Reinforcement Learning and Decision Making
Project 1: TD(λ)

Hanxiao Wu, hwu429@gatech.edu

Last Git Hash: 8ca906a11fd6bdba79df90806ee7350d915624a8

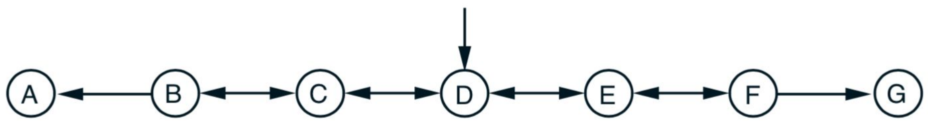
1 INTRODUCTION

Supervised learning solves prediction-learning problems using pairs of observed data and actual outcome to estimate the function and make predictions. However, the multi-step prediction problems, where new information is received along the way of reaching the actual outcome, reveals a limitation of supervised learning: the valuable sequential information is ignored. Temporal Difference learning, on the other hand, shows great value here.

In this analysis, we will replicate the random walk experiments in a paper written by Sutton in 1988, Learning to Predict by the Method of Temporal Differences, to show the advantages of TD methods, compared to classic supervised learning in multi-step prediction problems.

2 METHODOLOGY

The random walk example can be described as the following:



All walks begin in state D. At each state except the terminal states, the walk has a 50-50 chance of moving to either right or left, with 0 reward. There are two terminal states, A and G, where you will receive 0 reward or 1 reward when entered, and then the walk terminates.

It can be formed as a prediction problem: what is the expected return when the walk is at a certain state? For example, at state D, where you have 50% chance to reach G and get 1 reward, the expected return is 0.5. While at state F, where the walk is much more likely to terminate at G, intuitively the expected return should be larger than 0.5. It can also be formed as a Markov process where the expected return at each state is the value function estimation of each state.

For state B, C ,D, E, and F, there is an observation vector for each of them, with four 0 components and one 1 component appearing at a different place for each state. At time t, based on the state you are in (x_t), you will make a prediction on the return of this walk $P_t = w^T x_t$.

The learning process is the process of updating the weight vector: w. In supervised learning, the weight vector can only be updated when the actual outcome z is observed. As a result, Δw can not be computed incrementally.

$$w \leftarrow w + \sum_{t=1}^{m} \Delta w_t$$

$$\Delta w_t = \alpha (z - P_t) \nabla_w P_t$$

In TD Learning, by representing the error $(z - P_t)$ as a sum of differences in predictions, the Δw_t can be converted to the following equation:

$$\Delta w_t = \alpha (P_{t+1} - P_t) \sum_{k=1}^{t} \nabla_w P_k$$

As the equation suggested, Δw_t only depends on the successive prediction and the sum of past gradients. So this update rule can be computed incrementally, while producing the same weight changes as the Widrow-Hoff procedure.

The above update rule is called TD(1), indicating that all predictions are updated to an equal extent. However, it makes more sense to give the more recent predictions greater alterations as intuitively they are more 'accountable'. Here comes the TD(λ) procedure where the prediction made k steps past will be updated according to λ^k where $0 \le \lambda \le 1$.

$$\Delta w_t = \alpha (P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k$$

In a special case of TD(λ) where λ=0, only the most recent prediction is updated.

$$\Delta w_t = \alpha (P_{t+1} - P_t) \; \nabla_w P_t$$

In the following section, we'll replicate three experiments to study the property of TD methods, using 100 training datasets where each dataset has 10 sequences of the random walk simulation.

3 EXPERIMENT 1

3.1 Implementation

The first experiment is to compare the performance of TD methods with different lambda. This experiment is based on *Repeated Presentation* of each training dataset. For each training dataset, TD method is used to learn from the 10 sequences, and the weight is updated after showing all the 10 sequences. This procedure is performed repeatedly until converged (no significant change in w). Then for each training dataset, the error (RMSE) can be calculated compared to the true probabilities (⅙, ⅓, ½, ⅔, and ⅚ for state B, C, D, E, and F). The same process is performed across 100 training datasets and the average error rate is calculated.

3.2 Replication Result

The Figure 3 replication has the same shape as the original figure 3 in Sutton's paper, where TD(1) performs the worst, TD(0) performs the best, and the error rate increases as λ increases.

One difference is that the error rates are ~30% lower than Sutton's Figure 3. One hypothesis is that we benefit from stronger computing power in the repeated presentation process. When showing one training data repeatedly until the weight converged, my criteria of convergence is that the sum of absolute Δw_t is less than 0.001. This generally translates to hundreds of iterations in each training dataset. Sutton didn't mention the convergence criteria he used but we have reason to assume that due to the constraints of computing power back in 1988 the convergence criteria might be looser, so the error rates are higher than our replication today.

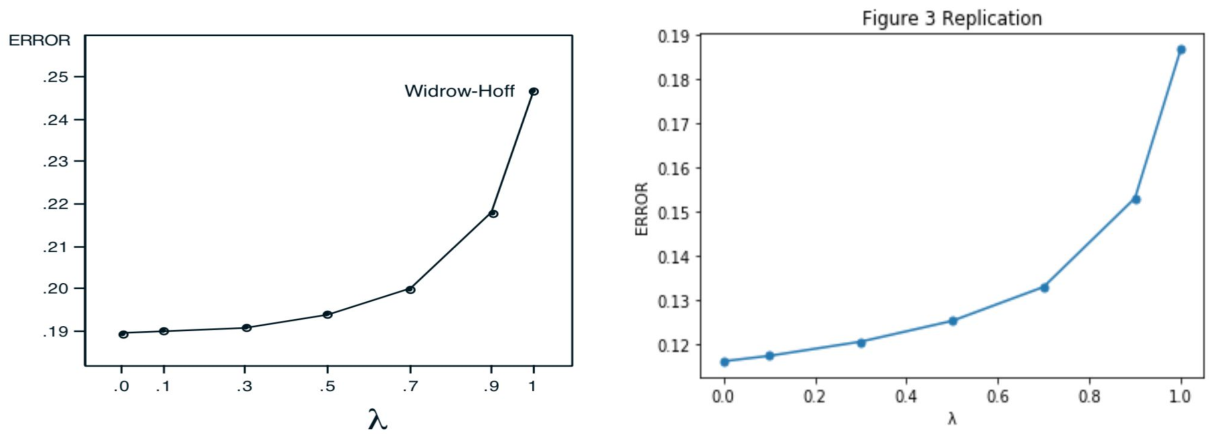


Figure 1 — Sutton's Figure 3 and the replication.

3.3 Analysis

It's interesting that TD methods perform the best at λ=0 and then get worse as λ increases. My intuition is that TD(1) converges to a solution to minimize MSE in the training dataset, and it sounds like 'overfitting' the training data instead of learning the hidden patterns, while TD(0) tries to estimate the underlying Markov process and makes better use of the experience.

Since it needs many steps until the termination state is reached, TD(1) suffers from the randomness introduced by those steps and has relatively high variance, while TD(0) only uses the successive prediction as the target and so has lower variance but higher bias. However in the repeated presentation setting, the bias can be well reduced by showing and learning the data repeatedly until converged, thus it results in better TD(0) performance.

4 EXPERIMENT 2

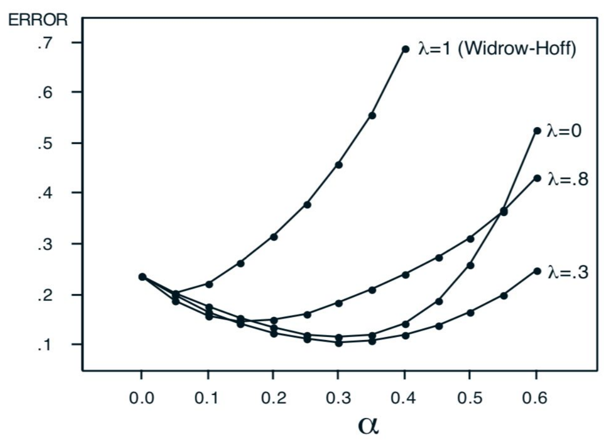
4.1 Implementation

Experiment 2 studies how the learning rate alpha affects the performance of different TD methods. We tested TD(0), TD(0.3), TD(0.8) and TD(1) with learning rates ranging from 0 to 0.6. One difference from the experiment 1 is that, instead of repeated presentation, single presentation is applied here, where each training data set is only learned once and the weight is updated immediately after each sequence is learned.

4.2 Replication Result

We are able to match the trends of Figure 4 in Sutton's paper, where TD(1) has the highest error across learning rate, TD(0) looks competitive with small learning rate however at greater learning rate, the performance is worse than TD(0.3) and TD(0.8) but still better than TD(1).

The error rates don't match exactly to Sutton's paper, with the hypothesis being that, due to the single presentation training, it's far from convergence, so the result is likely to vary across the training datasets. This hypothesis is supported by the fact that, trying out different random seed results in different error rates, however, the general trends still hold across training datasets.



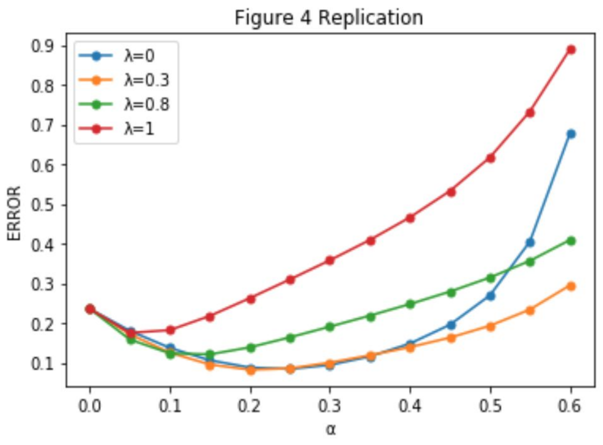


Figure 2 —Sutton's Figure 4 and the replication.

4.3 Analysis

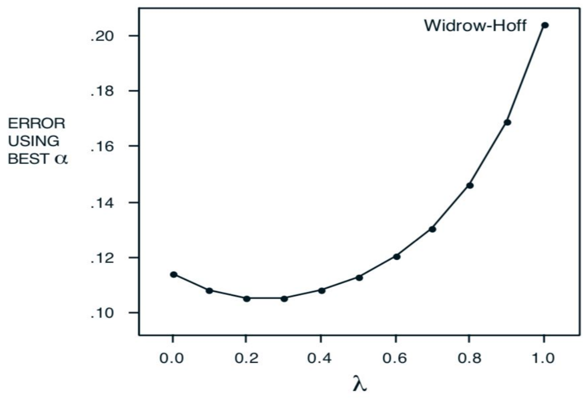
It's interesting that as alpha increases, the error rate decreases and then increases. It can be explained by the equation $\Delta w_t = \alpha (P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k$, where if alpha is too small, the Δw_t only steps a little towards the right direction, and when alpha is too large, it's likely that the update will overshoot and never converges to the optimal w. The optimal alpha should sit somewhere in between. This can be looked side by side with the fact that the best learning rate is smaller for TD methods with large λ. This is because they propagate back more intensively and when the alpha increases, it's easier for them to overshoot and get worse performance.

5 EXPERIMENT 3

5.1 Implementation

Experiment 3 studies the performance of TD methods using the best learning rate for each λ. We obtained the best learning rate for each TD(λ) algorithm by a similar process in experiment 2.

5.2 Replication Result



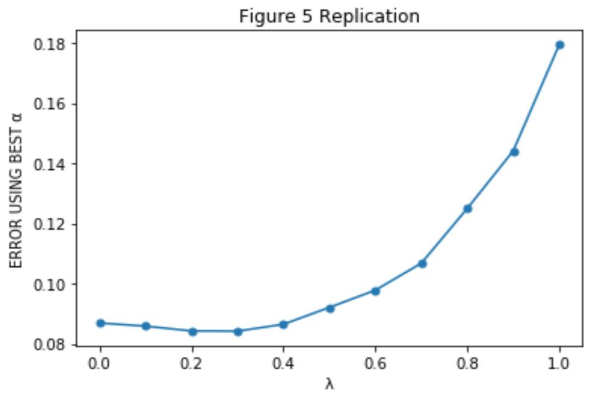


Figure 3 — Sutton's Figure 5 and the replication.

The shape of the line and the best performer matches Figure 5 in Sutton's paper. TD(0) performs much better than TD(1) while the best result comes from TD(0.3). As alpha increases, the error rate decreases from TD(0) to TD(0.3) and then increases from TD(0.4) to TD(1).

Similar to experiment 2, the error rate doesn’t match exactly, and the hypothesis being that the result varies slightly based on the training datasets we randomly generated.

5.3 Analysis

It’s interesting that in the single presentation context, TD(0.3) performs better than TD(0). It can be explained by bias and variance trade-off: when λ increases, variance increases and bias decreases, and it looks like λ = 0.3 is a sweet spot to balance the two errors. At TD(0), since it only updates the most recent prediction based on the successive prediction, it is very slow to back propagate the correct values. In experiment 1 it won’t be a problem as the training set is shown repeatedly for TD(0) to propagate the values. However in single presentation, it is far from convergence, so the drawback of TD(0), slow learning, reveals.

6 PITFALLS AND LEARNING

6.1 Divergence with large α value in repeated presentation

In experiment 1, I was trapped in the situation where the repeated presentation process for some training sets runs forever. Then I realized that since the Δw is cumulated for 10 sequences and updated at one time together, so, when the learning rate α is not small enough, the w update process will easily overshoot and never converge to the optimal w. The solution I adopted is to use a smaller α.

6.2 High error rates with large α value in single presentation

At the beginning of experiment 2, my error rates are higher than Sutton’s when α value is high. Then I found out that it happens especially when the sequence is long and the walk visits some states multiple times. Learning such a sequence is problematic since it will lead the weight update process to focus too much on those states and result in an abnormal w. I came up with a potential solution to alleviate the issue: introduce a sequence length threshold and only learn the sequences whose lengths are smaller than the threshold. This method is able to produce comparable results to Sutton’s paper.

7 REFERENCES

[SB20] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. 2nd Ed. MIT press, 2020. url: http://incompleteideas.net/book/the-book-2nd.html.

[Sut88] Richard Sutton. “Learning to Predict by the Method of Temporal Differences”. In: Machine Learning 3 (Aug. 1988), pp. 9–44.