Temporal Difference Methods and Random Walk

Surya Teja Adluri

Hash Code - 22f00670bb83810f7e30d85d0c2df60d7df8ae81

1. Introduction

The purpose of this report is to replicate and analyze the experiments conducted by Richard S. Sutton in the paper titled 'Learning to Predict by the Methods of Temporal Differences' (Sutton 88). The paper introduces a class of learning methods for prediction problems named *temporal-difference (TD)* methods. TD methods are used to predict the outcome of an incompletely known system, provided it is a multi-step prediction system, i.e., partial information about the end goal is revealed at each time step. However, almost every real-world problem can be modeled as a multi-step prediction problem. For example, consider the case of a self-driving car; the algorithm's goal is to predict an incoming obstacle's accurate location to avoid a possible collision. In this case, the algorithm receives the data about the obstacle's location step by step as the car gets close. The learner updates the assumed location of the obstacle every second based on the error between the assumption and its new obstacle's location at that time step. This problem follows the conventional concept *of learning from data* to predict future behavior, similar to the Supervised Learning methods. However, the TD methods can accurately predict the outcome by evaluating the difference between successive time intervals' predictions rather than waiting for the outcome. In the example above, TD methods can utilize multiple intermediate predictions to calculate the error during the car's movement towards the obstacle.

The TD methods' ability to predict the future outcome just by learning from the successive intermediate assumed predictions rather than waiting for the outcome at the end enables these methods with considerable computational advantages and memory benefits. However, TD methods' advantages do not end there; TD methods offer far more accuracy in predictions and a faster convergence rate than supervised learning methods. The natural question here is – how the methods that predict based on the intermediate assumptions do well rather than methods predicting based on the actual outcome? The answer to this question is addressed in this paper by conducting a few experiments. Later in the paper, the experiment results are inference to find whether the faster convergence and more accurate TD methods' predictions are valid. In the subsequent sections, a formal representation of TD methods is presented, and two numerical experiments were conducted using different TD algorithms on a simple case of predicting arbitrary events.

2. Temporal difference Learning

This section presents the TD methods formally and explains how different learning methods approach the prediction problem. Firstly, to present TD learning algorithms' formal approach, a multi-step prediction problem is constructed as used in the original Paper (Sutton 88 Page 13). Consider the observation-outcome sequence $x_1, x_2, \dots, x_m, z \ where\ x_1\ to\ x_m\ are$ the sequential observations available at different time steps till m and z is the final reward (or) outcome at the end of the sequence. If the learning algorithm would want to predict the Maximum Likelihood Estimate (MLE) of z of an incompletely known system, the learning algorithm needs to predict underlying probabilities, formally known as weights. Essentially all learning algorithms can be expressed as the rules for updating these weights, as shown in equation (1) below. Experiments in the subsequent section follow the procedure of updating the weights after processing a complete sequence and also after presenting an entire training set consisting of multiple sequences. For each step of the sequence, the learning algorithm also produces predictions $P_1, P_2, P_3, \dots, P_m$. Which are the assumptions of each observation and which are used to calculate the error Δw_t by the TD methods as shown in equation (2).

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

From theorem 1 in the Original Paper (Sutton 88 Page 15), the Widrow-Hoff procedure produces the same per sequence weight updates as TD (1). The paper's remainder represents TD (1) as a conventional supervised-learning or 'Widrow-Hoff' procedure. The following equation was derived in the original Paper (Sutton 88 Page 14&15), using which family of TD algorithms can be represented.

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

Equation (2) is known as the TD (λ) algorithm, and this algorithm unites TD (0) and TD (1) methods. Additionally, intermediate TD (λ) procedures between the above two methods can be generated using different values for λ. In a general way, the family of TD algorithms can be explained as multi-step look-ahead methods. In that sense, the TD (0) algorithm is a one-step look ahead, i.e., it calculates the Δw_t by considering the next time step's observation and bootstraps the remaining value with the assumed prediction (SB20 Page124). Similarly, TD (1) can be understood as an all-the-way look ahead, i.e., it ignores all the intermediate observation steps and directly calculates the Δw_t updates using the outcome z. Any other values for λ in the equation (2) produces multi-step look-ahead methods, in which depending on the value of λ, two or more next observations are computed to calculate the Δw_t.

Equation (2) can be modified to compute the right-side sum incrementally (Sutton 88 Page16), used in the following section's experiments to gain computational advantages.

$$\Delta w_t = \alpha(P_{t+1} - P_t) * e_t \qquad (3)$$

$$\text{where } e_{t+1} \textit{ can be computed incrementally using}, e_{t+1} = \nabla_w P_{t+1} + \lambda e_t$$

3. Bounded Random Walk

In this section, different learning algorithms discussed above are presented with a numerical experiment, in which each learner's goal is to estimate the outcome of an arbitrary event. The event here is a bounded random walk, for which a Markov Reward Process (MRP) [3] is used. The experiment consists of five non-terminal states, two absorbing states, and the random walk starts from state D, as shown in the diagram below (fig.1). Absorbing states are terminal states because once a walk enters states A or G, the walk terminates with a reward of 0 or 1, respectively. Non-terminal states have the probability of 0.5 on either side, which means once a random walk enters the states B, C, D, E, F, it will have an equal chance of transitioning to either of its neighbors. This model enables the generation of various sequences of random length without any bias. The critical property of MRP used here is that the transition to the next state depends only on the current state.

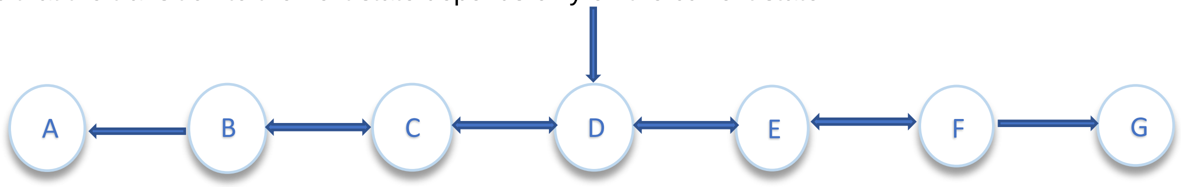


Figure 1 MRP of bounded random

Using the MRP above, the experiment gathers training data by generating multiple random walks consisting of multi-step observations and single outcome sequences as described in section 2. For example, a typical random walk includes the sequence of states – D E F E F G, where non-terminal states are the observations, and terminal state G is the sequence's outcome. A walk's outcome should be a scalar value z, as described in section 2. For instance, the sequence D E F E F G will have the outcome z = 1, and sequence D E D C B A will have the outcome z = 0. The learning algorithms used in this experiment are a family of TD (λ) algorithms and the Widrow-Hoff procedure – TD (1). The learning algorithm's end goal is to expect the estimated value of walk ending in the state G (z =1).

3.1 Training data

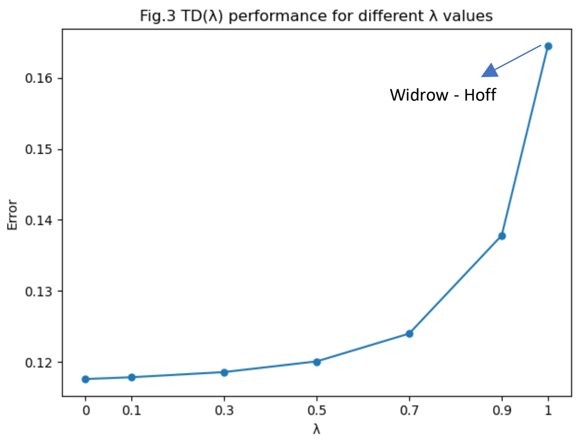
This section briefly presents the implementation details of the MRP in fig. 1 and the process of generating training sets for this experiment (Sutton 88 Page19). Firstly, the non-terminal states in a sequence are represented with a unit basis vector x_t of length 5. If the walk is in state D at time t, then $x_t = x_D$. So, the sequence D E D C B A will be represented as $x_D, x_E, x_D, x_C, x_B, 0$. These vectors are one-hot encoded, which means each state is represented with four 0s and one 1. $(e.g., x_D = (0,0,1,0,0)^T)$, $(e.g., x_B = (1,0,0,0,0)^T)$. As discussed in section 2, the TD learner predicts observations using a sequence of predictions $P_1, P_2, P_3, \dots, P_m$. For example, for a sequence D E D C B A TD learner generates predictions of $P_D, P_E, P_D, P_C, P_B, 0$. In the below experiments, a simple linear model is used, i.e., $P_t = w^T x_t$. Since there is a one-hot encoding for x_t, P_t is just the ith component of w^T where 'i' is the index of 1 in a one-hot encoded observation vector x_t. As a result, the predictions of terminal states are just the weights or probabilities underlying the MRP. For terminal states, the predictions are just rewards associated with those states. In the following section, two experiments are carried out by training the learning algorithms with 100 training sets, where each training set consists of 10 sequences. Each sequence is a random walk generated using figure 1. The algorithms compute weight updates according to equations (1) and (3). Since the non-terminal states' predictions are just the corresponding probabilities, the learning algorithm would ideally want to predict the unknown model's real probabilities. The actual probabilities of the unknown MRP for right side termination are given by $\frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}$ and $\frac{5}{6}$ for states $B, C, D, E,$ and $F,$ respectively (Sutton 88 Page 20). These probabilities are the final weights the learning models would want to converge ideally. RMS error calculates the difference between actual probabilities and weights predicted by learning algorithms in the following experiments to calculate the learning models' performance.

3.2 Experiment 1 – A Batch Presentation

3.2.1 Implementation

In this experiment seven different values were used for λ, the values are [0, 0.1, 0.3, 0.5, 0.7, 0.9, 1]. The learning rate (α) to be used by (3) is defined as 0.01. The starting weights are initialized randomly using a uniform distribution. The change in weights is computed according to equation (3), but the weights are not updated after each sequence as shown in (1) but updated after one complete presentation of a training set and continue to change until convergence. The weights are accumulated over ten sequences in a training set's single iteration, then all the accumulated weight increments are added at once to the original weight vector, then the updated weights are checked for convergence of two successive iterations. If the sum of the change in the weight vector is not changed by more than 0.001, the algorithm will move to the next training set. This process is repeated for 100 training sets, which were already gathered using the procedure described in section 3.1. Then the RMS error is calculated for each training set using the difference between final converged weights and true probabilities. These RMS errors are then averaged over 100 training sets for each value of λ. The obtained results are plotted in a graph in below figure 3.

3.2.2 Results and Analysis



The results demonstrate the accuracy of the TD methods. The results are statistically reliable as the experiment used 100 training sets, and each was presented until convergence, and each training set consists of random sequences sampled from a uniform distribution, thus eliminating any chance of bias. The performance of the TD (1) - Widrow-Hoff method is significantly lower than the other TD (λ) methods. The RMS error of TD (λ) is gradually increased as the increase in $\lambda\ value$. However, the significant increase in RMS errors has occurred only at $\lambda > 0.8$ and is worst at $\lambda = 1$. These results proved the experiment's primary goal, i.e., with any small learning rate, TD (0) is better than TD (1) for batch data. Because TD (0) tries to compute MLE using the predictions from intermediate steps (SB20 page128), it enables the TD(0) to take advantage of the underlying learned probabilities of other states. In contrast, the TD (1) or Widrow-Hoff method computes the predictions based on the outcome. It is a well-known fact that supervised learning tries to minimize the RMS error, but TD (1) only performs well on training data. However, the RMS error in fig.3 is an accurate error calculated based on the model's real probabilities. The RMS error minimized by TD (1) or any general supervised learning algorithm is the RMS error of the training set. TD (1) does not do well on new data when compared to TD (0) and other TD (λ) methods. Even though the result proves this experiment's primary goal, the values of the overall RMS errors produced by Sutton in his original Paper (Sutton88) are much higher than what was produced in this report. This scenario is analyzed in the below section.

3.2.3 Pitfalls and Assumptions

The discrepancy in error values is due to the original paper's unclear representation of specific parameters (Sutton88). Firstly, the training set's convergence criteria are not mentioned; this is a significant reason because any strict criteria would allow the learning algorithm with access to more training data, and each algorithm will perform better eventually. Nevertheless, the assumed convergence criteria in experiment 1 do not interfere with the experiment's goal, as it does not provide any bias towards any algorithm, so the key results are identical. Secondly, there is no strict learning rate used in the original experiment. Here the assumed learning rate in the experiment is 0.01, and this assumption is valid because, with any small learning rate, the learning algorithms would converge towards the best possible results.

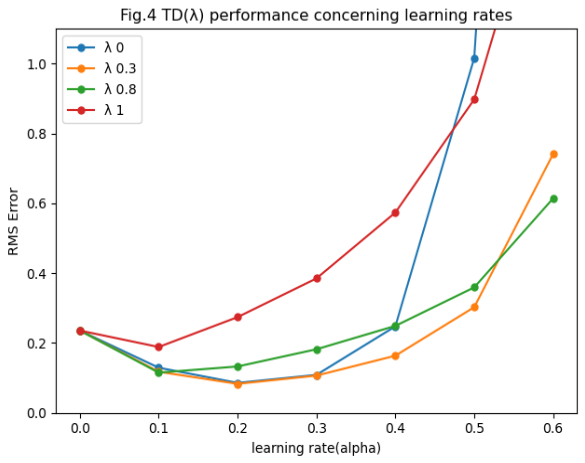
3.3 Experiment 2 – Effect of Learning rates on Limited Training Data

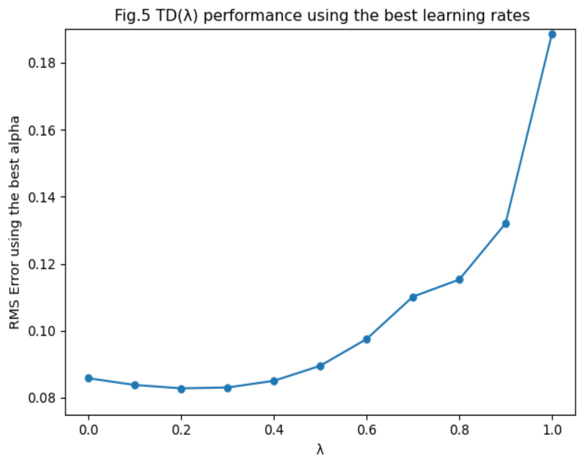
3.3.1 Implementation

As the title suggests, the second experiment considers different learning rates and their effect on the various λ values of the TD algorithm when presented with limited training data. The training set is only presented once to each learning algorithm, and weight updates are done after each sequence, rather than accumulating over a training set as done in experiment 1. As this experiment deals with presenting limited training data, the weight vector is initialized with equal weights of 0.5 each as it may take longer to converge with random sampled initial weights, and there is a chance that the limited training data will not be sufficient to converge accurately. Using this learning procedure, two figures were plotted, Fig. 4 shows the performance of each λ [0, 0.3, 0.8, 1] with the following learning rates α = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6]. This graph aims to show that TD methods outperform Widrow–Hoff procedure for every learning rate. Fig. 5 shows the performance of the following values of λ [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9,1]. Then each TD (λ) is applied to different learning rates i.e., α = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6]. Then for each value of λ, the alpha which gives the lowest error is calculated and plotted in figure 5. Both the figures use RMS error between the calculated weights and the true probabilities of the MRP. Both use equation (3) to compute the change in weights. The RMS error is averaged over 100 training sets and then presented in Fig. 4 and Fig. 5.

3.3.2 Results

The results in figure 4 demonstrate the relationship between learning rate (alpha) and RMS error; as the alpha increases, algorithms started to perform less accurately. However, when α < 0.4, all the TD algorithms except the Widrow–Hoff (TD1) procedure performed well. Moreover, even when α is very small, i.e., < 0.1, the TD (1) produced higher RMS errors than other TD (λ) algorithms. Nevertheless, the results are not as expected for higher learning rates α>0.4, i.e., TD (0) performed worse than TD (1), which is different from the original paper. The possible reasons are analyzed in the following section. The results in figure 5 demonstrate the relationship between λ values and the RMS error using the best alpha among the learning rates calculated in figure 4. The best results were achieved by λ values from 0.2 to 0.3, rather than TD (0), which is different from what was observed in figure 3 of Experiment 1 above.





3.3.3 Analysis

Figure 4 correctly proves the goal of accurate and faster convergence rates of TD methods, for values of learning rate α < 0.4, even presented with the non-batch data. Fig. 4 clearly shows the low RMS errors for TD (λ) methods other than TD (1), thus proving the accuracy. The faster convergence rates can be partly inference from the graph; TD (1) updates are moving towards minimizing the error in the training set, which is not the ideal direction. Whereas TD (0) updates are moving towards the MLE, thus producing low errors. Thus TD (0) may be faster than the TD (1), moving towards a better estimate. To better understand the results for higher learning rates α>0.5 in fig. 4, another experiment is conducted to show how each training set contributes to the average weights. The results are plotted in Figure 6 and Figure 7 below, where each figure plots TD (0) and TD (1), respectively. In **Figure 5**, the best results are achieved by λ values from 0.2 to 0.3; this is because, when limited data is present, TD (0) is slow to propagate values back. For example, at the beginning of the algorithm, the weights were initialized to 0.5, which means the only first change in weight vector happens when the sequence reaches the terminal state

A or G. This change for TD (0) is slowly propagated back by updating one value of the weight vector for the first sequence, then that change alone changes two more values. Whereas, for other λ values, the TD algorithm may change more than one value of the weight vector due to one single observation itself. Hence, TD (0) may not perform well with limited data. In any case, the TD (1) algorithm produced higher RMS values than all the other λ values. Thus, proving the primary goal of the experiment.

3.3.4 Pitfalls and Assumptions

Figures 6 and 7 show how a few training sets among 100 training sets with considerable single sequence length manipulate the total average RMS error over 100 training sets for alpha = 0.6. Because, for higher learning rates, while processing a sequence, Δw_t values are accumulating over that few lengthy sequences and forming a large sum. Moreover, If the sequence comes at the last position in a training set, then the weight updates cannot converge back to their original weights, as the TD (0) algorithm will need more sequences to calculate the difference in predictions to converge back. Whereas TD (1) calculates error based on the outcome, the weights do not jump much for high learning rates, as observed in figure 7. The same can be said about other values of λ, where the maximum weight jump is significantly less, this is because each Δw_t that is affected by the lengthy sequence can be mitigated by the other state's difference in predictions from equation (3). In the original Paper (Sutton88) author might have specifically chosen the training set that maintains the fixed average length over sequences in a training set. Since no precise details were given about choosing the training set, random walk sequences are generated using probability from a uniform distribution of random average length in this paper's experiment. However, If the no. of training sets or no. of sequences of the training data are decreased, the exact replication of figure 4 from the original Paper (Sutton 88) is possible as it would decrease the probability of such lengthy sequences. Nevertheless, when an appropriate learning rate is used, the algorithms will perform as intended, as Figure 5 correctly shows the intended results using the best learning rate for every λ. The lower overall RMS error than the original paper's (Sutton 88) figure 5 is due to the presence of lengthy sequences in this paper's training sets, which, when used with the best learning rate, helped the algorithms to achieve more accurate results.

Fig.6 Shows the weight updates for each training set for TD(0)

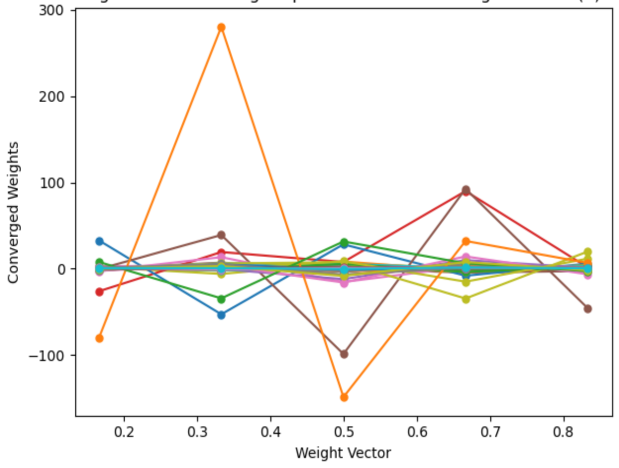
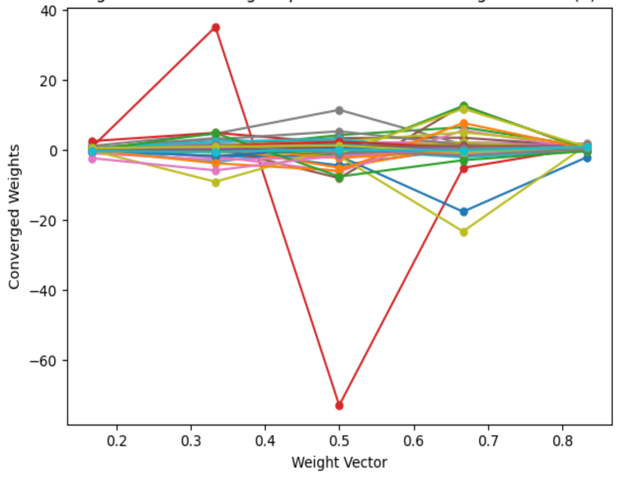


Fig.7 Shows the weight updates for each training set for TD(1)



4. Conclusion

In both the numerical experiments performed in section 3, Temporal difference learning methods performed well by offering accurate and faster convergence rates than conventional TD (1) Widrow – Hoff methods. The TD methods performed well with both batch data and limited data. TD methods are shown to be accurate and move towards the solution faster as the methods try to calculate MLE of the unknown model, whereas TD (1) tries to minimize the error in the training data as it calculates error based on the outcome and does not perform well on future predictions.

5. References

[Sutton 88] Richard Sutton. "Learning to Predict by the Method of Temporal Differences." (Aug. 1988).

[SB 20] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. 2nd Ed. MIT Press,2020.