Examining the performance Temporal Difference learning procedures: A replication of Sutton (1988)

Anh Duc Vu
College of Computing
Georgia Institute of Technology
avu41@gatech.edu

***Abstract*—This paper reexamines Sutton's (1988) study on temporal-differences methods' learning performance by replicating his findings. Sutton stated that temporal-differences methods can produce better predictions with less memory and peak computation compared to supervised learning methods. The author illustrated his statements through two computational experiments, showing that temporal-differences learning procedure outperform supervised learning procedure in learning to predict in both cases. The replications examine the performance of temporal-differences methods in prediction learning when using different data set and key parameters. Results indicates that temporal-difference learning procedures indeed learn faster than supervised learning, outperforming it in producing predictions.**

***Index Terms*—temporal difference learning**

I. Introduction

A. Supervised learning and temporal-difference learning

The prediction learning problem is defined as using observed data from a known or unknown model to generate prediction that best matches the actual outcome. While supervised learning methods and temporal-difference (TD) learning method both generate prediction using errors, the errors computed in the two procedures are different. Because supervised learning methods is the process of associating pairs of observed data and the actual outcome, the errors are calculated using the predicted outcome and the actual outcome. On the other hand, in TD methods, which is formally introduced by Sutton (1988), errors are calculated using successive predictions, also known as TD errors. According to Sutton (1988), TD methods have two main advantages over supervised learning methods. First, the calculation to generate prediction is incremental, and thus is easier to compute. Second, TD methods make better use of the observed data. We will further examine these arguments in the next subsection.

B. Mathematical formulation of temporal-difference learning

First, it is important to separate the two main types of prediction-learning problems: The single-step and the multi-step problem. Single-step problem is defined as problems where all information about the outcome is revealed after one step. By contrast, in multi-step problem, all information about the outcome is only revealed after multiple steps. Because TD methods learn to predict by using the errors between temporally successive prediction, in single-step problem, TD methods and supervised learning methods perform equally. As a result, we only examine TD methods and supervised learning methods' performance in the context of multi-step problems.

In multi-step problem, we define the observation sequences as $x_1, x_2, x_3, ..., z$, where x_t is a column vector, representing the observation at time step t, and z is a scalar value, representing the actual outcome at the final step. Similarly, we define $P_1, P_2, P_3, ..., P_m$ as the sequence of predictions made by the learning procedure, where P_t is a column vector, representing the prediction made at time step t, and P_m is the vector prediction made at the final step. By definition, $P_m = z$. Prediction is calculated as a function of the weight vector, w, and the observation at time t, x_t. Because prediction is a function of the observation and the weight vector, learning is expressed as an update in the weight vector, Δw_t, after each observation. Assuming that Δw_t is accumulated over observations, and the weight vector is only updated at the end of each observed sequence, we then have:

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

In supervised learning methods, errors are calculated using the difference between the actual and predicted outcomes. Hence, the formula to update the weight vector is:

$$\Delta w = \alpha(z - P_t)\nabla w P_t \tag{2}$$

where α is the learning rate, and $\nabla w P_t$ is the vector partial derivative of P_t with respect to w.

Note that the error, $z - P_t$, can be expressed as the summation of changes in predictions:

$$z - P_t = \sum_{k=t}^{m} (P_{k+1} - P_k) \tag{3}$$

Using the equation above, combining with equation (1), equation (2) can be expressed as:

$$\Delta w_t = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \nabla w P_k \tag{4}$$

The learning procedure in (4) is formally known as TD(1) procedure. A signature in TD learning methods, $TD(\lambda)$, is the consideration of sequential steps in generating prediction. Specifically, in $TD(\lambda)$, the eligibility trace vector is decayed by λ after every observation. This method of decaying the eligibility trace vector helps keeping check of which step is the most recent observation. The further the observation is in the past, the less influence it has in the updating of the weight vector. Thus, the weight vector update for $TD(\lambda)$ is formally expressed as:

$$\Delta w_t = \alpha(P_{t+1} - P_t)\sum_{k=1}^{t} \lambda_{t-k} \nabla w P_k \tag{5}$$

Note that $e_t = \sum_{k=1}^{t} \lambda_{t-k} \nabla w P_k$ is the eligibility trace vector. An important advantage of the exponential form in the eligibility trace vector is that it can be computed incrementally. Specifically, we have:

$$e_{t+1} = \nabla w P_{t+1} + e_t \tag{6}$$

C. Temporal differences learning advantages

From equation (5) and (6), we can see that the updating of weight vector for $TD(\lambda)$ learning procedure can be calculated incrementally after each observation, and thus is easier to compute. This is a significant advantage compared to the supervised learning method in (2), where z is only known at the end of the sequence. In supervised lealrning method, all observations must be remembered until the final step, where all Δw_t is computed at the same time, causing peak computation, and thus also require more memory than $TD(\lambda)$.

With the eligibility trace vector, $TD(\lambda)$ also makes more efficient use of observed data by taking into account the order of the observation when generating predictions. This is also an advantage compared to supervised learning procedures. In multi-step problems, because errors are only calculated at the end of the sequence using the actual outcome, supervised learning methods ignore the order structure of the observations, while TD methods leverage this information to produce better predictions.

In the next section, we will setup two computational experiments to replicate Sutton's finding and to examine the performance of $TD(\lambda)$ and supervised learning procedures in prediction learning.

II. Setting up experiments

A. A bounded random-walk

With the advantage mentioned in the last section, Sutton argued that $TD(\lambda)$ should learn better than supervised learning in any multi-step problems, where the outcome is only partially observed after each time step. The author set up a simple dynamical system, a bounded random walk, to illustrate his argument.

A sequence of state from A to G is set up for the experiment. The agent always start at state D, and from there has an equal probability of moving either left or right. The sequence ends when the agent enters either state A or G, where it will get a reward of 0 or 1, respectively. The learning method estimates the expected reward at each state, in other words, the probability of a right-side termination. Using conditional probability, we can set up 5 system linear equations to solve for the expected value of each state. Doing so, we have the ideal predictions at step B, C, D, E, F as $\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}$, respectively.

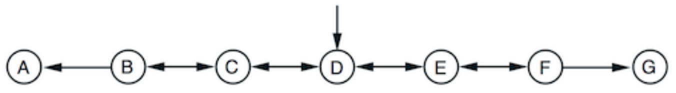

Fig. 1. A bounded random walk. Agent always start at state D. (Sutton, 1988)

Using a linear function approximator, $P_t = w^T x_t$, the procedure $TD(1)$ in equation (4) is reduced to the supervised learning Widrow-Hoff algorithm, $\Delta w = \alpha(z - P_t)x_t$. In the experiments, we compare the learning performance of the supervised learning method Widrow-Hoff, represented by $TD(1)$, with other TD methods for $0 \leq \lambda < 1$.

To guarantee statistically reliable results, 100 training sets, each contain 10 sequences, are used by all learning procedures to generate prediction. The weight updated after each observation is calculated using equation (5). A range of values of λ are used to generate predictions, where $\lambda = 1$ represent the supervised learning procedure Widrow-Hoff, and $0 \leq \lambda < 1$ represent the TD learning procedure.

B. Learning performance in repeated training presentation

In the first experiment, each learning procedures is repeatedly present with a training set until convergence of the weight vector. In addition, instead of updating the weight vector after every sequence, Δw_t is accumulated over sequences, and the weight vector is updated at the end of each training set. Sutton refers to this method as the repeated presentations training paradigm. For each learning procedure, the root mean square error (RMSE) is calculated using the ideal predictions and those found by the learning procedure. Averaging over 100 training sets for the RMSE of each learning procedure, we replicate figure 3 of Sutton's paper to examine the learning performance as λ increase from 0 to 1. Note that for this setup, all learning procedures use the same learning rate.

C. Learning rate when training set is presented once

The goal of the second experiment is to examine the effect of learning rate in learning performance. Each learning procedure is only present with the training set once. Also, the weight vector is updated after each sequence, rather than after each training set. We repeat this procedure for every learning procedures, with a range of values for the learning rate, α. With the RMSE calculated after each procedure, we replicate figure 4 of Sutton's paper to examine the effect of learning rate on learning performance for different values of λ.

Finally, using the learning rate which produce the best predictions, we plot the RMSE for each $TD(\lambda)$ learning procedure, replicating figure 5 of Sutton's paper. This figure examines the best errors achieved for each values of λ.

III. Results and analyses

A. Repeated training presentation paradigm implementation and result

We followed the procedure laid out by Sutton, and created a training set with 10 sequences for the random walk. Note that, because the agent always start at state D, with en equal probability of going either left or right, the length of each sequence varies. In the experiment, we do not attempt to control the distribution of right-termination and left-termination sequences. All learning procedures are repeatedly presented with the training set, with the weight vector initialized as $[0, 0, 0, 0, 0]^T$. The change in weight vector is accumulated after each sequence, and the weight vector is only updated at the end of the training set. The RMSE is averaged over the number of training sets.

Two main assumptions we made for the replication are regarding the learning rate and the number of training set presented to each learning procedures. Because Sutton did not explicitly mentioned the learning rate he used, only stating that the weight vector always converge for a small learning rate, we used $\alpha = 0.01$ for our replication. We will explore the effect of other learning rates in the analysis section. Sutton also set up the experiment such that each training set was presented repeatedly until the procedure no longer produced any significant changes in the weight vector. However, as the author did not mention his criteria for convergence, in our replication, we only presented each learning procedures with 100 training sets. We will further explore how the results change with a higher number of training sets.

Following the procedure laid out above, we attain figure 2, as a replication of Sutton's figure 3.

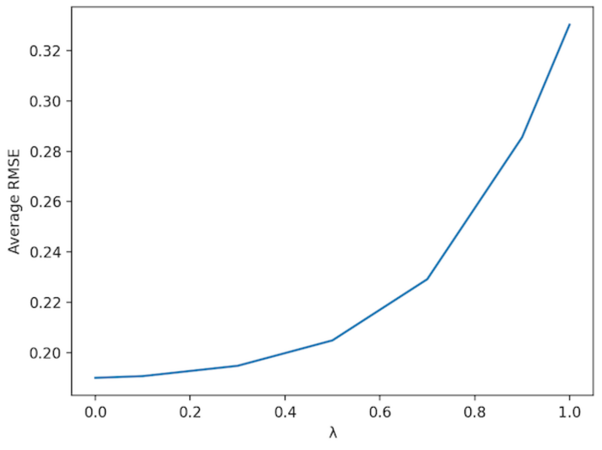

Fig. 2. Replication of figure 3 in Sutton (1988)

B. Repeated training presentation paradigm analyses

Our replication closely match with Sutton's result. The TD learning procedure perform best at $TD(0)$, and rapidly got worse as λ increases to 1. Since $TD(1)$ represent the Widrow-Hoff learning procedure, our experiment shows that supervised learning method indeed performs worse than TD learning method when $0 \leq \lambda < 1$. Intuitively, this makes sense when we consider that Widrow-Hoff procedure only aim to minimize the error for training experience, not for future experience, hence over fitting the training data. The main difference between our results and Sutton's is the range of RMSE. While the author's error range is from 0.19 to 0.25, ours is from 0.20 to 0.32. The difference is due to stochastic differences in generating data use for the experiment. In other words, we used a different data set from Sutton's. This can be confirmed, since we obtain a different range of RMSE when we used a different data set to generate figure 4.

Note that, for the data set used for figure 2, the errors did not change even when we increase the number of training sets to over 100. We can then conclude that, for this particular data set and learning rate, 100 training sets were enough for convergence. Experimenting with a different data set, we found that this is not always the case, as showing in the figure 3. It's interesting to note that, even before the weight vector converges, the Widrow-Hoff procedure already perform worse than other TD learning procure. In addition, note that $TD(0)$ performed worse with the same number of training set compared to other other $TD(\lambda)$ with $\lambda < 1$. This is because $\lambda = 0$ is slow at propagating predictions back along the sequence. We will further examine this phenomenon in the next experiment.

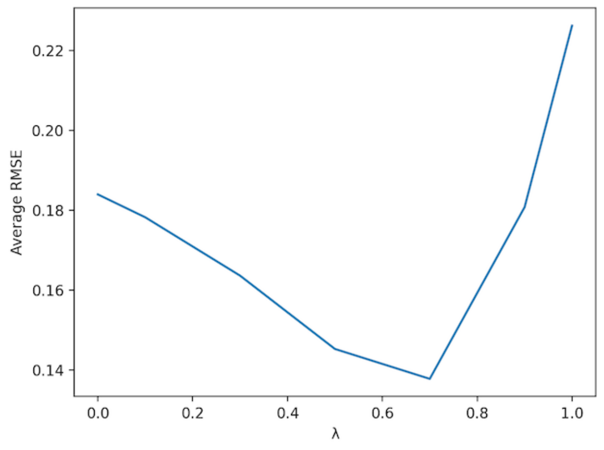

Fig. 3. Average error on the random-walk problem under repeated presentations (before the weight vector converges)

As we increase the number of training sets, the weight vector converged, confirming our expectation that Widrow-Hoff procedure perform worse as shown in figure 4. Note that, the learning rate also affect the number of training sets needed for the weight vector to converge. There is a trade-off between the learning rate and the number of training sets needed for convergence. Specifically, for a higher learning rate, we found that the TD learning procedure needs less training set for the weight vector to converge. However, when the learning rate passes a certain threshold, which is different depending on the training set, the weight vector will blow up and no longer be able to converge. Due to the fact that the weight vector is only updated after 10 sequences, a high learning rate will make the Δw_t learn too fast, thus keep accumulating with values that will pass the actual predictions. After one update, if the weight vector are already higher than the prediction, the weight vector will keep accumulating and blow up after every training set. We will further examine the effect of learning on TD learning procedure in the next experiment.

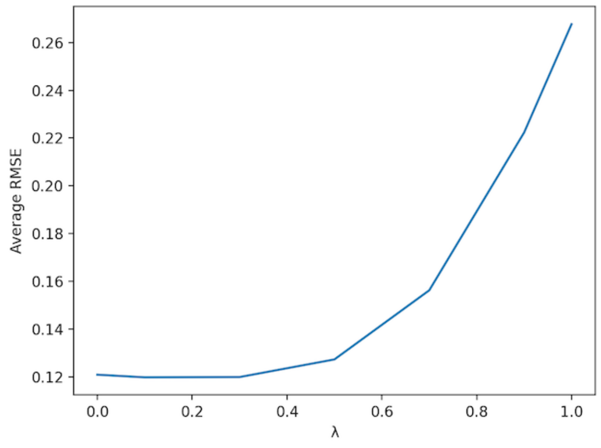

Fig. 4. Average error on the random-walk problem under repeated presentations (after the weight vector converges)

C. Learning rate experiment implementation and results

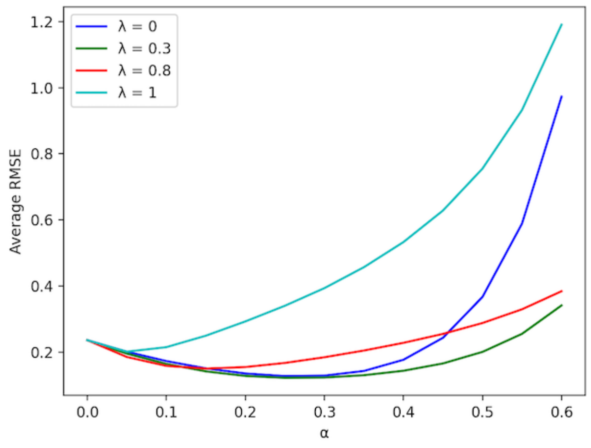

Fig. 5. Replication of figure 4 in Sutton(1988)

The second experiment explores how the learning rate affects TD learning procedure's performance when the sequence is only presented once. Similar to the first experiment, 100 training sets, each with 10 sequences of random walks are generated to present to the learning procedure. However, there are four main differences in this setup. First, the weight vector is initialized as $[0.5, 0.5, 0.5, 0.5, 0.5]^T$ so that there are no bias toward either right-side or left-side termination. Second, each sequence is only presented once to the learning procedure, instead of being presented repeatedly. Third, the weight vector is updated after each sequence, rather than after each training set. Finally, as we attempt to analyze how learning rates affect performance, each learning procedure will use a range of values of α, starting from 0 to 0.6, with a step of 0.05 increase. The RMSE is calculated in the same manner as the first experiment. Note that, for this replication, only 4 learning procedures are considered: $TD(0)$, $TD(0.3)$, $TD(0.8)$, $TD(1)$.

Following the procedure laid out above, we attain figure 5, as a replication of Sutton's figure 4.

Leveraging the setup above, we can find the learning rate that perform best for each λ value. We then plot the best RMSE achieved for each TD learning procedures to compare their performance. Note that, for this replication, we use a range of λ, increasing from 0 to 1, with a increasing step of 0.1. Doing so, we attain figure 6, as a replication of Sutton's figure 5.

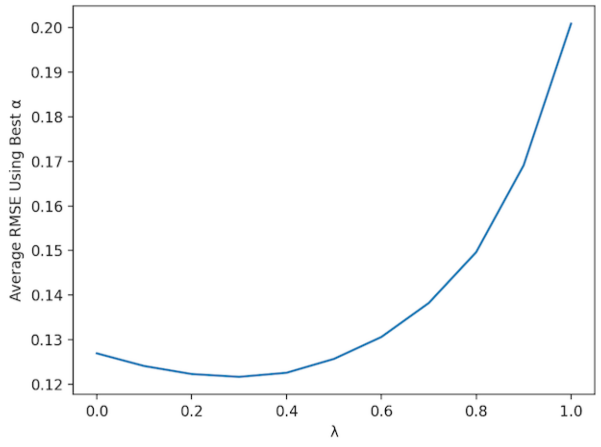

Fig. 6. Replication of figure 5 in Sutton(1988)

D. Learning rate experiment analyses

Looking at figure 5, we see that our replication closely match with Sutton's result. It's clear that the learning rate significantly affect the performance of each learning procedure. For all λ, the highest learning rate, $\alpha = 0.6$, performed worse than lower learning rate. The best learning rate for most learning procedures are intermediate values, with the exceptions of $TD(1)$, where the best learning rate is close to 0. Out of the four learning procedures, $TD(0.3)$ performed equal or better than other learning procedures with the same α values. Also, we see that $TD(1)$, represented the Widrow-Hoff procedure, performed worse than all other learning procedures for every α. This phenomenon again underlines Sutton's statement that

TD methods learn more efficiently than supervised-learning methods in multi-step problems.

Looking at figure 6, we also see that our replication closely match with Sutton's result. Similar to the first experiment, we again see that $TD(1)$ performed worse than all other learning procedures with $0 \leq \lambda < 1$. The best λ value was close to 0.3. This is different compared to the first experiment, where $TD(0)$'s performance was close to that of $TD(0.3)$. As mentioned in the first experiment's analysis, this phenomenon is because $TD(0)$ took longer to propagate prediction down the sequence. Specifically, consider the scenario right before the sequence terminates, when the agent is at state B or F and will move left or right, respectively. As the agent moves to the terminal state, the prediction will only be propagated to the state right before termination, which is either state B or F. This is different compare to the learning procedure with non-zero λ, where every state passed by the agent will be back-propagated with prediction values. This is not the case in the repeated training presentations paradigm, where after each sequence terminates, $TD(0)$ can propagate its prediction values one state further. As a result, $TD(0)$ learns slower in this experiment than in the first experiment. The main difference in our replication of Sutton's figure 4 and 5 is the range of RMSE, which is due stochastic differences in data set. Similar to the first experiment, we were able to obtain a different range of RMSE when we used a different data set.

It's interesting to note that Sutton was very particular when choosing representative λ values for his figure 4. It's clear that we would want to analyse how the extreme TD learning procedures, $TD(0)$ and $TD(1)$, perform. We can assume that Sutton chose $\lambda = 0.3$ as one of the representative values, since the best λ value was somewhere near 0.3, as was shown in our replication in figure 6. $TD(0.8)$ was chose to examine how it's performance change with different learning rate, compared to $TD(0.3)$ For $TD(0.3)$ and $TD(0.8)$, note that their performance were approximately equal for $\alpha < 0.3$, and $TD(0.3)$ only outperform $TD(0.8)$ after a certain threshold of the learning rate, which is somewhere near 0.3.

IV. Problems and Pitfalls

In this section, we discuss the problems and pitfalls encountered when replicating Sutton's experiments. The two main problems lies in the repeated training presentation paradigm experiments, which are the absence of convergence criteria and the value of learning rate.

As Sutton did not clearly define his convergence criteria, we chose to present each learning procedures with 100 training sets. However, depending on the generated data, we found that 100 training sets are not always enough for the weight vector to converge. After experimenting with different data set, we also found that there is a trade-off between the learning rate and the minimum number of training sets required for convergence. Specifically, the higher the learning rate, the lower the number of training sets needed for the weight vector to converge. As a result, the way to overcome the convergence issues is to experiment with different number of training sets and values of learning rate to find the configuration that lead to convergence. For our replication of figure 3, with the data set generated, we found that the learning rate 0.01 and 100 training sets are sufficient for our weight vector to converge. Another configuration that also works for our data set are the learning rate of 0.03 and 30 training sets (higher learning rate, lower number of training sets).

Another pitfalls in the first experiment is the absence of specific learning rate that was used. We found that, for our data set, for learning rate higher than 0.08, the weight vector values blew up and did not converged, no matter how low the number training sets presented to the learning procedures was. This happens because the weight vector is only updated after 10 sequences, and the Δw_t accumulated values over the sequences. If the learning rate is to high, and Δw_t accumulated with too much values, the weight vector will be updated with the wrong predictions which is too high. As a result, with a wrong weight vector that is already higher than actual prediction, Δw_t continue accumulate higher prediction, and thus blow up with subsequent training set. We overcame this problems by using a lower learning rate.

V. Conclusions and future works

Due to the nature of TD learning procedure, the algorithm back-propagate the observed outcome using the eligibility trace vector, taking into account the order sequence of the observations. As a result, TD methods make better use of the observed data to generate predictions. In addition, because of the way the eligibility trace vector is set up, the calculation made by the procedure is also incremental, thus avoiding peak computation while require less memory. Our replication results confirm that, in multi-step problems, temporal-difference learning procedures indeed learn faster than supervised learning procedures, represented by the Widrow-Hoff procedure, in producing prediction.

Recall that, for our replication, we use a linear approximator for the value function. For future works, it would be interesting to explore how the TD learning procedure perform compared to supervised learning with a more complex value function. In addition, it is interesting to explore the performance of TD learning procedure and other supervised learning procedure in a more complex model than the bounded random-walk.

References

[1] R. S. Sutton, "Learning to predict by the methods of temporal differences," Machine Learning, vol. 3, no. 1, pp. 9–44, 1988.