Implementation of Random Walk from ”Learning to Predict by the Methods of Temporal Differences - Richard S.Sutton”

Git Hash: bd6fa248823896ec8c563201eacc803d80d8e220

Devanathan Ramachandran Seetharaman
CS 7642 - Reinforcement Learning
Georgia Institute of Technology
Atlanta,GA
rsdevanathan@gatech.edu

***Abstract*—This project document details the implementation, discussion, and analysis of random walk example from the paper ”Learning to Predict by the Methods of Temporal Differences” by Richard S.Sutton [1].**

I. Introduction

The objective of the project is to implement the experiment from [1], where Sutton compares the performance of the Temporal Difference(TD) method and the traditional supervised learning method for the prediction problem using the random walk example. In this project, I have attempted to implement the TD algorithm two experiments(repeated presentation and single presentation) on the example given and try to replicate the results obtained by Sutton for both the experiments. I have also explained the implementation method, difference in results, possible causes for the difference, and the computational challenges faced during the implementation. The comparison of results of my implementation with [1] will be primarily based on Figures 3,4 and 5 from [1].

II. Multi Step Prediction Problem

In the paper [1], Sutton mainly focuses on the multi-step prediction problems to compare the TD method and supervised learning method. The multi-step prediction problems are the ones where ground truth or correctness is revealed in steps. Sutton cites the example of the weather forecast for the day ahead in the future as a multi-step problem, where the weather of each day preceding the target day will be intermediate observations for the target day weather forecast. Sutton also notes that for a single-step prediction, TD and supervised learning would effectively be the same, as we will have just the pairwise observations.

For the multi-step prediction problems, Sutton argues and proves that the Temporal Difference method would be superior to the supervised learning method. Sutton also mentions that the TD method will be computationally efficient as the computations are done incrementally and thereby arithmetic operations are distributed evenly over time. Sutton proves the above arguments using the random walk example implemented in this project and explained in detail in this document.

III. Temporal Difference(TD) Learning Method

This section summarizes the TD Learning method explained by Sutton in [1].

Considering a multi step prediction problem where the real experiences come in steps in the form $x_1, x_2, x_3, \ldots, x_m, z$ in which z is the final outcome, the learner comes up with the sequence of predictions given by $P_1, P_2, P_3, \ldots, P_m$ which are estimates for z. The Predictions are the function of the observations till that time t and the weights w.

The General learning procedure updates the weights after every iteration which is usually expressed as

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

where Δw_t is incremental weights.

In the case of supervised learning, the incremental weight is calculated only using the outcome z and prediction P_t at time step t. The incremental weight for supervised learning can be given as

$$\Delta w_t = \alpha \left(z - P_t\right) \nabla_w P_t \tag{2}$$

where α is a learning rate to control the increment and $\nabla_w P_t$ is gradient of P_t with respect to w.In the special case of linear function($P_t = w^T x_t$), the gradient $\nabla_w P_t$ with respect to w will be equal to x_t

In case of TD learning, instead of final outcome, the intermediate predictions can be used for calculating the the incremental weight. the error term $z - P_t$ can be expressed as

$$z - P_t = \sum_{k=t}^{m} \left(P_{k+1} - P_k\right) \tag{3}$$

Substituting equation(3) in equation(2) and simplifying the equation, the incremental weight for TD learning is given by

$$\Delta w_t = \alpha \left(P_{t+1} - P_t\right) \sum_{k=1}^{t} \nabla_w P_k \tag{4}$$

TD(λ) Learning

In the above equation(4) all the previous predictions are altered to the same extent. In [1], Sutton uses the variant of the above equation(4), where the exponential weighting with recency is used

$$\Delta w_t = \alpha\left(P_{t+1} - P_t\right)\sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k \tag{5}$$

where $0 \geq \lambda \leq 1$ for k steps in the past.

TD(1) and TD(0) Learning

Substituting $\lambda = 1$ in the above equation for TD(λ), the equation becomes

$$\Delta w_t = \alpha\left(P_{t+1} - P_t\right)\sum_{k=1}^{t} \nabla_w P_k \tag{6}$$

which is same as equation(4) which is TD method implementation of Supervised learning.

Similarly Substituting $\lambda = 0$ in the above equation for TD(λ), the equation becomes

$$\Delta w_t = \alpha\left(P_{t+1} - P_t\right)\nabla_w P_t \tag{7}$$

Sutton notes that the TD(0) equation above is very similar to the simple supervised learning equation given in equation(2), but only difference being error team $z - P_t$ is replaced with $P_{t+1} - P_t$. TD(0) considers error concerning the next observation, unlike supervised learning which considers the final outcome.

IV. Bounded Random Walk Example

To compare the Temporal Difference(TD) method and the Supervised Learning method, Sutton uses the bounded random walk example system shown below.

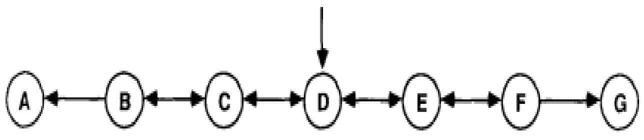

Fig. 1. Bounded Random Walk Example.

The bounded random walk generated by the above system is a state sequence that always starts with "D" and proceeds by taking steps to right or left with an equal probability until it reaches terminal states "A" or "G".

The prediction problem from the random walk example is to predict the probability of walk ending in the right terminal state "G". And hence the outcome z for a walk ending at right terminal state "G" is defined as 1 and for a walk ending at left terminal state "A" is defined as 0. For each non-terminal state the observation vector x_t will be a unit vector with 1 in that particular state and 0 in all other states. Also, the true probability for each non-terminal states is calculated as 1/6,1/3,1/2,2/3,5/6 for B, C, D, E, and F, respectively

In my implementation, the state is defined as a python dictionary with the non-terminal states as keys and right and left states as values. This state dictionary is used in the training data generation explained below.

Training Data

The training sequence is created by generating a random number between 0 to 1 which is used as a probability to decide the next state(left or right) by referring to the non-terminal state dictionary. A total of 1000 sequences are generated which are grouped as 100 training sets with 10 sequences in each set.

For the prediction problem, two experiments are implemented following the implementation briefed by Sutton in [1] using the training data generated. The first experiment uses repeated presentation, where the training set is presented repeatedly until the weight vector converges and the second experiment uses the single presentation where each training set is presented just once. Both the experiments and implementation are explained in detail below.

V. Experiment 1 - Repeated presentation

In this experiment, the same training set is presented repeatedly to the TD(λ) algorithm until convergence i.e. until there is a significant improvement in the weights. Also, the incremental weight vector(Δw) is accumulated for all the sequences(10) in the training set and the weight vector is implemented only at the end of the training set.

Technical implementation details are explained below.

Implementation

The experiment is implemented to run TD(λ) algorithm for the lambda values of [0,0.1,0.3,0.5,0.7,0.9,1] to mimic the Sutton's implementation in [1]. For the experiment, initial weight vector is considered as $[0.5, 0.5, 0.5, 0.5, 0.5]$ and the learning rate α is considered as 0.01 as used by Sutton in [1].

For each training set in the training data, multiple iterations of the TD(λ) algorithm is run and at the end of each iteration the weight vector is compared with the previous iteration weight vector and the iteration is stopped only if the difference between weights is less than 0.0001.

Within each iteration, the weight vector is updated using the cumulative incremental weight vector($\sum \Delta w$) which is calculated by adding incremental weight vector(Δw) of each sequence.

Within each sequence, the incremental weight vector(Δw) using the equation

$$\Delta w_t = \alpha\left(P_{t+1} - P_t\right)\sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k \tag{8}$$

where the last term $\sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k(e_t)$ is incrementally calculated using

$$e_{t+1} = \nabla_w P_{t+1} + \lambda e_t \tag{9}$$

Root Mean Squared Error(RMSE) is calculated for each training set. Finally average RMSE is calculated for all the 100 training sets. Below is the pseudo code version for my implementation of repeated presentation experiment. .

Algorithm 1 Repeated Presentation

```
for each Training Set do
    repeat
        for each Sequence do
            Calculate incremental weight vector($\Delta w$) using
            equation(8) and equation(9)
        end for
        Calculate cumulative incremental weight
        vector($\sum \Delta w$)
        Update weight vector($w$)
    until Convergence
    Calculate RMSE for the training set
end for
Calculate average RMSE across the training sets
```

Plot Analysis

Fig. 2. shows the plot between Lambda and average RMSE for my implementation and Fig. 3. shows the same plot for Sutton's implementation in [1]. Similarities and differences are discussed below.

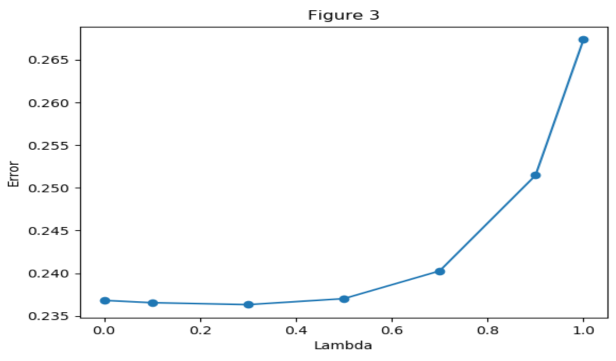

Fig. 2. Experiment 1 - My Implementation

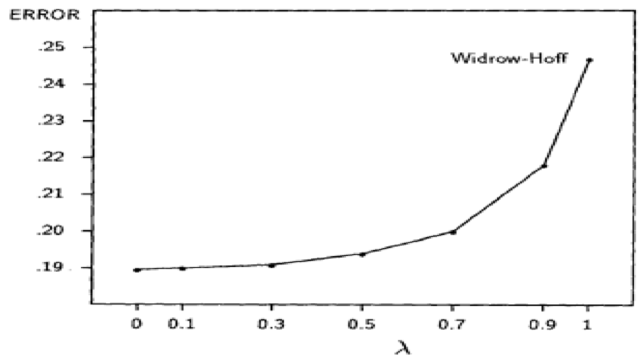

Fig. 3. Experiment 1 -From Sutton [1]

Although the implementation plot(Fig. 2) is directionally very similar to Sutton's plot(Fig. 3), there are a couple of minor differences.

The first difference is RMSE values. RMSE values for low λ values seem to be a little higher in my implementation. The possible cause could be the random nature of the sequences generated from the bounded random walk state. Another possible reason could be the epsilon value(0.0001) used to check the convergence. The actual epsilon value used by Sutton is unclear and that could result in early stopping or late stopping causing a difference in RMSE values.

The second difference is the shape of the curve for lower lambda values. In my implementation RMSE values for λ's 0, 0.1, 0.3, and 0.5 are very close and hence the curve is flat, whereas in Sutton's implementation the curve is slightly increasing since there is a considerable increase in RMSE values. This could have been caused by the same two reasons explained above.

Irrespective of these differences, the final result of the experiment, i.e. error of lower λ values is significantly better than $\lambda = 1$(Supervised learning) is consistent with Sutton's implementation.

VI. Experiment 2 - Single presentation

In the single presentation TD(λ) experiment, each training set is presented only once. Another major difference is instead of calculating the cumulative incremental weight vector($\sum \Delta w$), the weight vector is updated after each sequence directly using incremental weight vector(Δw).

The implementation details for a single presentation experiment are detailed below.

Implementation

The experiment is implemented to run TD(λ) algorithm for the lambda values of [0,0.1,0.3,0.5,0.7,0.9,1] and again the initial weight vector is considered as $[0.5, 0.5, 0.5, 0.5, 0.5]$.

However, instead of a single learning rate α value of 0.01, the experiment is implemented for multiple α values ranging from 0 to 0.6.

For each training set in the training data, only one iteration of the TD(λ) algorithm is run and the weight vector is updated after each sequence using incremental weight vector(Δw) of the sequence.

Within each sequence, the incremental weight vector(Δw) using the same equations equation(8) and equation(9) from the repeated presentation experiment. Since the weight vector is updated after every sequence, a cumulative incremental weight vector($\sum \Delta w$) is not implemented for the single presentation experiment.

After each training set, the Root Mean Squared Error(RMSE) is calculated for the training set. Finally, the average RMSE is calculated for all the 100 training sets similar to the first experiment. The experiment output would contain RMSE values for each λ and α combination for the 2 different plots explained below.

Below is the pseudo code version for my implementation of single presentation experiment. .

Algorithm 2 Single Presentation

```
for each Training Set do
    for each Sequence do
        Calculate incremental weight vector($\Delta w$) using
        equation(8) and equation(9)
        Update weight vector($w$)
    end for
    Calculate RMSE for the training set
end for
Calculate average RMSE across the training sets
```

Plot Analysis - All Learning Rates

The first figure is plotted with error values for lambda values of 0, 0.3, 0.8, 1 for different learning rates α. Fig. 4 shows the plot from my implementation and Fig. 5 shows the plot from Sutton's implementation.

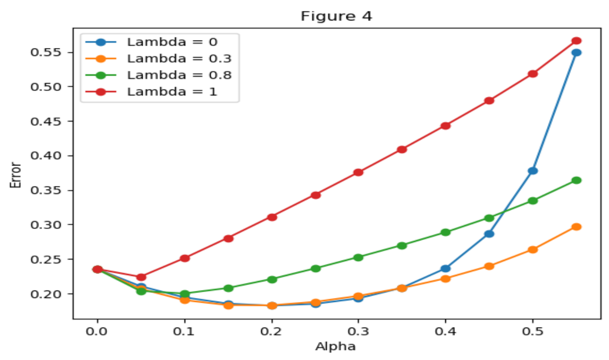

Fig. 4. Experiment 2(all learning rate) - My Implementation

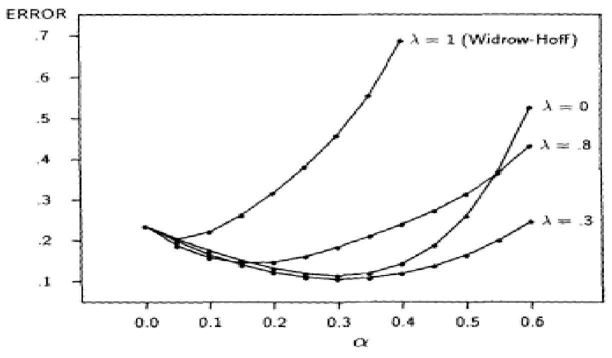

Fig. 5. Experiment 2(all learning rate) -From Sutton [1]

For all the λ values, the learning rate had a significant impact on the error values. The learning rates of 0.1 and 0.3 had a minimal error. Also irrespective of the learning rate, TD(1) performs worst than any other λ values.

Out of the λ values considered, λ value of 0.3 performs for all the learning rate values.

One of the considerable differences between the implementation plot and Sutton's plot is the TD(1) line. The TD(1) line more linear in the implementation plot but a little curved in Sutton's plot. Also, the overall RMSE error values are a little narrower compared to Sutton's plot. The possible reason again could be the random sequences generated for the bounded random walk system.

Plot Analysis - Best Learning Rate

The second figure for a single presentation is plotted for RMSE values for all λ's with only the best(minimum RMSE) learning rate for each λ.Fig. 6 shows the plot from my implementation and Fig. 7 shows the plot from Sutton's implementation.

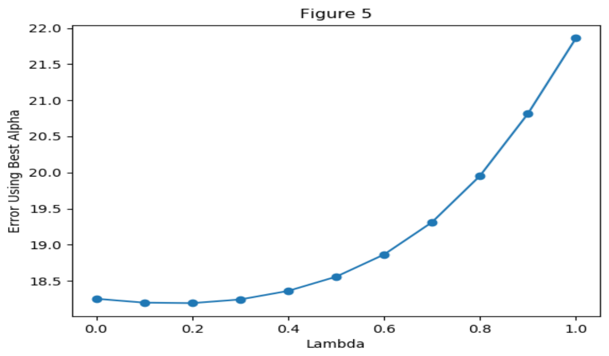

Fig. 6. Experiment 2(best learning rate) - My Implementation

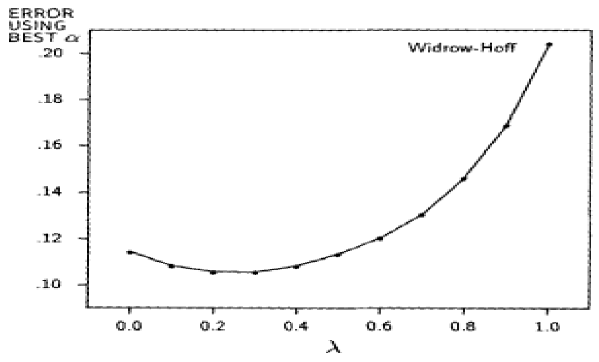

Fig. 7. Experiment 2(best learning rate) -From Sutton [1]

As with the other plots, TD(1) performs worst than any other λ values for the best alpha values, re-iterating Sutton's observation that the TD learning method performs better for multi-step prediction problems.

There is a significant difference in the actual RMSE values between the implementation plot and Sutton's plot. Also, the error for TD(0) is a little higher than TD(0.2) in Sutton's plot whereas in the implementation plot they are almost similar.

The reason for these discrepancies is not clear but again can be the result of the random sequences generated for the random walk.

VII. Assumptions and Challenges

This section details assumptions made and challenges faced in the implementation of bounded random walk example from [1]. Although TD(λ) learning method and equations were easy to follow in Sutton's [1], the instructions for bounded random walk example is not very detailed and hence some of the parameters and algorithm steps are assumed in the implementation.

One of the challenges is in understanding how the training set is presented to the TD method for the repeated presentation experiment. The procedure to calculate cumulative incremental weight was not clear and the piazza forum helped to understand the procedure. The epsilon value used for convergence of repeated presentation experiment was also not clear from the paper.

Lower epsilon values such as 0.000001 take a very long time and iterations to converge for each training set and the value of 0.0001 is used. Also to understand the gradient and eligibility trace calculation, Sutton's textbook [2]. And for selecting the best α for the last figure, the list of alpha values considered was not clear. In the implementation, α values between 0 and 0.5 in increments of 0.01 are considered.

VIII. Conclusion

In this project, I implemented the bounded random walk example mentioned in Sutton's [1], which he uses to prove that the TD(λ) method is superior to the supervised learning method and is computationally efficient. The results of implementation despite the minor differences in RMSE values, the results directionally matched with Sutton's results and established that the TD(λ) method is superior to the supervised learning method.

References

[1] Richard S. Sutton. 1988. Learning to Predict by the Methods of Temporal Differences. Mach. Learn. 3, 1 (August 1988), 9–44. DOI:https://doi.org/10.1023/A:1022633531479

[2] http://incompleteideas.net/book/RLbook2020.pdf