Reproducing Sutton (1988): Analysis of TD(λ) on the Random Walk Task

Hyesung Ji (hji61@gatech.edu)

***Abstract*—In this project, we reproduce the classic TD(λ) experiments on the random-walk task from Sutton's 1988 paper "Learning to Predict by the Methods of Temporal Differences". The goal is to replicate his figures, analyze the effect of the learning rate α and the trace-decay parameter λ, and discuss the behavior of TD(λ) under two different training schemes.**

I. Introduction

We will discuss the TD procedure based on the Widrow-Hoff rule. Particularly, the computation of the prediction of each state will be studied intensively and it is performed by two experiments. One experiment is to update the prediction until no significant change, another experiment is to update the prediction at every sequence without converge criteria.

II. Random-walk Example

The random-walk example is to reach the goal state by moving the state to the left or to the right direction. The probability of moving to the left and the right is the same to 0.5. And whenever moving the left or right, there is no penalty and no reward except for reaching the goal state. The random-walk consists of 7 states such as A, B, C, D, E, F, and G. The terminal states are A and G. If the state reaches to the A, it has no reward and the sequence is over. In the case the state reaches to the G, the sequence is over and +1 point are acquired as a reward. All steps from the initial state (D) to the terminal state(A or G) is called "sequence." For example, the sequence can be a D-E-F-G or D-C-D-C-B-A.

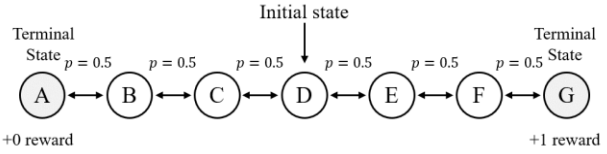

Figure 1. How to generate random-walk sequence

A. How to define the parameters

First of all, the states should be defined. In the Sutton's paper, the states are defined as a vector with length 5 except for two terminal states. However, in this paper, the states vector has 7 elements. Because, when generate the state vector randomly, it is necessary to generate the terminal state. With 5-elements vector, it is difficult to see whether the random-walk process has reached to the terminal or not. So, the state vector has 7 elements and A is (1, 0, 0, 0, 0, 0, 0) and G is (0, 0, 0, 0, 0, 0, 1). Considering the whole states in a sequence, the length of sequence is random process. Thus, the sequence is a matrix and let the sequence matrix be a x. Each row of x is a state vector. Thus, the size of x is *number of sequence* × *7*.

The second parameter is weight vector. As Sutton's paper written is the 'probability of right-side termination'. So, the elements of the weigh vector is 7 and the first element should be 0 and the final element is 1. Definitely, the A terminal has only 0 value and the G terminal has only 1 value. Other terminals' probabilities can be initialized as 0.5. As mentioned (2), these two values can play a role as reward. Because as the state goes to the right, the probability to get reward is getting higher and the prediction is calculated by multiplying the weight vector and x matrix. So, the prediction of each time can have the reward. (For G terminal, the value of weight vector is (0, 1/2, 1/2, 1/2, 1/2, 1/2, 1/2, 1) and the state vector is (0, 0, 0, 0, 0, 0, 1). The G terminal's probability is always 1 and the prediction can lead the righter element in the weight vector to be greater. It will be implemented the experiment code later.

The third parameter is prediction and it can be defined same as the Sutton's paper. The $P_t = w^T x_t$, where t is a time in a sequence. and the prediction is also 7-elements vector. It will be used to calculate the 'error' between prediction of the next time and the prediction for the current time in the each sequence. If the random-walk process has reached to the G terminal, the reward +1 will be acquired. In this case, the reward can be reflected to the prediction and it can affect the future's process. Because, the error between the current time and the next time will be calculated and if the next time prediction is G terminal, it will be reflected in the error.

B. How to generate the sequence

As explained in Figure 1, the sequence means the steps from the initial state to the terminal state. As stated in the Sutton's paper, the process of moving state is simple. The first start location is 'D' and if the state is reached to A or G terminal, the sequence is over.

Algorithm 1: Generating sequences

Function make_sequence()
Initiate state_vectors (x_i) , direction, initial_state
while 0 < state < 6
    where = random.choice(direction)
    if where = left
        $state = state - 1$
        $x.append(x_i)$
        if $state = 0$
            break
    else
        $state = state + 1$
        $x.append(x_i)$
        if $state = 6$
            break
return x

III. COMPUTATION AND EXPERIMENTS

The computation is followed the method in the Sutton's paper. Also, two experiments were performed as introduced by Sutton's paper.

A. Computation

The most important part of this paper is to update weight vector. It can affect the prediction directly. In this paper, the prediction vector contains terminal states and the weight vector has the weight value for the terminal state so that the reward is reflected to the weight vector. So, the weight vector computation is a little changed and the following. Simply, in Sutton's paper the upper bound for summation is m, but in this paper, $m-1$ is used.

$$w_{new} = w_{previous} + \sum_{t=0}^{m-1} \alpha(P_{t+1} - P_t) \sum_{k=0}^{t} \lambda^{t-k} \nabla_w P_k$$

In this formular, m is the number of sequence. Once a sequence is over, the weight vector is updated. The partial derivative $\nabla_w P_k$ is equal to x_k. (In this paper, x_k is kth row of x matrix.)

The TD error is defined $P_{t+1} - P_t$. In Sutton's paper, z$-P_t$ is the error between the prediction and real value. In this paper, the prediction P has 7 elements including the terminal state. The eligibility is computed as: $e_t = \sum_{k=0}^{t} \lambda^{t-k} \nabla_w P_k$. The final Root Means Square Error (RMSE) is calculated as $\sqrt{\frac{(w_{real}[1:-1] - w[1:-1]^2)}{5}}$. The denominator is 5, because the first element and the last element in the weight vector are for the terminal states and these values are fixed 0 and 1. So, the first and last elements does not affect the weight vector update. For the convenience, the expression for $w[1:-1]$ is equal to $w_{(5)}$ In this paper. Also, understanding the method to distinguish two kinds of experiment is very crucial for this project. One experiment is to update the weight vector once a training set and the second experiment is to update the weight vector after each sequence. The purpose of two different experiments is to compare which experiment is the more efficient method to predict the weight vector. The calculation for first experiment is the following.

B. The first experiment

The first experiment is to calculate repeatedly weight vectors until the change of weight vector converges to very small number. In this case, the weight vector is once updated after summing $\Delta\omega$ throughout all sequences. So, the delta-weight vector can be updated continuously if there is a significant changes in a training set.

$$w_{new} = w_{previous} + \sum_{j=1}^{all\ sequences} \left\{ \sum_{t=0}^{m-1} \alpha(P_{t+1} - P_t) \sum_{k=0}^{t} \lambda^{t-k} \nabla_w P_k \right\}$$

$$\text{Until } \left| w_{new} - w_{previous} \right| > \epsilon \qquad (1)$$

In this experiment, the number of sequence is 10 and the number of training set is 100. The ϵ is 0.001. The lambda is from 0.0 to 1.0 by 0.05 increment. Determining α value is also crucial to plot meaningful RMSE graph. Considering the property of learning rate, α is equal to 0.2 in the experiment. The pseudo-code for the first experiment is below.

Algorithm 2: The first experiment (repeatedly update)

```
Function experiment_1(α, λ)
Initiate w_real, number of training, number of sequence
for i = 1 to all number of training set
    while |w − w_prev| < ϵ
        w_prev = w
        for j = 1 to all number of sequence
            x = make_sequence ()
            P = w^T x
            for k = 0 m − 1(length of each sequence-1)
                err = P_{k+1} − P_k
                for l = 0 to k+1
                    e_k += Σ_{l=1}^{t+1} λ^{k−l} ∇_ω P_l
                Δw += α × err × e_k
            Δw_total += Δw
        w = w_prev + Δw_total
    rmse += sqrt((w_real(5) − w_(5)^2) / 5)
rmse = mean(rmse vector)
return rmse
```

After performing the first experiment, the RMSE figure is the following. As the Lambda values goes higher, the RMSE tends to go higher. The best lambda value looks like around 0.2 to 0.3.

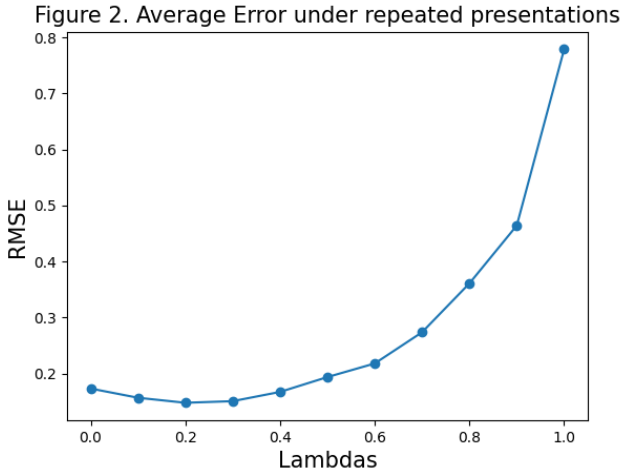

Figure 2. Average error under repeated presentations

C. The second experiment

The second experiment is to calculate weight vectors just once and it does not wait until weight vector converges. To be specifically, the weight vector is updating after each sequence and if the sequence is over, update process for the weight vector is over. Δw vector does not accumulate and it is reflected to the weight vector directly. It does not depends on the convergence condition like the first experiment. The computation is the same as the previous formula.

$$w_{new} = w_{previous} + \sum_{t=0}^{m-1} \alpha(P_{t+1} - P_t) \sum_{k=0}^{t} \lambda^{t-k} \nabla_w P_k$$

In this experiment, the number of sequence is 10 and the number of training set is 100. The λ has four values 0.0, 0.3, 0.8, and 1.0. The α is from 0.0 to 0.6 by 0.05 increment.

Algorithm 3: The second experiment

Function experiment_2(α, λ)
Initiate w_{real}, number of training, number of sequence
for i = 1 to all number of training set
  for j = 1 to all number of sequence
    $x = make\_sequence$ ()
    $P = w^T x$
    for k = 0 to $m - 1$(length of each sequence-1)
      $err = P_{k+1} - P_k$
      for l = 0 to k+1
        e_k+= $\sum_{l=1}^{t+1} \lambda^{k-l} \nabla_\omega P_l$
      Δw+= $\alpha \times err \times e_k$
    w+= Δw
  rmse += $\sqrt{\frac{(w_{real} - w^2)}{5}}$
rmse = mean(rmse)
return rmse

After performing the second experiment, the RMSE figure is the following. The tendencies depend on the α and λ values. Simply, when the α is less than 0.35, the RMSE is lower than other cases. Also, when the λ is 0.3, the RMSE is slightly lower than other cases except for $\lambda = 1.0$.

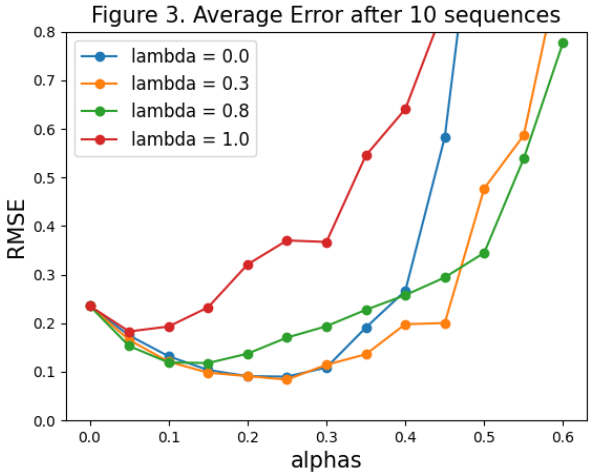

Figure 3. Average error after 10 sequences

D. The third experiment

The third experiment is equal to the second experiment. The purpose of this experiment is to find the best α value and to plot the RMSE according to λ values. As we seen the second experiment, we can visually figure out the best α value in the set $\{\alpha | \alpha = 0.0, 0.05, \ldots, 0.55, 0.6\}$.

$$\alpha_{best} = argmin\left(\sum_{\lambda=0.0}^{1.0} RMSE(\lambda, \alpha)\right)$$

To find out the α_{best} value automatically, algorithm 4 is the following.

Algorithm 4: The third experiment

Input $RMSE(\lambda, \alpha)$ from the second experiment
1D-RMSE(α) = $\sum_\lambda RMSE(\lambda, \alpha)$
$\alpha_{best} = argmin(1D - RMSE(\alpha))$ # λ=1.0, 0.8 can be excluded
for $\lambda = \{0.0, 0.1, 0.2, \ldots, 0.9, 1.0\}$
  $RMSE_{best} = experiment2(\alpha_{best}, \lambda)$
return $RMSE_{best}$

After performing the third experiment, the RMSE figure is the following. The overall tendency is similar as the Figure 2 and α_{best} is different depending on the random sequence and the value is from 0.1 to 0.15. In this case, the best λ is from 0.4 to 0.6.

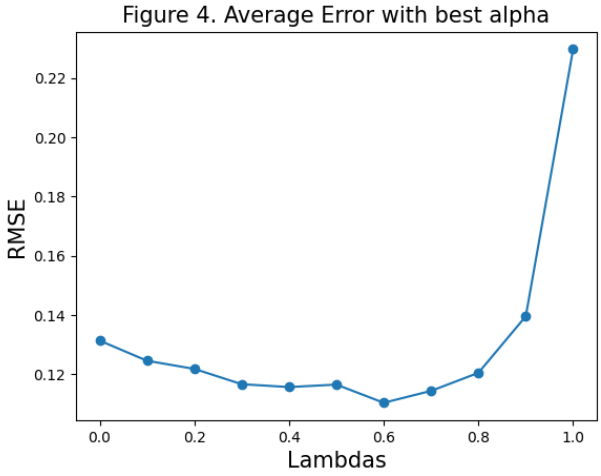

Figure 4. Average error with best alpha (In this case, α_{best}= 0.1)

On the other hand, when the λ is 1.0 or 0.8, we can know that the RMSE cannot be an optimal value according to Figure 3. So, we can exclude the $\lambda = 0.8$ and 1.0 cases and the α_{best} at that time is from 0.2 to 0.25. In this case, the figure is slightly changed. When the λ is from 0.2 to 0.3, the RMSE has the minimum value.

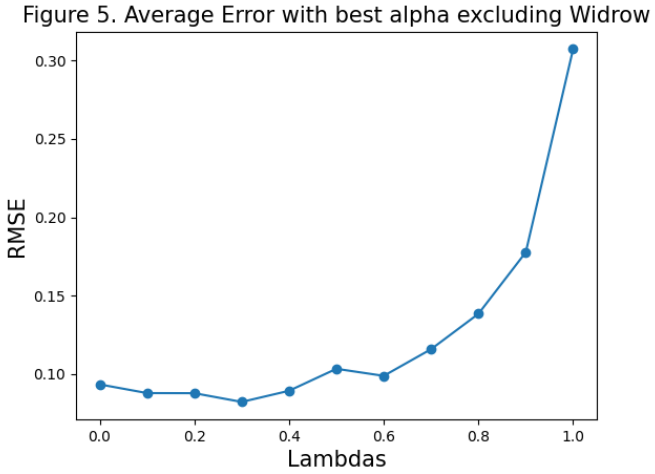

Figure 5. Average error excluding $\lambda = 0.8, 1.0$ (α_{best}= 0.2)

IV. ANALYSIS

A. The α value

The result of first experiment depends on the α value and the second experiment is also comparison along α values. In the Sutton's paper, the weight vector can converge with small α value, but the specific value was not explained. According to the property of the learning rate, The α should satisfy the following conditions.

$$\sum_T \alpha_T = \infty \text{ and } \sum_T \alpha_T^2 < \infty. \text{ So, } \alpha < \frac{1}{\sqrt{T}}$$

In this case, the number of episodes T is not fixed and it is changed in every sequence accordingly. Simply to estimate the episode, the average number of steps to reach terminal state is investigated in 100 training sets. Each training set has 10 sequences. Also, it implemented 10 times. The result is the following. The average steps to reach the terminal state is approximately 10.

trial	1	2	3	4	5	6	7	8	9	10
mean	10.02	9.89	10.27	10.19	10.37	9.81	9.78	10.04	10.42	9.92

Even if this is not statistically perfect value, it averaged 10,000 times. Thus, the estimated range of α is assumed as $\alpha < \frac{1}{\sqrt{10}} = 0.3162$. In this project, the summing Δw is not infinite. Thus, if α exceeds the upper bound, the RMSE would not goes to the infinity and we can see it in the Figure 3. However, it is clear that the RMSE should have the best value when α is less than 0.3. Due to randomness of the training set, the best α value can be from 0.1 to 0.3.

B. The λ value

As we can see the Figure 2, 3, and 5, the RMSE has normally the smallest value when λ is 0.3. If α is less than 0.4, there is no significant difference between λ is 0.0 and λ is 0.3. It can also be seen in Figure 4. As Sutton's paper indicated, TD(0) case can only change the next state's prediction. When we calculate the eligibility, the only λ^0 has a value and the next state's prediction only affect the current weight vector. In other cases, $\lambda^1, .., \lambda^m$ have own values and from next to the final stage can affect the weight vector. Therefore, the reason TD(0) has greater RMSE than other cases is the backpropagating process is not faster than other cases. On the other hands, if the λ is greater than 0.8, the changes for the eligibility is relatively faster than other cases. In this case, the weight vector can be updated excessively, so the difference between the ideal weight vector and the predictive vector blows up. The following picture can show the λ effect.

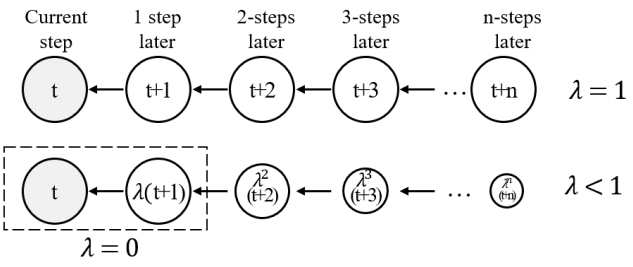

Figure 6. The λ effect

Even if there is not enough explanation for determining λ, the λ plays a role as a weighted coefficient for calculating the next stages. Simply, the TD method is to determine how much amounts of the future steps should be included for calculating the eligibility. In summary, the estimated optimal range of λ is the following: $0 < \lambda \leq 0.6$.

C. The result

The second result is better performance than the result of the first experiment. The reason that the second experiment shows better performance is the second result is to update the weight vectors gradually. The first experiment calculates the weight vector after finishing 10 sequences and it may increase the weight vector relatively excessive than the second experiment. The second experiment is to update the weight vector at every sequence and it affects the prediction and the prediction will reduce the error $(P_{t+1} - P_t)$. After that the error can reduce the delta weight vector. These procedures can increase the total weight vector gradually.

V. CONCLUSION

In this project, two experiments are implemented. The first experiment is to update the weight vector once a training set and the delta weight vector is accumulated until the norm of the weight vector is smaller than 0.001. The second experiment is to updated the weight vector in every sequences without converge criteria. Comparing two experiments, the second experiment may be better method to calculate the random-walk example. Also, the hyper parameter α and λ were investigated. To get the best performance, the α is from 0.15 to 0.25 and the λ is from 0.2 to 0.4. The λ is less sensitive than the α value. Even if the experiments gets similar results compared to Sutton's paper, there is not sufficient theoretical explanation or proof. For the next time, determining λ and α criteria should be more intensively analyzed.

REFERENCES

[1] R.S.Sutton, Learing to predict by the methods of temporal differences, Machine Learning 3, 9-44 (1988)

[2] R.S.Sutton, A.G.Barto, Reinforcement Learning, An introduction, 2nd Edition, MIT Press, (2020)