RL–2 Project (Revised Version): Utility of Traces in Online Value Prediction with TD(λ)

Shivam Garg
7th April 2020

1 Introduction

The two fundamental goals of Reinforcement Learning are to predict and control a stream of data. In this project, we focus on online on–policy value prediction for a fixed policy.

We first briefly outline our experimental setting and the notation by describing the agent, the algorithm (it uses for value prediction), and the environment (it interacts with) in Sections 1.1 and 1.2. After that, we motivate our hypothesis in Section 1.3.

1.1 Agent

An agent learns to control and predict a stream of data by interacting with an environment. In the context of RL, learning to control means to learn a policy π, and learning to predict means to learn an estimate of the true value function v_π corresponding to this specific policy. We specify the agent's policy in the next section when we describe the environment.

There are two different ways to estimate a state value function: (1) tabular function; and (2) function approximator. In the tabular setting, we maintain a value estimate for each state. In function approximation setting, we use a function parameterized by a weight vector $\boldsymbol{w}$, which takes the state features as an input and outputs the state's value. The goal then is to learn this weight vector $\boldsymbol{w}$ which minimizes the error between our value estimates and the true values of different states.

For our experiments, we consider a linear function approximator $\hat{v}(s, \boldsymbol{w}) \doteq \boldsymbol{w}^\top \boldsymbol{x}(s)$, where $\hat{v}(s, \boldsymbol{w})$ is the value estimate of state s and $\boldsymbol{x}(s)$ represents the feature vector for state s. Observe that linear function approximators contain tabular functions as a special case: when the feature vector of all the states is a one–hot vector (see Section 2.1).

For learning this weight vector $\boldsymbol{w}$, we will use the TD(λ) algorithm (Sutton, 1988). TD(λ) employs eligibility traces, which allows it to smoothly transition between TD(0) and the Monte–Carlo method. We pick TD(λ), instead of alternatives like GTD(λ) and others, because it is simple to implement and has lesser number of tunable parameters which makes it easier to use. The update equations for on–policy TD(λ) with linear function approximator are:

$$\boldsymbol{z}_t = \gamma\lambda\, \boldsymbol{z}_{t-1} + \boldsymbol{x}(S_t), \tag{1}$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha\delta_t\, \boldsymbol{z}_t, \tag{2}$$

where $\boldsymbol{z}_t$ is the eligibility trace vector which is initialized to zero at the beginning of each episode, γ is the discount factor, λ parameter decides the weighting on the different n–step returns ($\lambda = 0$ refers to the one step return used in TD(0) and $\lambda = 1$ refers to the full return used in Monte–Carlo estimation), α is the stepsize parameter, and δ_t is the standard TD error defined as $\delta \doteq R_{t+1} + \gamma\hat{v}(S_{t+1}, \boldsymbol{w}_t) - \hat{v}(S_t, \boldsymbol{w}_t)$.

1.2 Environment: Random Walk

We experiment with a standard five state random walk (Sutton & Barto, 2018) as shown in Figure 1. This is an undiscounted and episodic task. The environment has five states labeled from 1 to 5. At the beginning of the episode, the agent starts in the 3$^{\text{rd}}$ state. In each state, it can either take a left or right action which then moves it into the corresponding next state deterministically. If the agent takes the left action in state 1 or the right action in state 5, it moves to the terminal state and the episode ends. The transition into the right most state has a reward of 1. All other transitions have a reward of 0.

In all our experiments, we fix our agent's policy π to pick a left or right action in each state with equal probability. Under this policy distribution, we can analytically solve for the true value function v_π and the stationary state distribution d_π for our environment, which we show for each state in Figure 1.

We chose this environment and policy because of their simplicity: they are straightforward to implement, quick to run which allows an extensive parameter study, and relatively easier to reason when about compared with other MDPs.

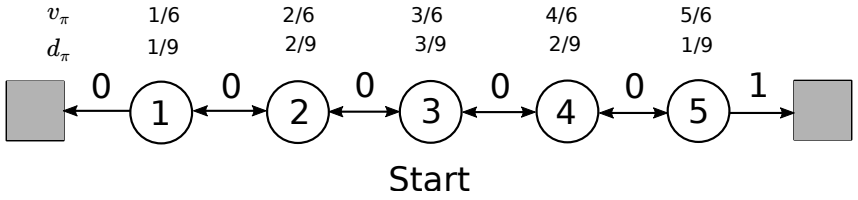

Figure 1: The five state random walk environment. v_π and d_π correspond to the true value function and the stationary state distribution under the random policy π which takes an action of left or right in each state with equal probability.

1.3 Hypothesis

The final goal of all prediction methods is to minimize the mean squared value error (MSVE) defined by

$$\text{MSVE}(\boldsymbol{w}) = \sum_s d_\pi(s)(v_\pi(s) - \hat{v}(s, \boldsymbol{w}))^2. \tag{3}$$

Monte–Carlo methods directly minimize this error by sampling returns $G_t \doteq \sum_{i=0}^{\infty} R_{t+1+i}$ and using them as estimates of the true value function. The return is calculated by summing over the rewards obtained at each timesteps. The return G_t can have a very high variance across independently sampled trajectories all generated by following the same policy. The reason for this is that G_t is the sum of accumulated rewards along the whole trajectory, and due to the stochasticity in the environment dynamics and the policy, the rewards and even the trajectory of the agent for a similar behavior can be very different at different trials (Dann et al., 2014). This difference in turn leads to a high variance of the Monte–Carlo returns. As an analogy, consider this: the outcome of dropping a ball from precisely the same height and position can result in very different final positions (each different randomness of the environment, such as wind, adds up and leads to a different final position).

On the other hand TD(0) uses the one step return $R_{t+1} + \gamma\hat{v}(S_{t+1}, \boldsymbol{w})$, as a proxy of the full return. The one step return only depends on one sampled reward and thus has a lower variance compared to the full return G_t. However, using one step returns in place of full returns leads to a bias in the final TD solution: TD(0) converges to the minimum of the mean squared projected bellman error (MSPBE). And it is known that the minimum of MSPBE can have an arbitrarily high MSVE on certain MDPs (Dann et al., 2014).

Thus, we can say that the error in the final TD prediction comes from, amongst others, two sources: sampling error and objective bias (Dann et al., 2014). In prediction, sampling error can correspond to how accurate the sampled returns G_t, are of the expected returns $v_\pi(S_t)$. Objective bias corresponds to how close the minimum of the prediction method (such as MSPBE in case of TD) is to MSVE.

Eligibility traces help to trade–off the sampling error (by controlling the variance of the updates) with the objective bias (moving between the minimum of MSVE and MSPBE). The magnitude of sampling error and objective bias also depends on the environment (how stochastic it is), the feature representation, and other factors. For example, when the true value function is perfectly representable by the features, the minimum of MSPBE exactly corresponds to the minimum of MSVE and the objective bias disappears.

Main Hypothesis: *In absence of objective bias, smaller values of λ in TD(λ) will lead to a lower MSVE than larger values of λ, i.e. in absence of objective bias, introducing eligibility traces in TD(0) doesn't improve the quality of value prediction asymptotically.* We believe this is true because in absence of objective bias, having $\lambda > 0$ in TD(λ), i.e. moving from using one–step returns to full returns increases the variance of the updates while having no effect on the quality of the final solution. The main source of error in this setting is from the variance of the updates and thus utilizing multi–step returns is not useful. For the same reason, we also think that as the stochasticity of the updates will increase, the benefit of using eligibility traces would decrease even further.

To verify the validity of our hypothesis, we conducted prediction experiments using TD(λ). We provide the experimental results and discussion in Sections 2 and 3. Finally, we give our conclusions in Section 4.

2 Experiment 1: TD(λ) on Simple Random Walk

Our first experiment is online on–policy value prediction using TD(λ) on the Simple Random Walk environment described in Section 1.2. The MDP is undiscounted, i.e. $\gamma = 1$. In all our experiments, we execute the random policy which takes actions left or right in each state of the random walk with equal probability for 110 episodes. As described before, we use a linear value function approximator for value prediction. The weight vector is initialized to zero at the start of each experiment. All the results are averaged over 100 independent runs. We give results for two different feature representations, both of which can perfectly represent the true value function and as a result there does not exist any objective bias in our final value estimates.

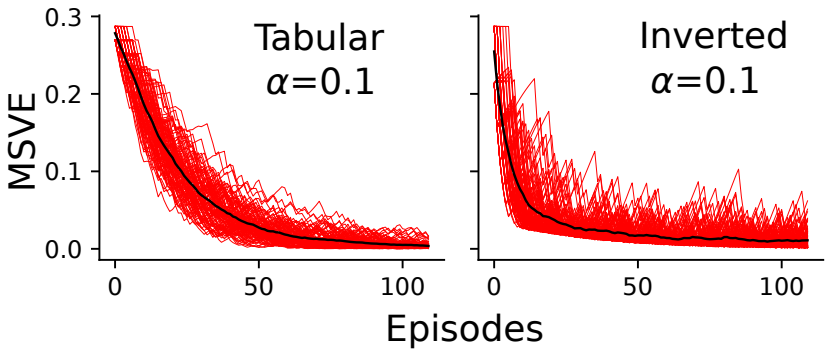

Figure 2: Learning curves for TD(0) using linear function approximation with tabular and inverted features on five state random walk. The black line shows the error average over 100 independent runs. The finer red lines correspond to the errors during the individual runs.

2.1 Results for Tabular Features

To develop intuition and understand the experiment better, we first show the results for tabular features. Tabular features refer to one–hot encoding for the states in the random walk. For instance, the feature vector for state 3 is $[0, 0, 1, 0, 0]^\top$. We call these features as tabular, because linear function approximation with such features reduces to the tabular setting. For these features, our weight vector would have five dimensions.

Learning Curve: We plot the MSVE for tabular features at the end of each episode (calculated using Eq. 3), for 100 different runs for TD(0) and stepsize parameter $\alpha = 0.1$ in Figure 2. We can see from the figure that the initial error was 0.29, since our initial value estimate was 0 for each state (refer Eq. 3). As the learning progressed, we see that the error went to almost zero. Also notice from the figure that some of the red lines were flat during the initial episodes; this means that the weights did not change for these runs initially. This happened because these runs must have ended in the left terminal state and thus all transitions had a reward of 0.

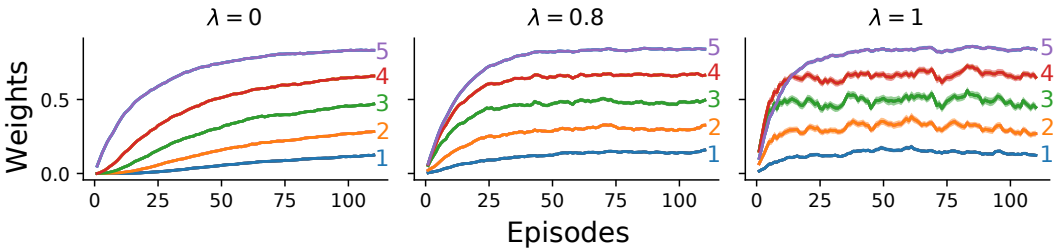

Figure 3: Learning curves for TD(λ) using linear function approximation with tabular features on five state random walk. The different subplots correspond to different values of λ. The five different curves correspond to different components of the weight vector, one for each state. Stepsize is set as $\alpha = 0.1$ in all the plots. The curves show mean and standard error (vanishingly small) over 100 independent runs.

To gain further insight into tabular TD(λ), we plot the weight vector at the end of each

episode for different values of λ and a fixed stepsize of $\alpha = 0.1$ in Figure 3. From the figure, we notice that for all the values of λ, the weight vector nearly converged to the true value function (given in Figure 1). Also notice that for $\lambda = 0$, the weight element 5 (corresponding to fifth state) started changing faster as compared to the other weight elements. Similarly, weight element 1 changed most slowly. We observe this because of the reward structure in the random walk: the updates started happening from the rightmost (fifth) state and then slowly bootstrapped to the left most (first) state. On the other hand, for larger values of λ all the weight elements increased much more quickly when compared to smaller λ values, because eligibility traces update the value of multiple states in a single TD update step (see backward view in Sutton & Barto, 2018). Further, notice that the weight vectors for larger values of λ showed higher variance than smaller values of λ.

Parameter Study: We present the parameter study for tabular features using TD(λ) in Figure 4. We sweeped over eight different values of λ and for $\alpha \in \{0, 0.0025, 0.005, \ldots, 0.0475\} \cup \{0.05, 0.06, \cdots, 0.99\}$. This range of α was chosen to produce smooth graphs. The plot shows the average MSVE over the last 10 episodes of the agent (the agent ran for 110 episodes). Further, we clipped the MSVE at 0.3 for better viewing of the curves.

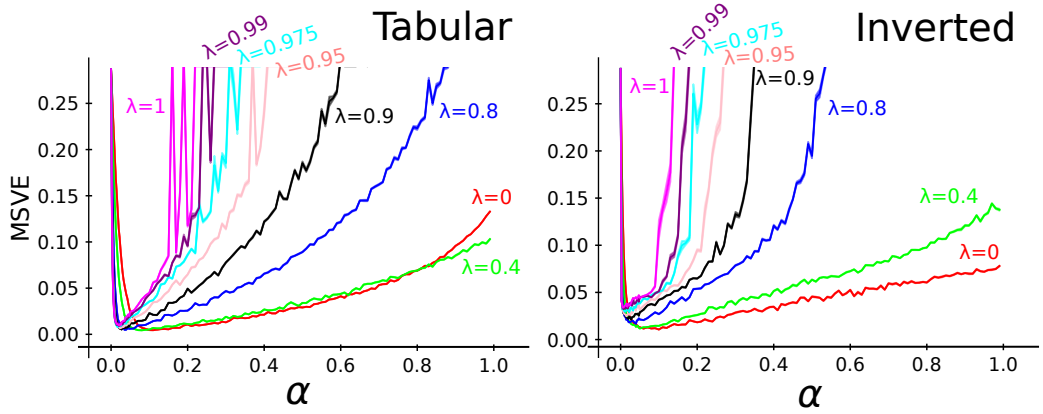

Figure 4: Parameter Study over α and λ for the final performance of TD(λ) on five state random walk. The error is calculated by averaging over the MSVE in the last 10 episodes of the 110 episodes during which the agent interacts with the environment. The curves show mean and standard error (vanishingly small) over 100 independent runs.

From the figure, we see that the curves for all the values of λ reached a final error of about zero for a small value of α. On this problem, $\lambda = 0$ and $\lambda = 0.4$ had the least sensitivity to α, whereas all curves for $\lambda \geq 0.8$ started diverging as α increased. We can explain this observation by noting that the forward and backward equivalence of λ returns relies on the value function not changing by much at each step. However, as α increases the value function changes drastically and the backward view is no longer equivalent to the forward view – the higher the λ is, the higher the difference between the two views would be. Also observe that the curves for larger λs had much more variance than those with small λs. This is due to the variance of the multi–step return updates. Also for large αs, an intermediate value of $\lambda = 0.4$ had the lowest error.

Note: We also report the parameter study for the initial performance of TD(λ) in Appendix

A. Those figures (1) Serve as a baseline for comparison against the graphs given in other sources (Sutton & Barto, 2018) and ensuring that our implementation is correct; and (2) Show how the initial performance of the prediction algorithm is affected by different learning rates and λ values.

2.2 Results for Inverted Features

We now give the results for TD(λ) with inverted features (Sutton et al., 2009) on the five state random walk. All the experimental details remain the same as in the previous section. Inverted features are obtained from tabular features by replacing 1s with 0s and vica–versa. For instance, the inverted feature representation for state 3 is $\left[\frac{1}{2}, \frac{1}{2}, 0, \frac{1}{2}, \frac{1}{2}\right]^\top$. Note that the feature vector is normalized. We chose this feature representation because it causes inappropriate generalization between different states which increases the stochasticity of the TD updates. Also, the true value function is exactly representable with this feature representation, and so as before there is no objective bias (see Section 1.3) present in the problem.

We show the error learning curves for TD(0) with inverted features in Figure 2. We observe that TD(0) with inverted features converged faster than tabular features. The reason for this is that all the TD errors in this MDP were positive. As a result, due to the inverted feature representation, the value of all the states was increased whenever any state got updated. Whereas in tabular setting, only one state got updated at a time. Due to this reason, the inverted setting converged way faster than tabular setting. However, the final solution was marginally inferior to the tabular case. The reason for this, most likely, is that the updates in the inverted setting are more stochastic and since we use a constant stepsize in the updates, the noise terms don't not averaged out nicely. We believe that had we used a decreasing stepsize, the final solution in the inverted setting would have converged to an equally good MSVE as the tabular setting.

The learning curve for weight vectors of TD(λ) are given in Figure 5. All our observations are similar to those given for Figure 3.

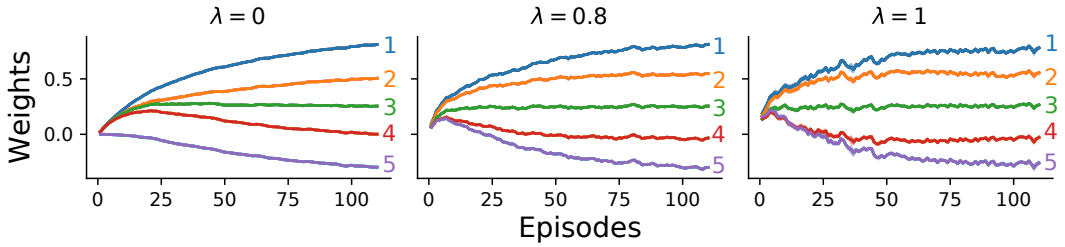

Figure 5: Learning curves for TD(λ) using linear function approximation with inverted features on five state random walk. Details are same as given in Figure 3.

Finally, we give the parameter study for inverted features in Figure 4. We observe that the inverted features in general had worse final performance than tabular features. Further, the curves for larger λs were more sensitive to the stepsize parameter than for tabular features. Interestingly though, we observe that, in contrast to tabular features, the best performing λ was zero in this problem across all values of α. This can be attributed to the increased randomness

in the updates to the state value estimates arising due to the inverted features. This increased randomness in turn lead to lower values of λ being preferred.

2.3 Conclusions

We observe that for the five state random walk, TD(λ) with large values of λ has high sensitivity to α with tabular features. This effect gets even more significant with inverted features since there is an increased stochasticity in the updates. This result verifies our hypothesis that in absence of objective bias (for example with tabular features), eligibility traces are not helpful in obtaining better value predictions asymptotically. We also observed that inverted features converge to an inferior solution when compared with tabular features. We believe that this holds in general for all MDPs – tabular features provide the best possible solution but can be slow to converge. Finally, we experimentally verified that the weight vectors have more variance in the TD updates with large λ (multi–step returns) compared with low values of λ.

3 Experiment 2: TD(λ) on a More Stochastic Random Walk

3.1 Left Negative Random Walk

We repeat the previous set of experiments on a similar five state random walk. The only change, as shown in 6, is that now a transition into the left most terminal state yields a reward of -1. We call this random walk as the "left negative random walk". With a new reward structure, the true value function v_π for the problem changes as shown in the figure.

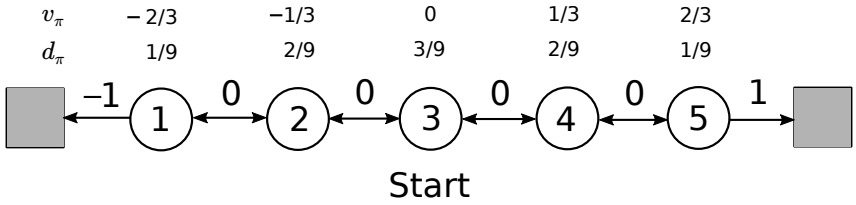

Figure 6: The five state left negative random walk environment. Details are same as given in Figure 1.

Adding this -1 reward makes this MDP have more stochastic updates. The reason is that now all transitions in an episode, irrespective of whether they end in the left or right state, are going to have non–zero TD errors and thus will make updates. Further, the states towards left are going to make negative updates, whereas the states towards right are going to make positive updates. This overall increases the variance of the updates. All the other experimental details remain similar to those given in Section 2.

3.2 Results and Discussion

Learning Curves: We show the learning curves for TD(0) on the left negative random walk in Figure 7. We plot the curves for two good performing values of αs (found from parameter study in Figure 8).

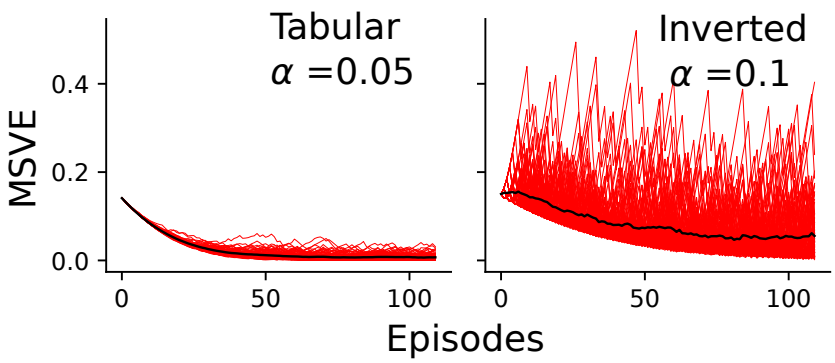

Figure 7: Learning curves for TD(0) using linear function approximation with tabular and inverted features on five state left negative random walk. Details are same as given in Figure 2.

Learning curves in Figure 7 are similar to those in Figure 2 except for two differences: (1) There were no flat regions in the individual learning curves (corresponding to no updates during initial episodes) in tabular setting since there was a non–zero reward at the end of each episode; and (2) The Inverted features converged more slowly as compared to the tabular features and had much higher variance amongst individual runs than the tabular features. This happened because in the left negative random walk environment, there were both negative and positive TD updates. As a result, the inappropriate generalization due to inverted features made the updates overwrite each out leading to higher variance and slower convergence.

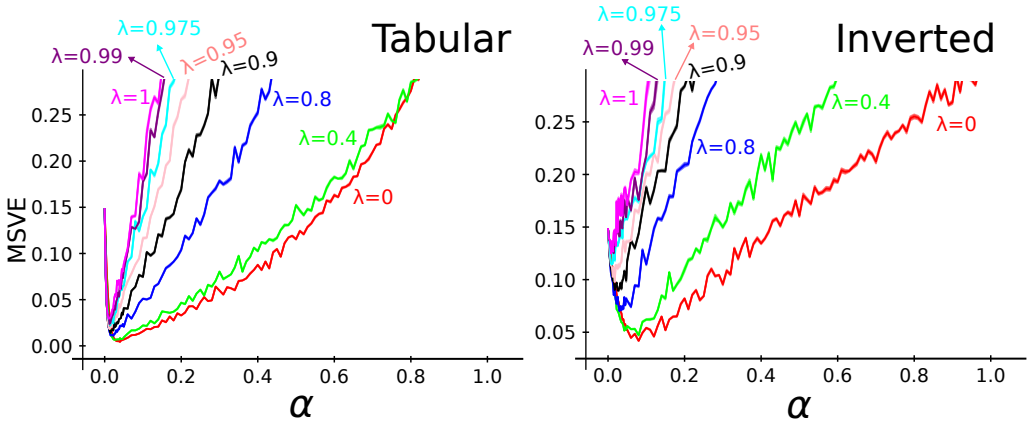

Figure 8: Parameter Study over α and λ for the final performance of TD(λ) on five state left negative random walk. Details are same as given in Figure 4.

Parameter Study: We show the parameter study over different values of λ and α for TD(λ) on the five state left negative random walk in Figure 8. The experimental details for the parameter study are same as given in Section 2.1. All the curves in the tabular setting were able to converge to a good solution for a small enough value of α. We also observe that, compared to Figure 4, all the curves became much more sensitive to α for the left negative random walk. This again happened because of higher variance in the TD updates. As a result, the larger values of λ performed very poorly even with a slightly large value of α. Also note that this effect gets more pronounced with inverted feature representations: the curves were a lot more sensitive compared to the tabular setting – they diverged at smaller values of α. Even the final

solutions, in case of inverted features with large λs, were as bad as a system with $\boldsymbol{w} = 0$.

Note: We again report the parameter study for the initial performance of TD(λ) in Appendix A.

4 Conclusions

Our original hypothesis was that with no objective bias, since the minimum of MSPBE is as good as the minimum of MSVE, using multi–step returns is of limited use in value prediction asymptotically. Our motivation for making this hypothesis was that multi–step returns can result in high variance updates as compared to single step returns and thus lead to inferior value predictions. The experiments we conducted verified this as we previously specified.

Further, we observed that the utility of introducing eligibility traces in TD(0) decreases even further as the stochasticity in the TD updates increases: left negative random walk has more stochastic updates than simple random walk and thus TD(λ) on it performs poorly for most values of α. Similar conclusions hold true for the use of inverted features.

We also noted that features (like inverted features) which have inappropriate state generalization lead to faster or slower convergence, compared to tabular features, depending on the MDP properties. They also lead to inferior solutions compared to the tabular features.

In conclusion, eligibility traces can effectively trade–off objective bias with the sample error. However, if no objective bias is present in the problem, it is likely better (in terms of achieving lower asymptotic value error) to use single step prediction methods such as TD(0).

References

Dann, C., Neumann, G., Peters, J. (2014). Policy evaluation with temporal differences: A survey and comparison. *Journal of Machine Learning Research* 15 (2014): 809-883.

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

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 *Proceedings of the 26th Annual International Conference on Machine Learning* (pp. 993-1000).

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction,* Second Edition. MIT Press.

A Results for Initial Learning Performance of TD(λ)

We report the initial performance (the average MSVE over the first 10 episodes) of TD(λ) for tabular and inverted features on the five state random walk in Figure 9 and on the five state

left negative random walk in Figure 10. We observe some similarities and differences from the previous parameter studies reporting the final performance of TD(λ) on the same tasks (Figure 4 and Figure 8).

Similarities: As before, we observe that curves with larger value of λ were more sensitive to large stepsizes and started diverging on smaller values of stepsizes as compared to curves with smaller values of λ. The other similarity is that this sensitivity increased as the stochasticity of the updates increased (such as in the left negative random walk experiment).

Differences: The key differences in the curves for initial performance, as compared to the curves for final performance, was that for very low values of stepsizes, higher λ helped in achieving a lower initial error. The reason for this is that eligibility traces helped propogate the reward faster along multiple states. We observed a similar phenomenon in Figure 3 and Figure 5 which showed that the weight vectors for larger λ changed faster as compared to smaller values of λ. However, again as we discussed in the previous paragraph, as the updates became more stochastic, this benefit of faster propogation of rewards decreases and became less and less significant. The reason for this is that since the updates were very stochastic, a faster propogation just resulted in poor quality updates to the weight vectors and a higher value error.

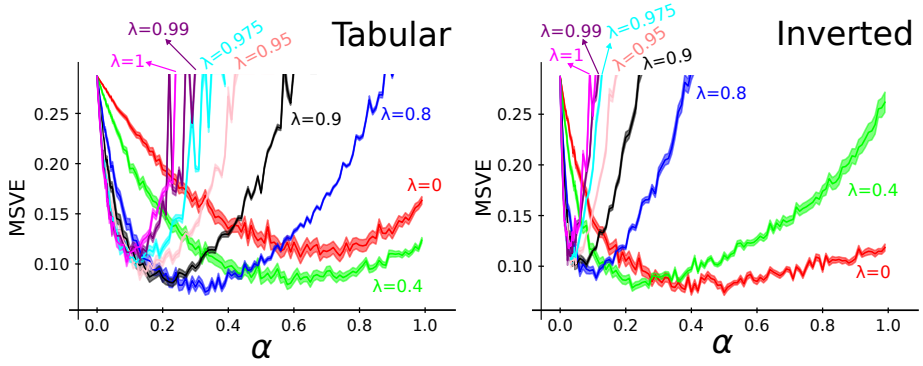

Figure 9: Parameter Study over α and λ for the initial performance of TD(λ) on five state random walk. The error is calculated by averaging over the MSVE in the first 10 episodes during which the agent interacts with the environment. The curves show mean and standard error (vanishingly small) over 100 independent runs.

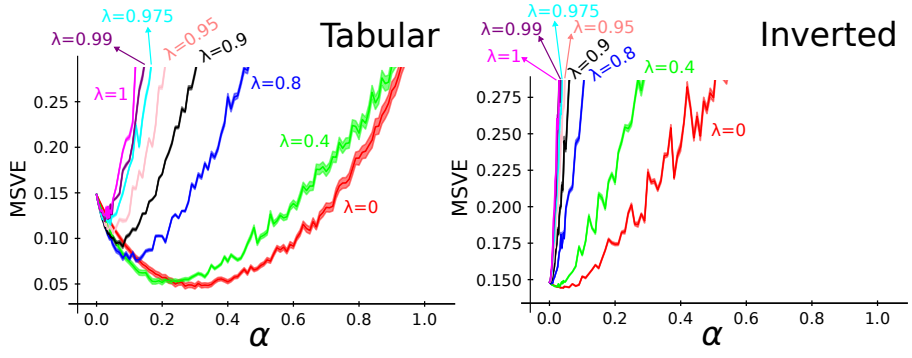

Figure 10: Parameter Study over α and λ for the initial performance of TD(λ) on five state left negative random walk. Details are same as given in Figure 9.