# CS 8803-O03 Reinforcement Learning: Project 1

Liangliang Yang (lyang338)

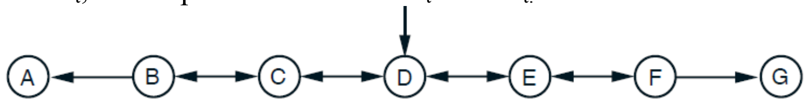
Youtube video link: https://youtu.be/mAnUGR5HTP0

## I. Introduction

The purpose of this project report is to replicate several results about temporal-difference learning described by Richard Sutton in his paper "Learning to Predict by the Methods of Temporal Differences" (1988), in order to gain a deeper and better understanding of the TD method. In the report, I will apply the TD method for the "Random Walk" example, and compare my results with that in the paper.

## II. Random Walk and TD(λ) Methods

A bounded random walk is a simple dynamical system. As in the example of paper, state D is the start state, A and G are termination states. At states B-F, it can move to left/right. When it reaches at A, outcome z=0, and for G, z=1. So a random walk example sequence can be DCDEFG(z=1) or DCBA(z=0). To make the example simple, the author defines states B-F as vectors $\{x_i\}$, where $x_B=(1,0,0,0,0)^T$, $x_C=(0,1,0,0,0)^T$ and so on. Assume at time t the state is $x_t$, do the prediction will be $P_t = w^T x_t$.



In order to apply linear supervised-learning, we will generate 100 training sets, and each training set contains 10 sequences of actions. The learning procedure is about rules of updating the weight ω of observation vectors $x_t$. In the experiment, we will update ω in two ways: after each complete sequence and after a training set of several sequences. The update rule can be expresses as below equation(1):

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

In the paper, by considering a trace decay parameter lambda (λ), the author creates a family of TD(λ) learning procedures. In particular, previous k steps are weighted according to λ as $\lambda^k$, as shown in equation (2):

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

The advantage of the exponential form enables us use the *eligibility trace* , which is the sum portion in the equation (2):

$$e_t = \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k \quad \text{and} \quad e_{t+1} = \nabla_w P_{t+1} + \lambda e_t = x_t + \lambda e_t \tag{3}$$

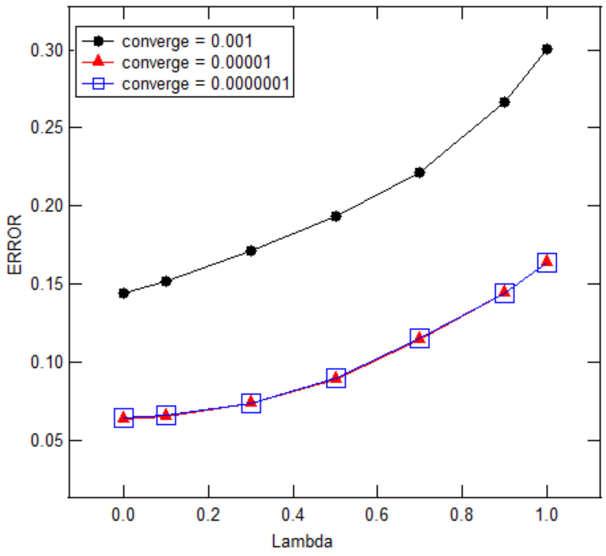
$$\Delta w_t = \alpha(P_{t+1} - P_t)e_t \tag{4}$$

By using these update rules, and recall the ideal predictions for the nonterminal states B-F are (1/6, 1/3. 1/2, 2/3. 5/6), we can calculate the root mean squared error (RMSE). The following parts will replicate Figure 3, 4, 5 in the paper. All experiments were coded with Python according to the procedures described in the paper.

## III. Experiments and results

### (a) Experiment 1 and Figure 3

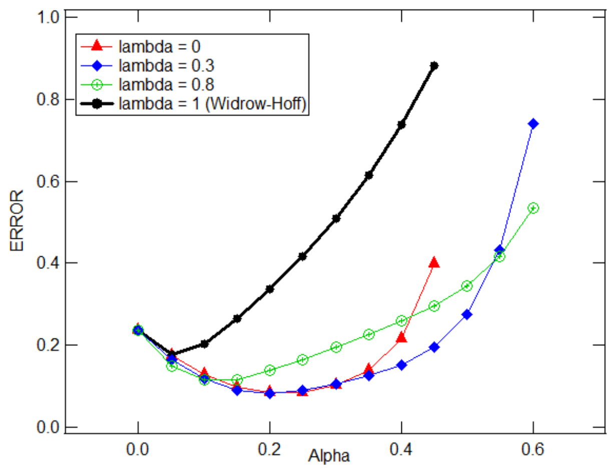
In the first experiment, the weight vector will be updated only after the complete presentation of a training set, and during the sequences Δw will be accumulated. The prediction RMSE at λ = 0, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0 were tested. The result is shown as the figure below.



In the experiment I set the learning rate as 0.001, and I tuned several converge condition of the update rule. Obviously the running time will gets longer when you choose a small converge condition. From the plot we can see that it is almost similar as the paper when we set the condition to be 0.00001 or smaller. Also we can see that when we set a bigger converge condition (0.001), we will get bigger error. So it is important for us to consider the trade-off between the running time and accuracy.

### (b) Experiment 2 and Figure 4

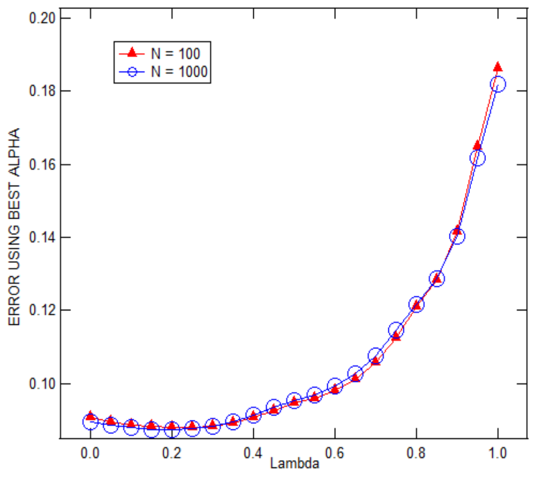
The second experiment tests the effect of learning rate α at λ = 0, 0.3, 0.8, 1.0. In the setup, each training set was only presented once, and the weight vectors were updated after each sequence. The RMSE results are averaged over the 100 training sets. The result is shown as the figure below.



The experiment result is similar as in the paper. The learning is largely depend i-on the rate, and the Widrow-Hoff (TD(1)) does the worst job. Also we find that when the learning rate increases above some point around 0.3, the result of TD(0) will quickly get worse.

## (c) Experiment 3 and Figure 5

In the last experiment, I tested the best error achieved for each λ, using the best α value. As seen from the figure, the result is similar as in the paper, which the best λ is about 0.2-0.3. Also I tuned the training sets number from 100 to 1000, and there is no big difference as we can see from the result.



## IV. Summary

In this project I tried to replicate 3 figures in the Sutton’s paper. All results are very similar to the original figures. By playing with the parameters, I gained a better understand of the TD(λ) methods. I think it is a very interesting project.