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

The formula (1) describes weighted alterations where each observation is weighted by lambda(k) for 0 < l <= 1, with k denoting the number of steps for elapsed since that observation. P is the prediction, alpha is the learning rule, and w is the weight vector

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. [1] describes two experiments on a particular random-walk model as shown in Fig. 1.

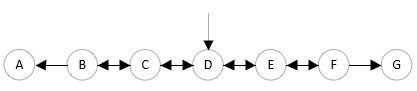
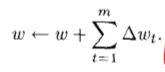


Fig. 1 A 7-state random walk, where we start in D and with equal probability move to either the left state or the right state. The walk terminates once we reach either state A or G. A sample walk could be D-C-D-E-F-G. Adapter from Sutton 1988 [1].

The outcome for the model in Fig. 1 is described to be z = 1 for a walk ending in the state G, and z = 0 for the ending in state A. For m states in the sequence, (1) for the last transition can be written as

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), described as

 (1)

where m equals number of accrued delta weights over time.

In section II and III we recreate the two experiments in sutton’s paper. In section IV of this paper we compare the results of the two experiments and critique why the methods work the way they work. Section V concludes this paper with a short note on temporal difference learning.

**Generating the Training Data**

To generate the data for the model described in Fig. 1, as prescribed by [1], we generate 1000 sequences of the data divided into 100 training sets. For the experiments, 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 helps us have the procedure converge for a moderate range of learning rates without having to have relatively more data, and compute the same in relatively less iterations. We can justify this omission by the fact that in large data sets, all sequences with length k have an approximate probability of occurring equal to for all k > 20. Second, to save ourselves from random chance, we make sure half of the sequences end in the +1-terminal state (G) and other half end in 0 terminal state (A). This is to ensure our small 10 sequence training sets are not skewed.

**Experiment I**

**A. Describing the input**

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

Pt = wTxt = sum wixi

Where w(i) and xt(i) are the ith components of the weight vector w and the input vector at time t xt.

For the 7-state random walk model we maintain weights for the non-terminal states for which predictions are supposed to be estimated. Both vectors w and x are hence vectors of length 5. The input vector are unit basis vectors with their ith component set to 1 where i maps to the non-terminal state (eg. For state D Xd = (0,0,1,0,0). Hence for an input Pt is simply the ith component of that weight for an input x(i).

**B. Training**

As described in the introduction, in this 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 randomly between 0.01 to 0.99 from a uniform distribution.

C. Picking a suitable learning rate and convergence criteria

*“For small a, the weight vector always converged in this way, and always to the same final value, independent, of its initial value.”* (Sutton 1988, [1]). To pick a suitable alpha we ran this experiment with varying values of alpha. We created a range of 16 alphas ranging from 0.0004 to 0.1024. The alphas were generated according to the rule 2i^2/10000 for 1 < i <=16.

For the convergence criteria, the procedure was terminated for 3 different conditions

1. if the magnitude of the delta weights fell below 0.05 |Aw | < 0.05
2. if the magnitude of the delta weights became significantly low and stabilized on that 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.

We picked the alpha for which majority of the training sets converged while others did not diverge, i.e, converged to a significantly low norm.

D. Results

From the experiments done with the range of alpha, we pick a suitable alpha and produce the following Fig. 2

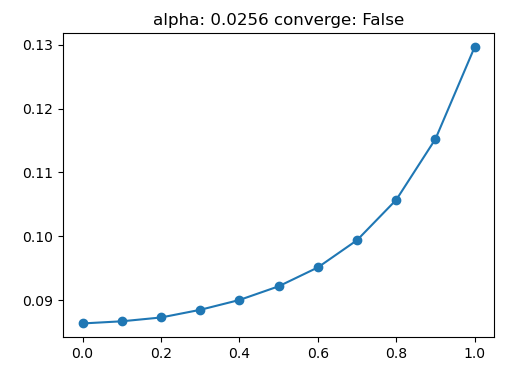
x

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

The error noted in the figure is the RMSE error between the asymptotic predictions for the procedure and the ideal predictions, [1/6, 1/3, 1/2, 2/3, 5/6].

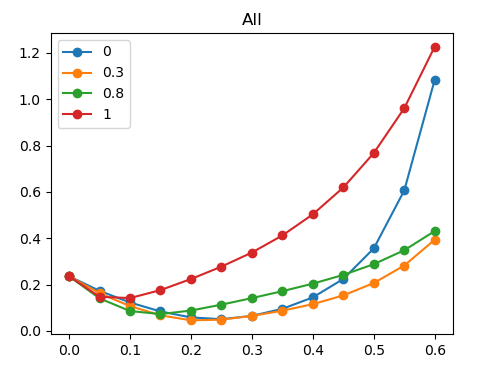
We see significantly lower values of RMSE in our experiment’s result as noted in Fig 2. as compared to the values presented in *Figure 3*. of [1]. This can mostly be attributed to choosing a lower learning rate and possibly the constraints on the data set. The result however presents a similar trend with respect to Sutton’s result.

**Experiment II**

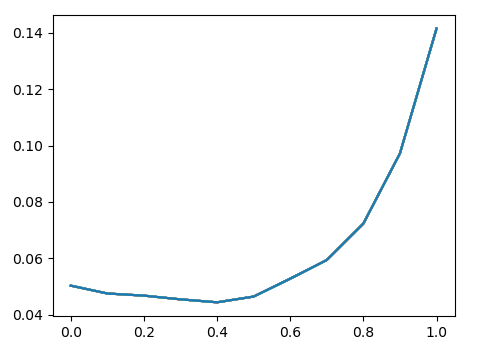
For experiment 2, we use the same description of the problem and the input along with the same training data used for experiment 1. The weights in this experiment however were initialized to 0.5 so as to not bias any one side.

**A. Training**

For experiment 2 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.



**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.**

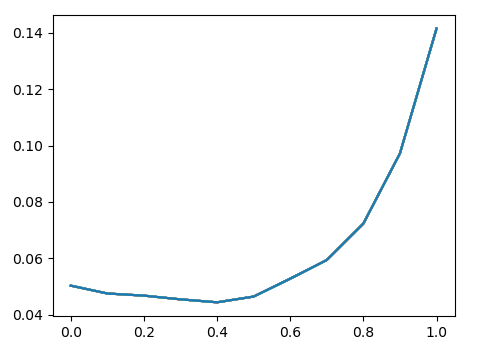


**Fig 4. The plot for the average RMSE values for the best alpha for a lambda. The alpha chosen for each**

**Comparison**

**A. Optimality of lambda(0)**

We note that lambda = 0 is not optimal for this problem. “One reason I = 0 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, we plotted the RMS values for the first 10 updates of each experiment.



In Fig 5. We note that RMSE undergoes a big drop in the first few iterations for the second experiment

We also plotted the prediction values for each of the individual states for the same experiment

