*Analyzing TD learning with random-walk*

**Karan Jit Singh   
College of Computing  
Georgia Institute of Technologykaranjs@gatech.edu**

*Abstract*—This paper analyzes the performance of two difference approaches of TD learning specific example of a random-walk process. We recreate, analyze, and compare the two experiments for estimating probabilities of a one-dimensional random walk process as done in Sutton’s 1988 paper “Learning to Predict by the Methods of Temporal Differences”. Finally, we also validate the reasonings provided by Sutton on the behavior of different TD procedures.

Keywords—Temporal difference, reinforcement learning, random walk

# Introduction (*Heading 1*)

This article concerns with recreating, analyzing, and critiquing the performance of TD(λ) procedure with the two random-walk experiments described in Sutton 1988 [1]. Sutton introduces the TD(λ) family of procedures with the following equation:

The formula (1) describes weighted alterations where each observation is weighted by for all where m denotes the number of steps elapsed till the thobservation. is the prediction/estimate for the time step, is the learning rate, and is the weight vector. denotes the partial derivative of with respect to the components of .

## Random-Walk Process

A random walk is a stochastic process of transitioning from one state of a mathematical state space to the next with equal probability until a terminal state is reached. Reference [1] describes two experiments on a model of random-walk with 7 states as shown in Fig. 1.

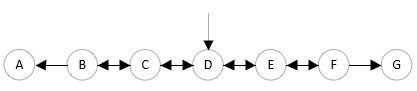


Fig. 1 A 7-state random walk, where we start in state D and with equal probability move to either the left state or the right state. The walk terminates once we reach either state A or G. A sample walk could be D-C-D-E-F-G. Adapted from Sutton 1988 [1].

All walks start from the middle state D, in each non-terminal state we have an equal probability of going in either adjacent state. The process ends when the terminal state A or G are entered.

The outcome for the model in described to be for a walk ending in the state G, and for the ending in state A. Where is the outcome of the prediction for the last transition (i.e. ) in the process (either *F to G* or *B to A*). The weight update rule for the last transition can therefore be written as

## Describing the Experiments

In the first of the two experiments described, we accumulate the delta weight values and apply update rules to the weight vector only after a complete presentation of the training set, we repeat this process until we reach convergence. In the second experiment, we present the training set to the procedure only once and apply update rules immediately after present a sequence. In both the experiments we use the learning rule (1) and the update rule (3), formally described as

# Generating The Training Data

To generate the data for the model described in Fig. 1, as prescribed for the experiments in [1], we generate 1000 sequences of the data divided into 100 training sets. We create a random walk generator that creates each sequence as follows:

1. Start in the middle state D.
2. Choose a random side and transition to it.
3. Repeat (2.) until terminal state is reached.

In our experiment we constrain the generator such that all sequences of length greater than 20 are omitted. This removes the chance of generating too divergent training sets for the small number of sequences we produce. It eventually helps the procedure converge for a moderate range of learning rates without having to have relatively more data. We can justify this omission by the fact that such large sequences have a very low probability of occurring. But if by randomness one or two such sequences creep into our limited data set, can cause divergent behaviors for that training set for our range of learning rates.

# Experiment I

## Describing the Input

In Section I, we introduced the learning rule (1), we now fully describe the prediction function . For a linear problem like the random walk in question, can be written as a linear function and as follows

where and are the th components of the weight vector, and the input vector, at time step .

For the 7 state random-walk model we maintain weights values for the non-terminal states for which the predictions are supposed to be estimated. Both vectors and are hence vectors of length 5. The input vectors, , are unit basis vectors with their th component set to 1 where maps to the non-terminal state (e.g. for the middle state D, ). Substituting an input of this form in (4) we get

The equation (5) hence concludes that with training the individual components of the weight vectors should converge to the ideal prediction, i.e. the true probabilities for the states .

## Training

As described in the introduction, in the first experiment we present the training set to the learning procedure repeatedly until the weights converging, while only updating the weights after a complete presentation of the training set. Before each training set the weight vectors were initialized to a zero vector of length 5.

## Picking a Suitable Learning Rate

For training and convergence, Sutton states, “For small alpha, the weight vector always converged in this way, and always to the same final value, independent, of its initial value.” (Sutton 1988, [1]). To arrive at a suitable alpha, we create a range of 16 values ranging from to , generated according to the rule

and ran experiment 1 with varying values of generated. The for which all training sets suitably converged for all the presented λ values was hence picked for the result.

## Convergence Criteria

To achieve convergence, the procedure was terminated for any of the 3 following conditions:

1. If the norm of change in delta weights fell below , . This also ruled out significantly small learning rates which would take too long to converge.
2. If the magnitude of the delta weights became significantly low and stabilized on that value, i.e. in each proceeding iteration oscillates between the true value.
3. If the magnitude of the delta weights started increasing after every iteration and eventually diverged, in which case the respective alpha was ruled out.

## Results

From the experiments done with the range of alpha, we pick a suitable value and plot the RMSE vs to produce the figure Fig. 2

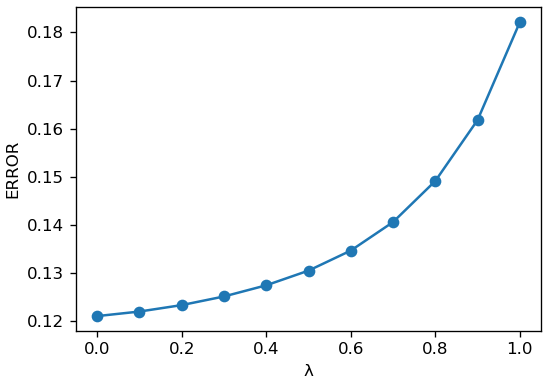


Fig 2. Average error on the random-walk problem under repeated presentations of the training set. Each data point represents the average RMSE over the 100 training sets presented for different values of lambda incrementally ranging from 0 to 1 as plotted on the x-axis.

The error noted in the figure on the y-axis is the RMSE error between the asymptotic predictions for the procedure and the ideal predictions.We note that as the value for  increases the asymptotic RMSE increases as well. We further analyze the reasoning for this in Section V.

We see slightly lower values of RMSE in our experiment’s result as noted in Fig 2. as compared to the values presented in Figure 3. of [1]. This can mostly be attributed to choosing a lower learning rate, the constraints on the data set, and the fact that floating point operations are much more precise today as compared to when the original paper, [1], was written. The result, however, presents a similar trend with respect to Sutton’s result.

# Experiment II

For the second experiment, we use the same descriptions of the problem and the input as in Experiment I. Along with presenting the same data set, and again as prescribed in [1], for this experiment, the individual components of the weight vectors are initialized to 0.5, to not bias any one side.

## Training

We present each training set to the learning procedure just once, while updating the weights after each sequence. We collected the average of RMSE over the 100 training sets for a range of alpha and lambda values.

## Convergence and Learning Rate

Since this experiment concerns with the optimality of TD procedures when update rule is applied on each sequence once, we do not have to concern ourselves with convergence of the procedure. We do however analyze in this experiment the best values of learning rate for different values of .

## Results

For the second experiment, we plot for different values of , the averaged RMSE vs the learning rate with Fig 3.

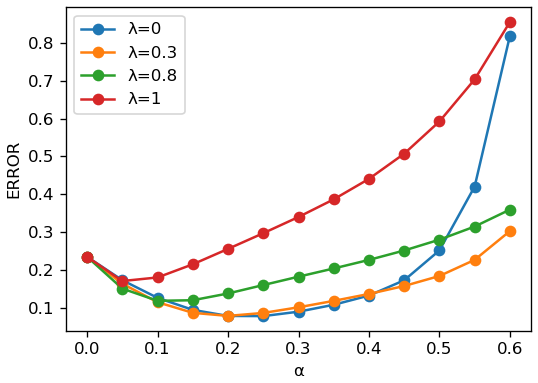


Fig 3. Average error on the same random walk model after presenting each training set just once. Each data point presents the RMSE between the prediction and ideal values for the plotted values of alpha and lambda. From Fig. 3 we pick the averaged RMSE for the best value of learning rate and plot it against to generate Fig 4.

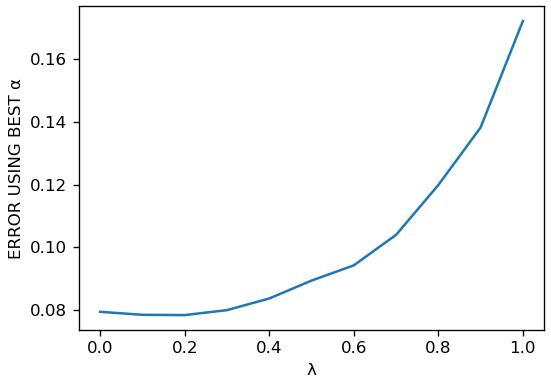


Fig 4. The plot for the average RMSE values for the best alpha for a . The alpha is chosen for each value of was the one that produced the lowest RMSE.

For the second experiment as well, we concur with the results presented in [1] by noting the slight dip in the error in Fig 4. In the proceeding section, we analyze the results of the two experiments and deep dive into the reason for the trend as given by Sutton.

# Analysis

## Contradictory result of

From the first experiment, we find the results contrary to the belief: “It is well known that, under repeated presentations, the Widrow-Hoff procedure minimizes the RMS error between its predictions and the actual outcomes in the training set” (Widrow & Stearns, 1985). After convergence, for all results , the averaged asymptotic RMSE over the training sets is less than Widrow-Hoff procedure.

As Sutton notes, the Widrow-Hoff procedure minimizes the immediate RMSE and is overall bad for minimizing future error. We analyze this reason with the following figure

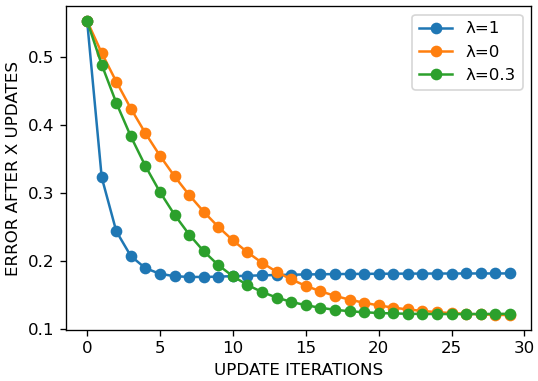


Fig 5. The RMSE averaged over all training sets after each presentation of the training set. The x-axis notes the succession of update iterations, while the y-axis plots the averaged RMSE after x iterations.

Fig 5. plots the averaged RMSE after each update iteration for three procedures, TD(1), TD(0), and the near optimal result from the second experiment, TD(0.3). Each data point presents the RMSE averaged over the same 100 training sets for each update iteration plotted on the x axis.

Our analysis confirms with the fact that Widrow-Hoff fails to minimize for future experience, we note in Fig 5 that Widrow-Hoff drops significantly in the first few iterations and fails to improve after some iterations. While other TD counterparts continually improve and eventually achieve lower errors than .

## Optimality of TD(0)

As Sutton, notes, is not an optimal solution for this problem. “One reason is not optimal for this problem is that TD(0) is relatively slow at propagating prediction levels back along a sequence.” (Sutton 1988, Pg. 22) [1]. To validate this reasoning, we analyze individual weight component errors per iteration of experiment 2 averaged over all training sets for different values of as shown in Fig 6.

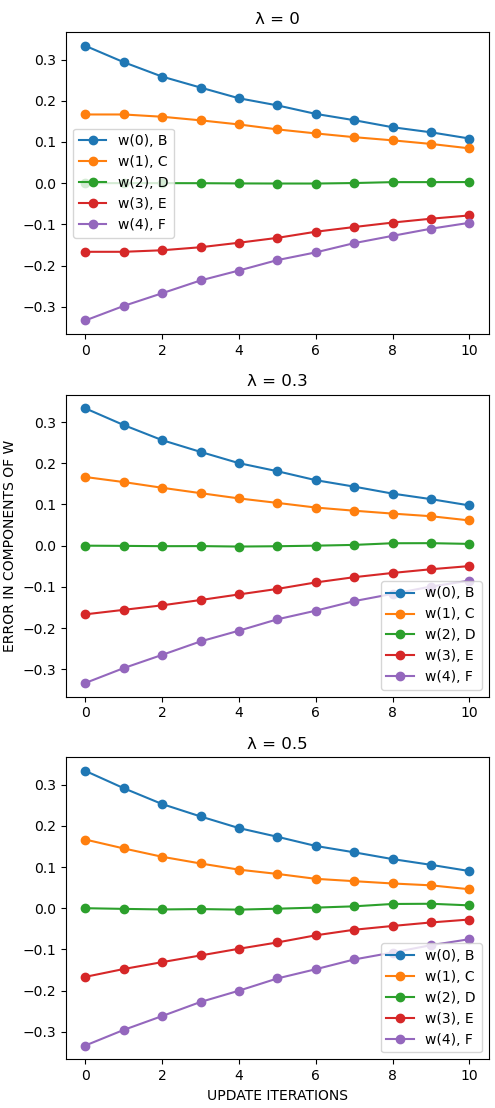


Fig 6. Error in individual components of the weight vector after presenting each sequence in experiment 2, averaged over all training sets.

Fig 6. plots 3 graphs, each for an increasing value of . Each plot contains 5 lines representing the error in the individual prediction of the 5 non-terminal states. We can see that as increases, the curves for each state squeeze towards 0 error. Because higher values of give more weight to the preceding steps in an update rule, for increasing values of we see a better propagation of the prediction estimates to the middle states B,C,E, and F.

## Initialization with Random Weights

In the first experiment, since not much was said about weight initialization for the weight vectors, we were closely able to replicate the results by assuming the initial weights as zero. For experiment 2, all components of vector were set to 0.5.

A well-known technique in the statistical learning and machine learning communities is that for ideal convergence, the initialization of parameters / weight vectors is randomized. For a slight variation in both experiments, we initialize the weight vectors with random values from a uniform distribution with values ranging from 0.01 to 0.99 and replot Fig 3. from the first experiment, and Fig 4 from the second experiment as Fig 7. and Fig 8. respectively.

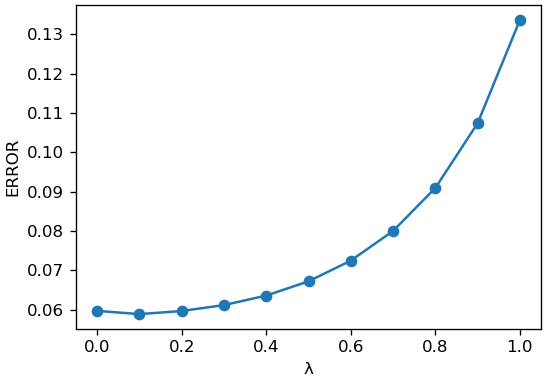


Fig 7. Replot of Fig 3 with the first experiment starting with randomly initialized weights.

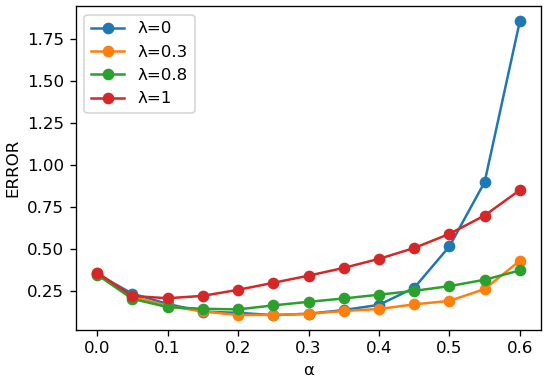


Fig 8. Replot of Fig 4 with the second experiment starting with randomly initialized weights.

With the weight vectors being initialized with random values, the first experiment performs significantly better. Due to the random values, and the fact that we repeatedly present the training set to the procedure until convergence, any bias that could be caused by the initial weights is randomized and averaged out over the training sets. With weight vectors initialized to zero values, a small bias creeped into the system for all training sets and hence increased the average RMSE significantly.

For the second experiment, however, this does not hold true. With weight updated being done only for 10 sequences per training set. The number of weight updates is not enough to remove the errors introduced by randomization and hence performs worse than when wight components are set to 0.5.

##### References

1. Sutton, R.S. Learning to predict by the methods of temporal differences. *Mach Learn* **3,**9–44 (1988).
2. Sutton, R.S, Barto A.G, Reinforcement Learing-An Introduction, 2rd ed., 2020