$_\gamma$, that no longer depends on a choice of λ and performs well empirically in a variety of policy learning benchmarks. The incremental approximation to TD$_\gamma$ also performs reasonably well, but appears to be somewhat sensitive to the choice of meta parameter C, and often requires large C values to obtain good performance. This can be problematic, as the complexity grows as $O(Cn)$, where n is the length of the trajectories—not linearly in the feature vector size. Nonetheless, TD$_\gamma$ constitutes a reasonable way to reduce parameter sensitivity in the on-policy setting. Garcia and Serre[4] proposed a variant of Q-learning, for which the optimal value of λ can be computed online. Their analysis, however, was restricted to the tabular case. Finally, Mahmood et al. [11] introduced weighted importance sampling for off-policy learning; though indirect, this is a strategy for enabling larger λ to be selected, without destabilizing off-policy learning.

This related work has helped shape our intuition on the role of λ, and, in special cases, provided effective strategies for adapting λ. In the next section, we add to existing work with an empirical demonstration of λ-greedy, the first λ-adaptation algorithm for off-policy, incremental learning, developed from a well-defined, greedy objective.

7. EXPERIMENTS

We investigate λ-greedy in a ring-world, under both on and off-policy sampling. This ring-world was previously introduced as a suitable domain for investigating λ [5]. We varied the length of the ring-world from $N = 10, 25, 50$. The reward is zero in every state, except for two adjoining states that have +1 and -1 reward, and are terminal states. The agent is teleported back to the middle of the ring-world upon termination. The target policy is to take action "right" with 95% probability, and action "left" with 5% probability. The feature representation is a tabular encoding of states with binary identity vectors, but we also examine the effects of aliasing state values to simulate poor generalization: a common case where the true value function cannot be represented. The length of the experiment is a function of the problem size, $N \times 100$, proportionally scaling the number of samples for longer problem instances.

We compared to fixed values of $\lambda = 0, 0.1, \dots, 0.9, 1.0$ and to two time-decay schedules, $\lambda_t = 10/(10+t), \lambda_t = 100/(100+t)$, which worked well compared to several other tested settings. The discount factor is $\gamma = 0.99$ for the on-policy chains and 0.95 for the off-policy chains. We include the optimal λ-greedy, which computes $\mathbf{w}^{\text{sq}}$ and $\mathbf{w}^{\text{sq}}$ using closed form solutions, defined by their respective Bellman operators. For λ-greedy, the initialization was $\mathbf{w}^{\text{sq}} = 0.0$ and $\mathbf{w}^{\text{err}} = \frac{Rmax}{1-\gamma} \times \mathbf{1}$ for on-policy, to match the rule-of-thumb of initializing with high lambdas, and the opposite for off-policy, to match the rule-of-thumb of more caution in off-policy domains. This max return value is a common choice for optimistic initialization, and prevented inadvertent evaluator bias by overly tuning this parameter. We fixed the learning-rate parameter for λ-greedy to be equal to equal to α used in learning the value function ($\mathbf{w}$), again to demonstrate performance in less than ideal settings. Sweeping the step-size for λ-greedy would improve performance.

The performance results in on and off-policy are summarized in Figure 1 and 2. We report the absolute value error compared to the true value function, which is computable in this domain for each of the settings. We average over 100 runs, and report the results for the best parameter settings for each of the algorithms with 12 values of $\alpha \in \{0.1 \times 2^j | j = -6, -6, ..., 5, 6\}$, 11 values of $\eta \in \{2^j | j = -16, -8, -4, \dots, 4, 8, 16\}$ ($\alpha_{\mathbf{h}} = \alpha\eta$).

In general, we find that λ-greedy works well across settings. The optimal λ-greedy consistently performs the best, indicating the merit of the objective. Estimating $\mathbf{w}^{\text{sq}}$ and $\mathbf{w}^{\text{sq}}$ typically cause λ-greedy to perform more poorly, indicating an opportunity to improve these algorithms to match the performance of the optimal, idealistic version. In particular, we did not optimize the meta-parameters in λ-greedy. For the fixed values of λ and decay schedules, we find that they can be effective for specific instances, but do not perform well across problem settings. In particular, the fixed decay schedules settings appear to be un-robust to an increasing chain length and the fixed λ are not robust to the change from on-policy to off-policy.

We also examined the λ value selected by our λ-greedy algorithm plotted against time. For tabular features, λ should converge to zero over-time, since the value function can be approximated and so, at some point, no bias is introduced by using $\lambda_t = 0$, but variance is reduced. The algorithm does converge to a state-based λ ($\lambda(s) \approx 0$ for all states), which was one of our goals to ensure we have a well-defined fixed point. Second, for aliased features, we expect that the final per-state λ should be larger for the states that have been aliased. The intuition is that one should not bootstrap on the values of these states, as that introduces bias. We demonstrate that this does indeed occur, in Figure 3. As expected, the λ-greedy is more robust to state aliasing, compared to the fixed strategies. State aliasing provides a concrete example of when we have less confidence in $\hat{v}$ in specific states, and an effective strategy to mitigate this situation is to set λ high in those states.

8. CONCLUSION AND DISCUSSION

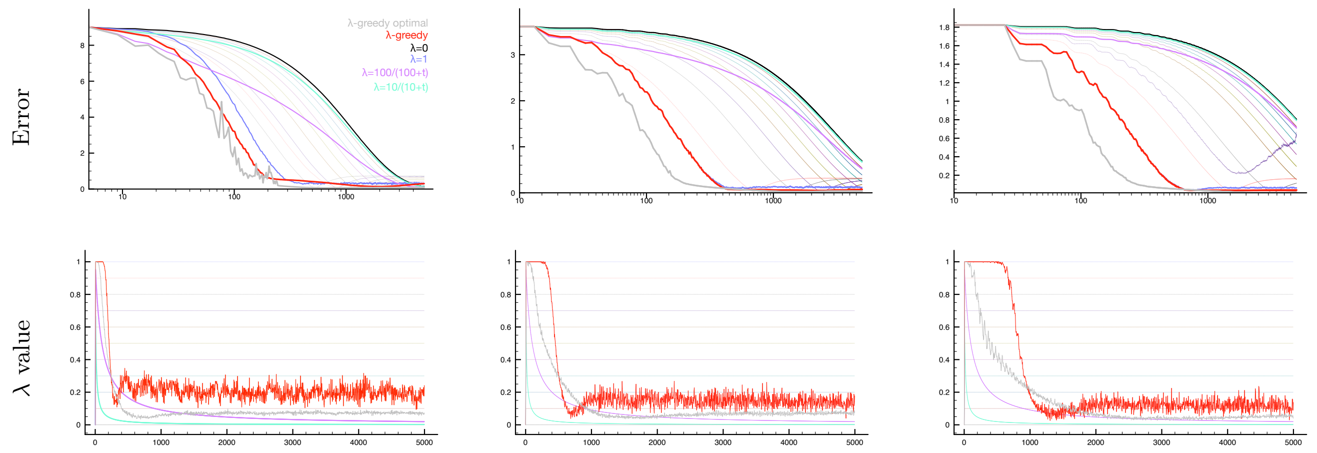
In this paper, we have proposed the first linear-complexity λ adaptation algorithm for incremental, off-policy reinforcement learning. We proposed an objective to greedily trade-off bias and variance, and derived an efficient algorithm to obtain the solution to this objective. We demonstrate the efficacy of the approach in an on-policy and off-policy setting, versus fixed values of λ and time-decay heuristics.

This work opens up many avenues for efficiently selecting λ, by providing a concrete greedy objective. One important direction is to extend the above analysis to use the recently introduced true-online traces [32]; here we focused on the more well-understood λ-return, but the proposed objective is not restricted to that setting. There are also numerous future directions for improving optimization of this objective. We used GTD($\lambda = 1$) to learn the second moment of the λ-return inside λ-greedy. Another possible direction is to simply explore using $\lambda < 1$ or even least-squares methods to improve estimation inside λ-greedy.

There are also opportunities to modify the objective to consider variance into the future differently. The current objective is highly cautious, in that it assumes that only λ_{t+1} can be modified, and assumes the agent will be forced to use the unbiased $\lambda_{t+i} = 1$ for all future time steps. As a consequence, the algorithm will often prefer to set λ_{t+1} small, as future variance can only be controlled by the setting of λ_{t+1}. A natural relaxation of this strictly greedy setting is to recognize that the agent, in fact, has control over all future $\lambda_{t+i}, i > 1$. The agent could remain cautious for some number of steps, assuming λ_{t+i} will likely be high for $1 < i \leq k$ for some horizon k. However, we can assume that after k steps, λ_{t+i} we be reduced to a small value, cutting off the traces and mitigating the variance. The horizon k could be encoded with an aggressive discount; multiplying the current discount $\bar{\gamma}_{t+1}$ by a value less than 1. Commonly, a multiplicative factor less than 1 indicates a horizon of $\frac{1}{1-\text{factor}}$; for example, $\frac{1}{1-0.8} = 5$ gives a horizon of $k = 5$. The variance term in the objective in (2) could be modified to include this horizon, providing a method to incorporate the level of caution.

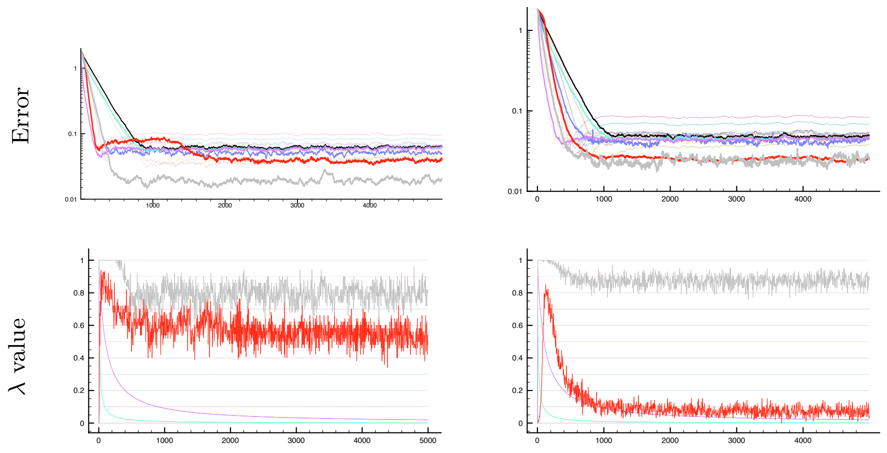
Overall, this work takes a small step toward the goal of automatically selecting the trace parameter, and thus one step closer toward parameter-free black-box application of reinforcement learning with many avenues for future research.

Acknowledgements
We would like to thank David Silver for helpful discussions and the reviewers for helpful comments.



Chain 10 Chain 25 Chain 50

Figure 1: **On-policy** learning performance of GTD(λ) with different methods of adapting λ, for three sizes in the ringworld domain. The first row of results plots the mean squared value error verses time for several approaches to adapting λ: the optimal λ-greedy algorithm, our approximation algorithm, several fixed values of λ, and two fixed decay schedules. The second row of results plots the λ values over-time used by each algorithm. We highlight early learning by taking the log of y axis, as the algorithms should be able to exploit the low variance to learn quickly. For off-policy learning, there is more variance due to importance sampling and so we prefer long-term stability. We therefore highlight later learning using the log of the x-axis.



$\pi(s, \text{right}) = 0.95, \mu(s, \text{right}) = 0.85$ $\pi(s, \text{right}) = 0.95, \mu(s, \text{right}) = 0.75$

Figure 2: **Off-policy** learning performance of GTD(λ) with different methods of adapting λ, for two different configurations of π and μ in the size 10 ringworld domain. The first row of results plots the mean squared value error verses time for several approaches to adapting λ. In off-policy learning, there is more variance due to importance sampling and so we prefer long-term stability. We therefore highlight later learning using the log of the x-axis. The second row of results plots the λ values over-time used by each algorithm.

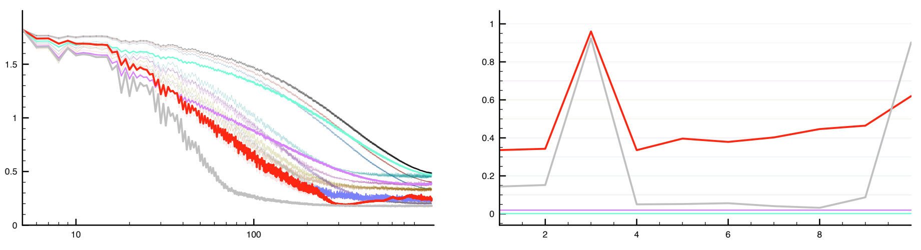


Figure 3: A closer examination of the effects of state aliasing on the choice of state-based λ, for the length 10 chain with $\gamma = 0.95$, under on-policy sampling. State three is aliased with the last state. The left graph shows the learning curves over 5000 steps, and the right graph shows the learned λ function in each state at the end of the run.

REFERENCES

[1] P. L. A and M. Ghavamzadeh. Actor-Critic algorithms for risk-sensitive MDPs. In *Advances in Neural Information Processing Systems*, 2013.

[2] W. Dabney and A. G. Barto. Adaptive step-size for online temporal difference learning. In *AAAI*, 2012.

[3] C. Downey and S. Sanner. Temporal difference Bayesian model averaging: A Bayesian perspective on adapting lambda. In *International Conference on Machine Learning*, 2010.

[4] F. Garcia and F. Serre. From Q(lambda) to Average Q-learning: Efficient Implementation of an Asymptotic Approximation. In *International Joint Conference on Artificial Intelligence*, 2001.

[5] M. J. Kearns and S. P. Singh. Bias-Variance error bounds for temporal difference updates. In *Annual Conference on Learning Theory*, 2000.

[6] G. Konidaris, S. Niekum, and P. S. Thomas. TDgamma: Re-evaluating Complex Backups in Temporal Difference Learning. In *NIPS*, 2011.

[7] L. Li. A worst-case comparison between temporal difference and residual gradient with linear function approximation. In *International Conference on Machine Learning*, 2008.

[8] L.-J. Lin. Self-Improving Reactive Agents Based On Reinforcement Learning, Planning and Teaching. *Machine Learning*, 1992.

[9] H. Maei. *Gradient Temporal-Difference Learning Algorithms*. PhD thesis, University of Alberta, 2011.

[10] H. Maei and R. Sutton. GQ (λ): A general gradient algorithm for temporal-difference prediction learning with eligibility traces. In *AGI*, 2010.

[11] A. R. Mahmood, H. P. van Hasselt, and R. S. Sutton. Weighted importance sampling for off-policy learning with linear function approximation. In *Advances in Neural Information Processing Systems*, 2014.

[12] S. Mannor, D. Simester, P. Sun, and J. N. Tsitsiklis. Bias and variance in value function estimation. In *International Conference on Machine Learning*, 2004.

[13] S. Mannor and J. N. Tsitsiklis. Mean-Variance pptimization in Markov Decision Processes. In *ICML*, 2011.

[14] J. Modayil, A. White, and R. S. Sutton. Multi-timescale nexting in a reinforcement learning robot. *Adaptive Behavior - Animals, Animats, Software Agents, Robots, Adaptive Systems*, 2014.

[15] T. Morimura, M. Sugiyama, H. Kashima, H. Hachiya, and T. Tanaka. Parametric return density estimation for reinforcement learning. In *Conference on Uncertainty in Artificial Intelligence*, 2010.

[16] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development ()*, 1959.

[17] R. E. Schapire and M. K. Warmuth. On the worst-case analysis of temporal-difference learning algorithms. *Machine Learning*, 1996.

[18] S. P. Singh and P. Dayan. Analytical mean squared error curves in temporal difference learning. In *Advances in Neural Information Processing Systems*, 1996.

[19] S. P. Singh and R. S. Sutton. Reinforcement learning with replacing eligibility traces. *Machine Learning*, 1996.

[20] M. J. Sobel. The variance of discounted Markov Decision Processes. *Journal of Applied Probability*, 1982.

[21] R. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 1988.

[22] R. Sutton. TD models: Modeling the world at a mixture of time scales. In *International Conference on Machine Learning*, 1995.

[23] R. Sutton and A. G. Barto. *Introduction to reinforcement learning*. MIT Press, 1998.

[24] R. Sutton, H. Maei, D. Precup, and S. Bhatnagar. Fast gradient-descent methods for temporal-difference learning with linear function approximation. *International Conference on Machine Learning*, 2009.

[25] R. Sutton and S. P. Singh. On step-size and bias in temporal-difference learning. In *Proceedings of the Eighth Yale Workshop on Adaptive and Learning Systems*, 1994.

[26] R. S. Sutton, A. R. Mahmood, and M. White. An emphatic approach to the problem of off-policy temporal-difference learning. *The Journal of Machine Learning Research*, 2015.

[27] R. S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 1999.

[28] A. Tamar, D. Di Castro, and S. Mannor. Temporal difference methods for the variance of the reward to go. In *International Conference on Machine Learning*, 2013.

[29] G. Tesauro. Practical issues in temporal difference learning. *Machine Learning*, 1992.

[30] J. N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *Automatic Control, IEEE Transactions on*, 42(5):674–690, 1997.

[31] H. van Hasselt, A. R. Mahmood, and R. Sutton. Off-policy TD (λ) with a true online equivalence. In *Conference on Uncertainty in Artificial Intelligence*, 2014.

[32] H. van Seijen and R. Sutton. True online TD(lambda). In *International Conference on Machine Learning*, 2014.

[33] A. White. *Developing a predictive approach to knowledge*. PhD thesis, University of Alberta, 2015.

[34] M. White. Transition-based discounting in Markov decision processes. *In preparation*, 2016.

[35] M. White and A. White. Interval estimation for reinforcement-learning algorithms in continuous-state domains. In *Advances in Neural Information Processing Systems*, 2010.