Random Walk analysis using Temporal Difference methods.

Introduction: This short paper is a summary of experiment procedures and obtained results for one of the examples, a random-walk (section 3.2), as described in Richard Sutton's 1988 paper 'Learning to Predict by the Methods of Temporal Differences'. While primary goal of this effort was to reproduce the associated results for the said example, it gave excellent opportunity to understand Temporal Difference method in great detail. Link to Video can be found here.

Brief description of the Temporal Difference (TD) methods: TD learning methods are a class of incremental learning methods for prediction problems. In contrast to conventional prediction methods, TD methods use error between successive predictions and learning happens over time whenever there is change in prediction. TD methods have 2 main advantages over conventional methods. A) They are more incremental and require lesser computation and space. B) They make more efficient use of experiences, as they use sequential learnings from past time episodes. Thus TD methods are found to be more accurate. If we define an Estimate of outcome from a state s while looking n steps into future as $E_n(s)$, then TD(λ) method can be understood as way of combining various n-step estimates. In particular, TD(λ) estimate of outcome = $\sum_{n=1}^{\infty} \lambda^{n-1}(1-\lambda)E_n(s)$

Problem description and significance of results: Random-Walk example as shown in Fig 1 and as described in Sutton's 1988 paper, is modeled. Goal is to find probability from any of the non-terminal states (B,C,D,E,F) to end up in terminal state G. Problem is modeled by assigning reward of +1 when terminating in state G and 0 when terminating in state A. A non-terminal state 'i' was represented by a basis vector xi such that i'th location in vector=1 and all others=0. Prediction for state 'i' was represented as, Pi = w*xi; where w is weight vector to be learned by TD(λ) algorithm.

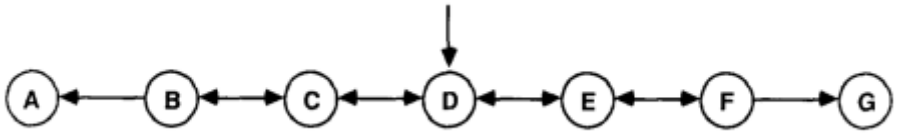

Figure 1 A generator of bounded random walks. This Markov process generated the data sequences in the example. All walks begin in state D. From states B, C, D, E, and F, the walk has a 50% chance of moving either to the right or to the left. If either edge state, A or G, is entered, then the walk terminates.

This problem is simple enough to analyze effect of choosing various values of λ. In fact actual probabilities from any non-terminal state to state G can simply be calculated using given constraints (shown in [1], section 4.1). As will be shown though empirical analysis, TD(λ) for λ<1 show better results than TD(1), where TD(1) is similar to outcome based approach (for no repeated states).
As suggested in the paper, off-line learning of weights was performed, where causal increments[2] of all previous state predictions are performed whenever there is a change in prediction of current state. As suggested in paper[1] (page 16), eligibility traces were used, where traces indicate degree to which each state is *eligible* for learning should a reinforcing event occur[2].

Experiment setup: For learning purposes, loss function is defined as L^2 norm between actual and predicted values. 100 Random walk training sets were constructed, where each set contained 10 sequences.

Two experiments were setup to see benefits of TD(λ) approach.

1) Batch update of weight vector after showing all sequences in a training set. Each training set was repeatedly shown until convergence. That is, weight updates from all sequences are accumulated over a training set and applied to the weight vector repeatedly.
2) Stochastic gradient descent after showing every sequence in a training set just once. This is called stochastic while defining cost function over a full sequence. That is, weight increments from each step in a sequence are accumulated over the sequence and applied to the weight after every sequence.

Results from 100 training sets were averaged and standard error recorded as well to validate statistical significance.

Experiment code was written in Python[3], where Dynamic Programming was used for weight updates using eligibility traces [1], [2].

Experiment results:

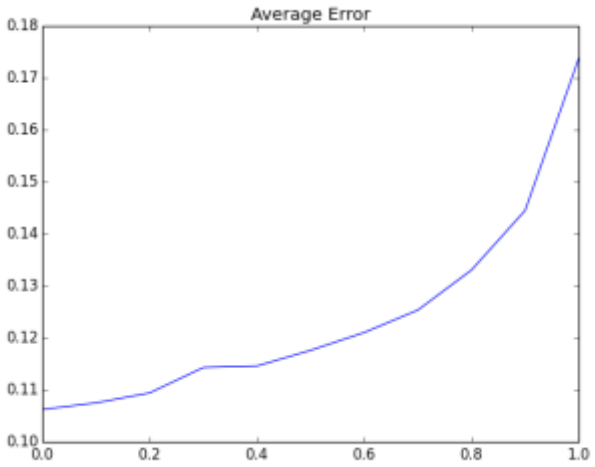

Fig 2: Experiment 1. Average of RMSE from 100 training sets (λ on x-axis)

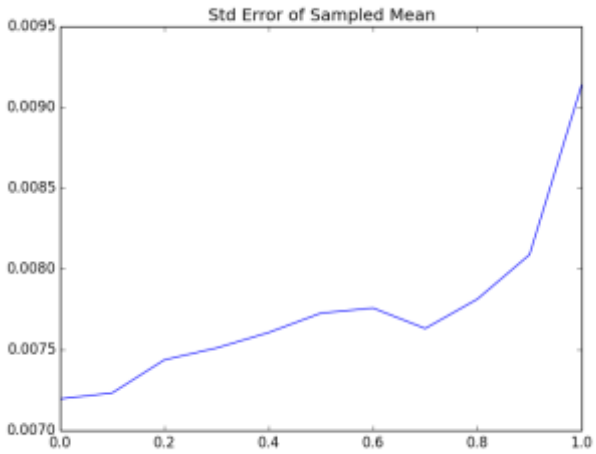

Fig 3: Experiment 1. Standard Error of samples means (λ on x-axis)

Figure 2 shows experiment 1 results, which closely resembles results shown in [1] except the absolute values are better than the original paper. TD(0) performs the best, as also shown in [1]. Difference in values could be attributed to various differences in experiment conditions done now and 31 years ago, e.g. convergence criteria, floating point precision of machines etc. However it is interesting to note that standard error (Fig 3) is almost similar to that discussed in [1] (=0.01). This shows statistical significance of results that TD(λ) for $\lambda<1$ performs better than outcome based updates (TD(1)).

Figure 4 shows experiment 2 results of average of RMSE from 100 training sets. Absolute values don't match exactly for similar reasons discussed earlier, but success of the experiment can be seen as close match of waveform trends. In fact Fig 6 shows better RMSE values than [1]. TD(1)

performs poorly compared to all other cases. Fig 5, shows standard error of sample means. It is interesting to note that for large learning rate α, standard error significantly increases for TD(1) and TD(0). So the average values for λ=0 and 1 for large α are not very trust worthy.
Also interesting to note that for experiment 2, an intermediate λ>0 value is better than 0, which is a similar conclusion from [1].

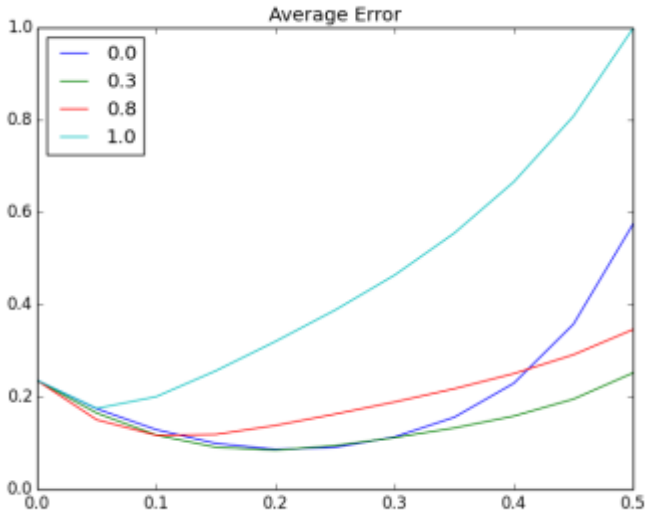

Fig 4: Experiment 2. Average of RMSE from 100 training sets (learning rate α on x-axis)

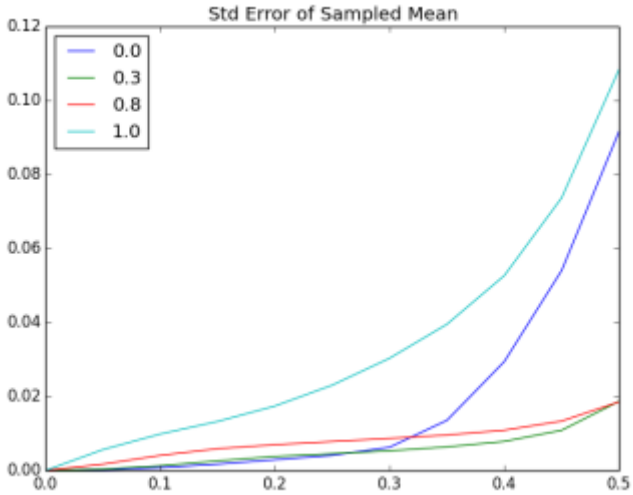

Fig 5: Experiment 2. Standard Error of samples means (learning rate α on x-axis)

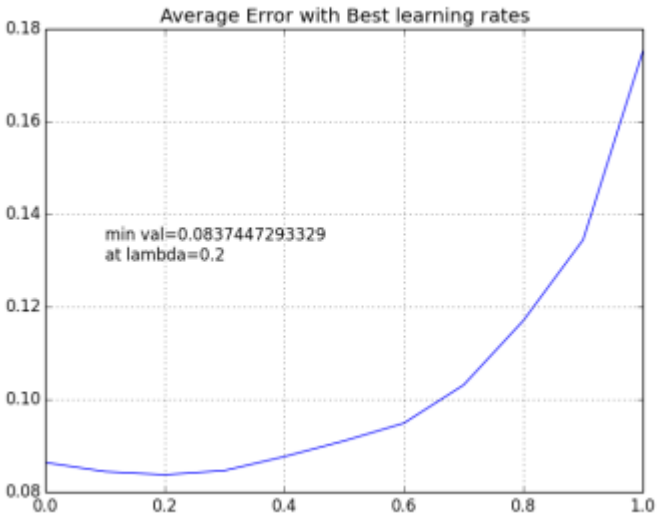

Fig 6: Experiment 2. Results with best α chosen for every λ. (λ on x-axis)

Conclusion:
The random-walk experiment discussed in [1] was successfully replicated, where the said claims about TD(λ) method were reproduced with high degree of fidelity.

References:
[1] https://files.t-square.gatech.edu/access/content/group/gtc-6bdd-3129-56f0-9a09-85aaf63fa53b/Sutton-1988.pdf
[2]https://webdocs.cs.ualberta.ca/~sutton/book/ebook/node75.html
[3] GitHub Repo