Abstract

TD Lambda is a temporal-difference based learning algorithm invented by Richard S. Sutton. The novelty of TD Lambda laid in its capability of incremental learning through time, whereas conventional prediction-learning algorithms of the time learns only after final outcome is seen. In Sutton's paper "Learning to Predict by the Methods of Temporal Difference", he proved the convergence and optimality of TD Lambda and argued TD-Lambda is a better solution to most of the prediction problems of that time. In this experiment, results in Sutton's original paper are replicated. It is shown that the results in Sutton's 1988 paper are generally reproducible, with minor explainable discrepancies. It is shown that when repeated presentations training paradigm is used, TD (0) had the best performance, while TD (1) had the worst performance; under single presentation, TD with $\lambda < 1$ almost always out-performed TD (1) at all α between [0,0.6].

1. Introduction

Temporal-difference (TD) methods are a class of prediction algorithm that learns, incrementally through time steps, to predict outcome of multi-step prediction problem (which is a prediction problem where the correctness of the prediction is not revealed until more than one step after the prediction is made, but partial information relevant to its correctness is available at each step). In a multi-step prediction problem, conventional prediction algorithms make one prediction about the problem and learn from the error between the prediction and the final outcome. TD methods, on the other hand, make predictions through time steps and learns from errors between temporally successive predictions. One such problem, given in Sutton's paper, is the weather forecast problem, where one needs to predict weather on Saturday prior to Saturday. In this example, conventional prediction algorithms, under the supervised learning paradigm, tackle the problem by learning to associate individual observation and outcome pairs. In contrast, TD methods learn from a sequence of observations through time steps (from Monday to Friday for example) by making prediction to the final outcome at each step and learning from errors between temporally successive predictions.

The next section introduces the TD-lambda method invented by Sutton and derives some key equations. Section 3 explains Sutton's convergence and alpha optimality experiments and illustrate results published in his paper. Section 4 discusses the replication procedure and results. Lastly section 5 discusses obstacles encountered during the replication and takeaways from this exercise.

2. Methods review

In a multi-step prediction problem, observation-outcome sequences are seen by the learner of the form $x_1, x_2, x_3, \dots, x_m, z$ where each x_t is a vector of observations available at time t, and z is the outcome of the sequence. The intuition behind the algorithm is that by going through the observation-outcome sequences, the algorithm tries to learn a weight vector w. The dot product of w and the x_t vector gives the prediction of the sequence outcome for time t. The learning procedure is thus a process of updating w with small increments through time after an observation-outcome sequence, given by Equation 1. Assuming w is only updated at the end of each observation-outcome sequence, for each time step (i.e., each observation), an increment to w, denoted Δw_t, is determined and accumulated. At the end of the sequence, the accumulated Δw_t is added to the probability weight w.

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

The core to the algorithm is then finding the weight update vector Δw_t, which is described by Equation 2,

$$\Delta w_t = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k \qquad \textit{Equation 2}$$

where $P_t(w, x_t) = w^T x_t$ is the prediction for z at time t, $\nabla_w P_k$ is the gradient of function P_t, which is the partial derivatives of P_t with respect to w. Since P_t is a linear function of x_t and w, $\nabla_w P_k = x_t$. λ^{t-k} is an exponential weighting with recency, in which prediction errors between two successive predictions in each past k steps, $(P_{t+1} - P_t)$, are weighted according to λ^k for $0 \leq \lambda \leq 1$, and α is the learning rate that controls the step size of the weight update Δw_t.

Notice the past prediction error $(P_{t+1} - P_t)$ is weighted by the term $\sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k$. Assume e_t is the value of $\sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k$ for t,

$$e_t = \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k \qquad \textit{Equation 3}$$

$$e_t = \nabla_w P_t + \sum_{k=1}^{t-1} \lambda * \lambda^{t-1-k} \nabla_w P_k \qquad \textit{Equation 4}$$

$$e_t = \nabla_w P_t + \lambda e_{t-1} \qquad \textit{Equation 5}$$

Where at t = 1, $e_0 = 0$, $e_1 = \nabla_w P_1 = x_t$.

Equation 2 then becomes,

$$\Delta w_t = \alpha(P_{t+1} - P_t)e_t \quad \text{or} \quad \Delta w_t = \alpha(P_{t+1} - P_t)(x_t + \lambda e_{t-1}) \qquad \textit{Equation 6}$$

It is important to observe that when $\lambda = 1$, Equation 2 becomes

$$\Delta w_t = \alpha(P_{t+1} - P_t)\sum_{k=1}^{t} \nabla_w P_k \qquad \textit{Equation 7}$$

which is the TD implementation of the Widrow-Hoff rule (Widrow & Hoff, 1960), and which leads to Sutton's Theorem 1:

On multi-step prediction problems, the linear TD (1) procedure produces the same per-sequence weight changes as the Widrow-Hoff procedure.

For $0 < \lambda < 1$, TD (lambda) will weight all previous prediction errors differently, where higher weights are assigned to the most recent prediction errors and gradually decreased weights are assigned to earlier prediction errors. As λ decreases, weights put on recent prediction errors increase, while on past prediction errors decrease. As $\lambda = 0$, all weight will be assigned to the most recent prediction error, and Equation 2 becomes

$$\Delta w_t = \alpha(P_{t+1} - P_t)\nabla_w P_t \qquad \textit{Equation 8}$$

which is very similar to the prototypical supervised-learning update procedure $\Delta w_t = \alpha(z - P_t)\nabla_w P_t$, with the only difference being a different error is used.

3. Experiment design

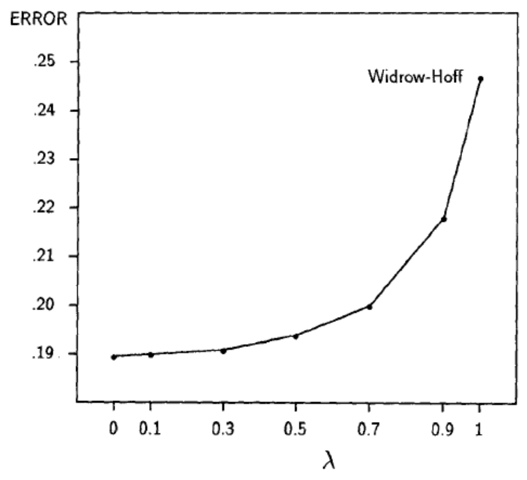
In his paper, Sutton designed a simple bounded random walk dynamic system to prove his claim that under a dynamic system, TD methods make more efficient use of their experience than do supervised-learning methods, that they converge more rapidly and make more accurate predictions along the way. In his bounded random walk dynamic system, there are total 7 states namely {A, B, C, D, E, F, G}. Every walk begins in the center state D. At each step the walk moves to a neighboring state, either left or right with equal probability. States A and G are the terminate states, if the walk ever reach any of these two states, the walk terminates. A walk's outcome was defined to be z=0 for a walk ending on the left at A and z=1 for a walk ending on the right at G. TD (Lambda) is tasked to estimate z at each state, that is, the probability of ending at G at each state in {B, C, D, E, F}. A typical walk will look like, for instance, DEFEFG. The resulted observation-outcome sequence will then be $x_D, x_E, x_F, x_E, x_F, x_G, 1$, where each x_i is a unit basis vector of length 5, with a 1 at the index of the current state and 0s for other states, for example $x_D = [0, 0, 1, 0, 0]$.

Two experiments were performed using observation-outcome sequences generated as described. The first experiment, under repeated presentation training paradigm, concerns with the prediction error of different λ, while the second experiment concerns with the learning rate for different lambda when training set is presented just once.

3.1. Experiment 1

In the first experiment, 100 training sets, each consisting of 10 observation-outcome sequences, were constructed for training the TD (λ). Weight increments were computed according to equation x, and seven different values are used for λ. For this experiment, Δw was accumulated for a full training set, 10 sequences in other words, before weight vector is updated. Each training set was repeatedly presented to TD (λ), until convergence of the w, and then RMS error between w and the true probability (1/6, 2/6, 3/6, 4/6, 5/6) is calculated. RMSE is averaged over the 100 training sets for each λ. Figure 1 below is the result of this experiment obtained by Sutton.

Figure 1: RMSE for different values of λ after repeated presentations. (Sutton, 1988)



The result showed that the Widrow-Hoff procedure had the highest RMS error among other TD methods with $\lambda < 1$. In this experiment, Sutton also claimed that for small α the weight vector always converged to the same final value, independent of its initial value.

3.2. Experiment 2

In experiment 2, the same 100 training sets are used for learning procedures. However, each training set was presented only once to each procedure. In addition, weight updates were performed after each observation-outcome sequence as in equation x and weight vector w were initiated to 0.5 for all nonterminal states. Figure x below shows prediction error by learning rates for each TD (1), TD (0), TD (.8) and TD (.3). Figure 2 shows the average error at best α found for each TD (lambda).

Figure 2: RMSE at different learning rate for different TD (λ). (Sutton, 1988)

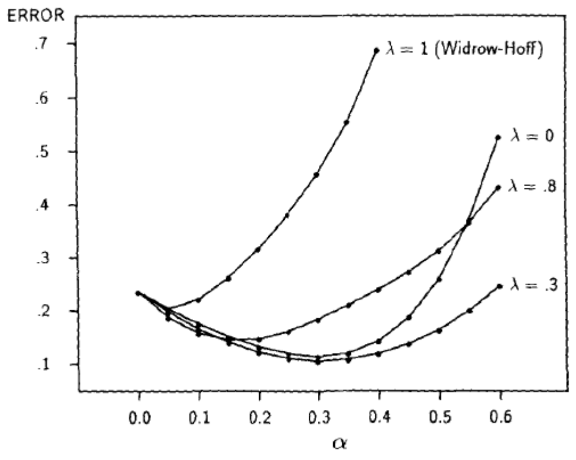


Figure 3: RMSE of TD (λ) with optimum learning rate. (Sutton, 1988)

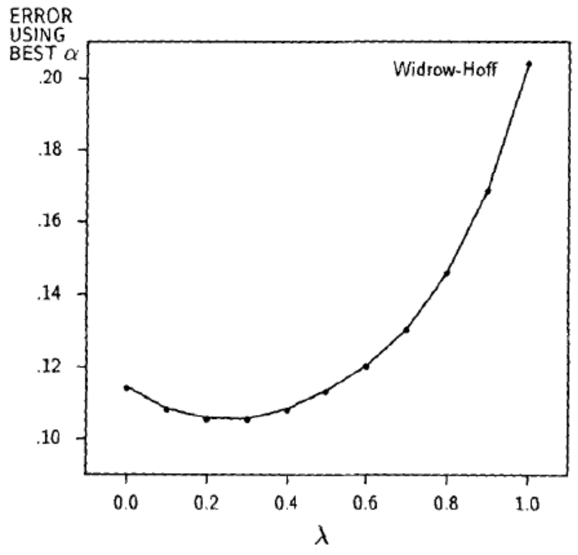


Figure 3 shows that for all α values TD methods with $\lambda < 1$ are always superior than the Widrow-Hoff procedure, TD (1). From figure x, it is observed that the best λ is around 0.3, as oppose to 0 in experiment 1, when the best α for each λ is used.

4. Replication

4.1. Procedure

The two experiments in Sutton's paper are replicated in Python environment. This section discusses procedures taken in the replication and presents the replicated results.

4.1.1. Training set

The first step in replicating Sutton's experiments is to construct the training sets. A python class named *Sequence* is created to perform this task. The *get_data* method in the class takes *epochs* (number of training sets) and *episodes* (number of sequences in each training set) as inputs, and outputs a nested lists *train_samples*, where length of the *train_samples* is specified by *epochs*, each element of the *train_samples* is a list of the length specified by *episodes*. Each element of this inner list is a tuple contains 3 elements, the state matrix, outcome, and letter path taken, that describe a complete observation-outcome sequence.

For both experiments, 100 epochs and 10 episodes are specified when calling *Sequence.get_data*. In total 1000 sequences are generated.

4.1.2. Experiment 1

Pseudo code for experiment 1

```
Initialize tolerance with a small number
Initialize $RMSE_\lambda$ as empty dictionary
Initialize $\alpha = 0.01$
Initialize $true_{prob} = [\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}]$
for $\lambda$ in [0, 0.1, 0.3, 0.5, 0.7, 0.9, 1] do
    Initialize $RMSE_{epoch}$ as empty list
    for epoch in 1, 100 do
        initialize w with random weights
        initialize $delta$ with a large number
        while $delta$ > tolerance do
            $w_{t-1} = w$
            initialize $\Delta w$ with 0s
            for episode in 1, 10 do
                initialize $e_{t-1}$ with 0
                for state in states do
                    $\begin{cases} P_t = w^T x_t, & if\ frist\ state \\ P_{t+1} = reward, if\ last\ state \\ P_{t+1} = w^T x_{t+1}, other\ states \end{cases}$
                    $e_t = x_t^T + \lambda e_{t-1}$
                    $\Delta w = \alpha(P_{t+1} - P_t) e_t$
                    $e_{t-1} = e_t$
                    $P_t = P_{t+1}$
                end for
            end for
            $w = w + \Delta w$
            $delta = RMSE(w, wpast)$
        $RMSE_{epoch}$ append $RMSE(w, true\_prob)$
    end for
    $RMSE_\lambda = Mean(RMSE_{epoch})$
end for
```

4.1.3. Experiment 2

Pseudo code for experiment 2a

```
Initialize tolerance with a small number
Initialize $RMSE_{\alpha,\lambda}$ as empty dictionary
Initialize $true_{prob} = [\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}]$
for α in 0 -> 0.6, step=0.05
    for $\lambda$ in [0, 0.3, 0.8, 1] do
    Initialize $RMSE_{epoch}$ as empty list
    for epoch in 1, 100 do
        initialize w with 0.5
        for episode in 1, 10 do
            initialize $\Delta w$ with 0s
            initialize $e_{t-1}$with 0
            for state in states do
                $\begin{cases} P_t = w^T x_t, & if\ frist\ state \\ P_{t+1} = reward, if\ last\ state \\ P_{t+1} = w^T x_{t+1}, other\ states \end{cases}$
                $e_t = x_t^T + \lambda e_{t-1}$
                $\Delta w = \alpha(P_{t+1} - P_t)e_t$
                $e_{t-1} = e_t$
                $P_t = P_{t+1}$
            end for
            $w = w + \Delta w$
        end for
        $RMSE_{epoch}$ append $RMSE(w, true\_prob)$
    end for
    $RMSE_{\alpha,\lambda} = Mean(RMSE_{epoch})$
end for
```

Pseudo code for experiment 2b

```
Initialize tolerance with a small number
Initialize $RMSE_{\lambda,\alpha}$ as empty dictionary
Initialize $true_{prob} = [\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}]$
For $\lambda$ in 0 -> 1, step=0.1
    for α in 0 -> 0.6, step=0.05 do
    Initialize $RMSE_{epoch}$ as empty list
    for epoch in 1, 100 do
        initialize w with 0.5
        for episode in 1, 10 do
            initialize $\Delta w$ with 0s
            initialize $e_{t-1}$with 0
            for state in states do
                $\begin{cases} P_t = w^T x_t, & if\ frist\ state \\ P_{t+1} = reward, if\ last\ state \\ P_{t+1} = w^T x_{t+1}, other\ states \end{cases}$
                $e_t = x_t^T + \lambda e_{t-1}$
                $\Delta w = \alpha(P_{t+1} - P_t)e_t$
                $e_{t-1} = e_t$
                $P_t = P_{t+1}$
            end for
            $w = w + \Delta w$
        end for
        $RMSE_{epoch}$ append $RMSE(w, true\_prob)$
    end for
    $RMSE_{\lambda,\alpha} = Mean(RMSE_{epoch})$
end for
```

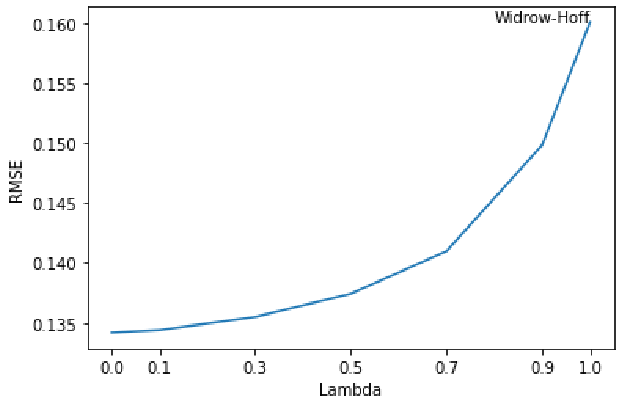
4.2. Results & Discussion

This section illustrates replication results and discusses some key observations.

4.2.1. Experiment 1

Figure 4 shows the replicated experiment 1 result.

Figure 4: Replicated RMSE for different values of λ after repeated presentations.

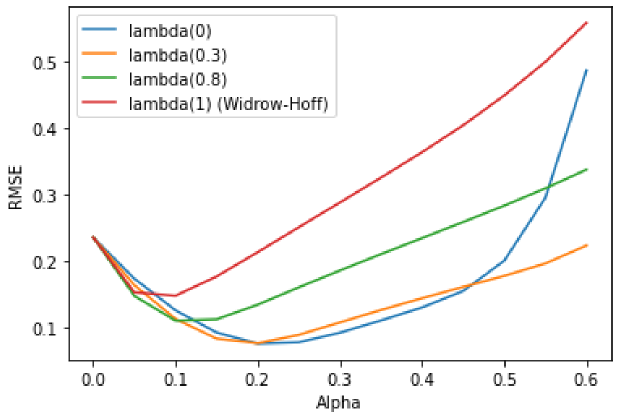


It is observed that the trend of the RMSE by Lambda plot generally aligned with that in Figure 1, with TD (0) resulted in the lowest RMSE, while TD (1) resulted in the highest RMSE. However, the magnitude of the RMSE for each Lambda in the replicated result is generally lower than that in the paper. This discrepancy can attribute to three main reasons. First, in the paper, Sutton did not specify the exact α he had used. In the replication, an assumption of $\alpha = 0.01$ is made, which could be well different then the one used by Sutton, and in turn causes the difference. Second, in the first experiment, repeated presentations were performed until convergence. However, Sutton did not provide specific convergence criteria, other than "...until the procedure no longer produced any significant changes in the weight vector." During the replication, the convergence criteria is set to: RMSE between current w and previous w less than 1e-7. Lastly, randomness in the generate sequences in the training sets could cause discrepancies between results.

4.2.2. Experiment 2a

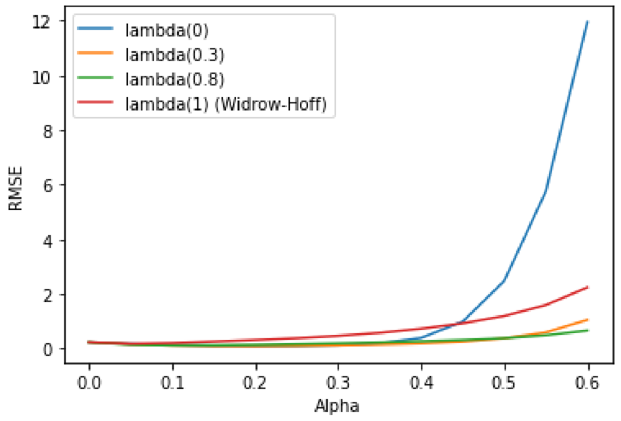
Figure 5 shows the replicated RMSE by α plot for four different λ values, which mostly aligned with results in Figure 2.

Figure 5: Replicated RMSE at different learning rate for different TD (λ).



The replicated result confirmed Sutton's conclusion from Figure 2 that all TD methods with $\lambda < 1$ performed better than the TD (1) method. Minor discrepancies in RMSE at different α values are again observed. These discrepancies can again due to randomness in the sequences generated. However, in order to replicate the result of experiment 2 in Figure 5, a major assumption has to be made. That is, the length of all the randomly generated sequences in the training sets are less or equal to 15 steps. Figure 6 below shows the same plot with no limit on the length of the generated sequences.

Figure 6: RMSE at different learning rate for different TD (λ) without limiting length of the randomly generated training sequences.

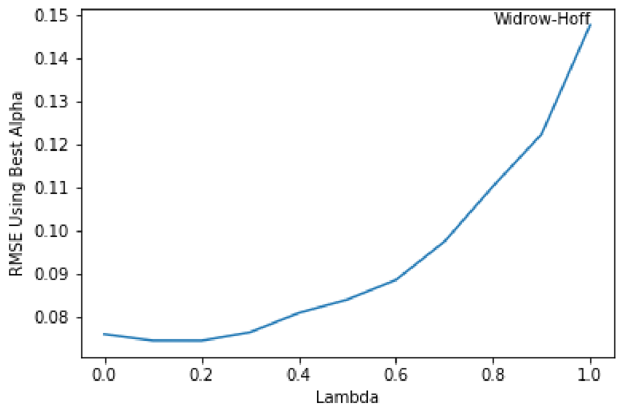


Based on the plot above, non of the observations made previously remained true. The reason for the totally different result for experiment 2 when not limiting sequence length is believed to be with the eligibility e_t. One of the objectives for e_t is to keep track of number of times a state is visited by the learner, so that Δw can be updated accordingly with the right amount of errors. When sequences are too long, e_t will be generally larger, which in turn results larger Δw. Since in experiment 2, each train set is only shown once, the learner might not converge well with a large Δw.

4.2.3. Experiment 2b

Finally, Figure 7 shows the RMSE for each λ when the optimum α is used.

Figure 7: Replicated RMSE of TD (λ) with optimum learning rate.



The replicated result generally aligned with published result shown in Figure 3. The best λ, however, occurred at around 0.2 in the replication, instead of 0.3 in the paper; and the generally lowered RMSE is again observed. Both of these discrepancies are thought to be caused by the randomness in the sequences generated.

5. Conclusion

In this work, Richard Sutton's 1988 paper "Learning to Predict by the Methods of Temporal Differences." is replicated. By doing the replication, the concepts of TD (λ) are studied in detail. It is shown that results published in the paper is generally reproducible. Some small discrepancies were noticed in the replication. These discrepancies are thought to be attributable to the randomness in the generated training sets, unspecified values for certain parameters and potentially unspecified rules when creating training sets (i.e., a limit on length of generated sequences).

References

Sutton, R. (1988, August). Learning to predict by the methods of temporal differences. *Mach Learn 3*, pp. 9-44.