CS 7642 - Reinforcement Learning
Project 1 - Desperately Seeking Sutton

Prasanna Venkatesan Srinivasan
MS in Computer Science at Georgia Tech
psrinivasan48@gatech.edu
Git Hash: c4feef43e8f4d25d8a32f80b693a280752703fe3

Abstract

This report explores an incremental learning procedure called the Temporal Difference method as a way of solving prediction problems. The problem we have chosen for analysis is the infamous bounded random walk experiment. We have analyzed the impact of model parameters and hyperparameters on the learning process thoroughly and the results seem to be fairly consistent with Sutton's paper on "Learning to Predict by the Methods of Temporal Differences."

1 Introduction

Conventionally, supervised learning methods are used to model past experiences to predict future behavior. Supervised learning procedures like SVM, make use to large amounts of data being entirely presented to it make reliable predictions. These learning procedures are apt for solving single-step prediction problems, where the model is constructed to associate observation-outcome pairs and the correctness of prediction is revealed all at once.

Temporal Difference Methods are a class of learning procedures that aim to solve multi-step prediction problems. Similar to supervised learning approaches that learn by measuring errors between the expected output and the model output, temporal difference methods learn by computing errors between temporally successive predictions. TD methods are incremental and hence place lesser demands on time and memory. They make better use of experience and converge faster.

2 Background

2.1 Notation

Let x_1, x_2, x_3,...,x_m be the vectors of observations, where x_t is the observation made at time t. We use z, a scalar, to denote the outcome obtained as a result of t observations. For each observation x_t, the learner produces a prediction P_t. P_t is a function of x_t and w and can be expressed as $P_t(x_t, w)$.

2.2 Widrow-Hoff Rule

Generally learning a model is accomplished by iteratively updating the weights as a function of the difference between the observation and outcome.

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

where ,

$$\Delta w_t = \alpha(z - P_t)\nabla_w P_t \tag{1}$$

α controls the rate at which model moves towards convergence called the learning rate and $\nabla_w P_t$ is partial derivative of P_t with respect to w. If $P_t = w^T x_t$, then the above equation reduces to the Widrow-Hoff learning procedure.

$$\nabla w_t = \alpha(z - w^T x_t)x_t \tag{2}$$

2.3 TD(1) learning method

Equation (2) is suited to solve single-step prediction problems where a model is constructed to learn the mapping between predictors and the response. Models used in the multi-step process produce a set of intermediate predictions at every time-step t (P_t). Thus, the error term in equation (1) must be represented by the sum of errors in predictions up to time step t.

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

We can plug equation (3) in (1) to get the following equation.

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

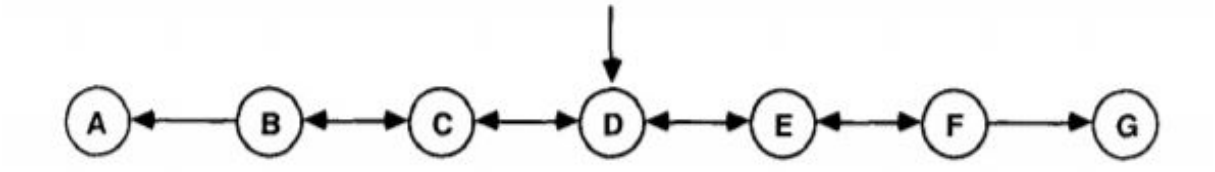

Figure 1: A model of the bounded random walk experiment. D is the starting state. B,C,E,F are intermediate states. A and G are terminal states. The agent is given a positive reward when the walk terminates at G.

Equation (4) is the TD(1) learning rule.

2.4 TD(λ) learning method

TD(λ) is an extension to TD(1) learning procedure, where the weight updates are done as per the following equation.

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

The rationale behind TD(λ) is quite similar to that of the TD(1) learning procedure. TD(λ), similar to TD(1) learning procedure, measure the prediction error at time t to be the sum of all prediction errors up to t, but differs by the use of a model parameter - λ ($0 \leq \lambda \leq 1$) that helps it weigh on the relative importance of its experience obtained at each of previous time steps.

$\lambda = 1$ leads to providing equal importance on all the predictions encountered up until time-step t. $\lambda = 0$ gives rise to a myopic policy that only cares about the most recent observation during the weight update.

3 Problem

3.1 The Random Walk Experiment

We use the bounded random walk example to perform our analysis of temporal difference learning methods. Bounded random walk is a sequence of states, traversed by an agent starting from the initial state, until a terminal state is reached. A model of the random walk experiment that has been assumed for our analysis has been shown in Figure 1. The description of the problem is as follows:

- S - The state space - $\{A, B, C, D, E, F, G\}$
  - D - The start state
  - B, C, E, F - Intermediate states. D can also be an intermediate state.
  - A, G - Terminal states
- A - The agent is allowed to execute two actions:
  - Move to the left
  - Move to the right
- T - State transition probability function

$$T_{sa}(s') = \begin{cases} 1 & \text{if a=left and s' is to the left of s} \\ 1 & \text{if a=right and s' is to the right of s} \\ 0 & \text{otherwise} \end{cases}$$

- R - The reward function

$$R(s) = \begin{cases} 1 & \text{if s = G} \\ 0 & \text{otherwise} \end{cases}$$

- The game terminates as soon as the agent reaches the terminal states (A or G).

3.2 Implementation Details

The learning method that we employ must be able to arrive at the probability of right side termination at state G, for an agent in all of the non-terminal states (B, C, D, E, F). Every state is represented by a unit vector. For example $X_D = (0, 0, 0, 1, 0, 0, 0)^T$ represents the vector for state D.

During every time-step, we compute $P_t = w^T x_t$ to compute the weight updates. We can use TD(λ) equation in (5) repeatedly to learn the vector of probability values of the right-side termination, represented by w. There are two parameters involved that control how the agent learns.

- Model parameter λ that controls how much of the past predictions matter to compute the weights.
- Hyper parameter α that influences the rate of learning.

Further analysis on the impact of each of these parameters on the learning is done in section 4.

4 Experiments and Results

4.1 Repeated Presentations Paradigm

To obtain statistically reliable results, 100 training sets each consisting of 10 sequences were generated for the problem. For each of the hundred training sets, update in weights were computed according ton TD(λ) procedure detailed in section 2.5, using the repeated presentations training paradigm. The ideal predictions of right side termination are computed as $\frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3} and \frac{5}{6}$ for states B, C, D, E and F respectively.

The pseudo-code for TD(λ) repeated presentation training paradigm is given below:

Algorithm 1: Repeated Presentation Training Procedure - Random Walk

Result: RMS error for each value of λ
foreach λ **do**
  foreach *trainingset* **do**
    Initialize w
    Initialize Δw
    repeat
      foreach *sequence* **do**
        foreach *state* **do**
          $\Delta w \leftarrow \Delta w + update$
        end
      end
      w $\leftarrow$ w + Δw
      calculate $rmserror$
    until *convergence*;
  end
  Compute mean of all $rmserror$s
end

The repeated presentation training paradigm holds off the weight update until convergence is obtained on an entire training set. The learning at a time-step or an entire sequence is not transferred onto the subsequent ones, immediately.

We start by initializing w to a vector of 0.5 for each training set, to indicate equal probability of right side termination on each of the non-terminal states (B, C, D, E, F). Δw is calculated according to equation (5) for every iteration and are accumulated up until the learning procedure does not produce any significant weight updates. Once, the model has converged for a training set, the error between the expected and the actual w is obtained by computing the root mean square value between them. The mean of error over all training sets is assumed to be the actual error value. The procedure is repeated for multiple λ values. The learning rate α is set to a constant value less than 1 throughout the process.

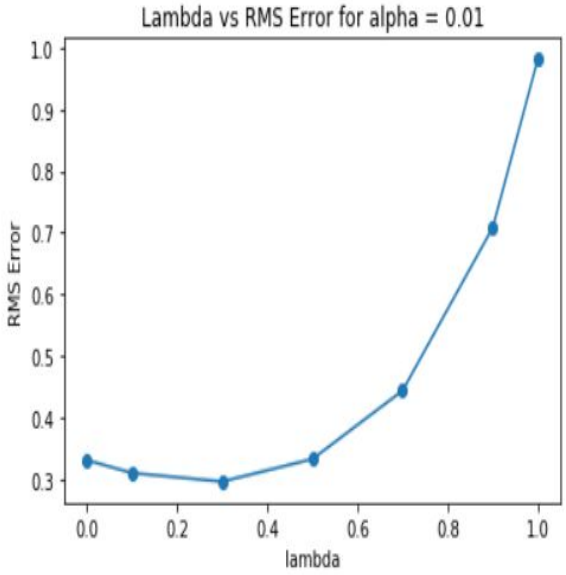

Figure 2: λ vs RMS Error

The values of root mean square error are plotted as a function λ. The RMS error increases with increase in the value of λ in Figure 2. It is to be noted that TD(1) procedure, which is similar to the conventional Widrow-Hoff procedure exhibits the maximum error. This proves the fact that, TD procedures of $\lambda < 1$ are better at making future predictions that the conventional methods.

4.2 Single Presentation Method

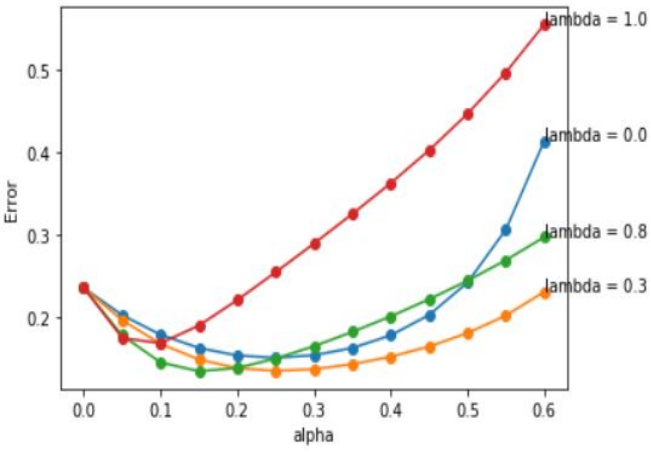

Figure 3: α vs RMS error for multiple λ values using Single presentation procedure

Algorithm 2: Experiment 2 - Random Walk

Result: RMS errors for each λ and α
foreach α **do**
  foreach λ **do**
    foreach *trainingset* **do**
      Initialize w
      foreach *sequence* **do**
        foreach *state* **do**
          $w \leftarrow w$ + update
        end
        calculate $rmserror$
      end
      Compute mean of all $rmserror$s
    end
    Compute mean of all $rmserror$s
  end
end

Algorithm 2 differs from Algorithm 1 in the following aspects:

- We do not present the training sets repeatedly until convergence. The learning procedure is run exactly once on each of the training set.
- Weight updates are done at each and every time-step. The learning (about the weight updates) is transferred between states and between sequences in a training set.
- The procedure is run for varying values of α and λ
- The weight vector is initialized to values of 0.5 for each and every training set
- RMS errors are computed for every sequence and training sets and are averaged to produce a single error measure for a particular value of λ.

It can seen in Figure 3, that learning rate α has significant impact on learning. Given a particular value of λ, lower values of α offer better performance. Models with lower learning rates tend to be less aggressive learners and guarantee against the overshooting of the global optimum, which has been substantiated as per the results obtained.

It can also be inferred from the results that $\lambda = 1$ offers the worst performance on a predominant subset of α. TD(λ) methods with $\lambda < 1$ are seen with better performances than the conventional Widrow-Hoff (supervised learning) procedure. The graph suggests that the intermediate values of λ can lead to better learning, than the farsighted (TD(1) - Cares about all the predictions) and the myopic (TD(0) - Cares about the most recent prediction alone) policies. The best value of λ seems to be 0.3.

Algorithm 3: Experiment 3 - Random Walk

Result: RMS errors for each λ using the best α
foreach λ **do**
  foreach α **do**
    foreach *trainingset* **do**
      Initialize w
      foreach *sequence* **do**
        foreach *state* **do**
          $w \leftarrow w$ + update
        end
        calculate $rmserror$
      end
      Compute mean of all $rmserror$s
    end
  end
  Identify the min $rmserorr$
end

The third experiment has been outlined in Algorithm 3. For every value of λ identify the best value of RMS error. The best value is the minimum value of error obtained by running the single presentation procedure on the training set on all values of α.

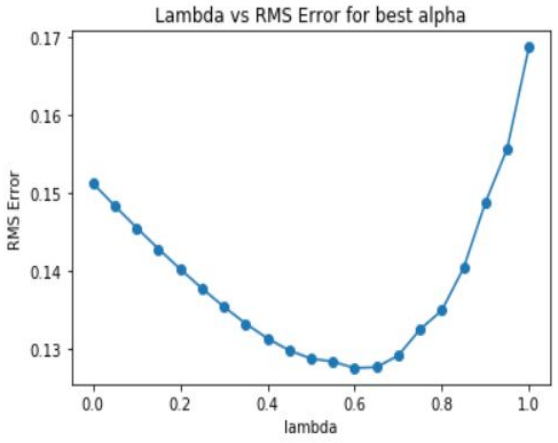

Figure 4: RMS error vs λ for best α

It can be seen in Figure 4 figure that the value of RMS error drops as λ increases with the best value of α and then rapidly increases. The plot also suggests that $\lambda = 0.4$ to 0.6 would be appropriate if an optimal learning rate is selected. It can also be concluded that very low values and very high of λ exhibit sub-standard performances. Very low λ values are not recommended due to the fact that the smaller λ values are slow at propagating the experiences back along the sequence.

5 Conclusion

The following assumptions were made during the implementation of the above experiments.
Length of the sequence: It has been assumed that the length of the random walk is limited to a value of 14. Longer sequences caused the errors to blow up especially for the far-sighted learning procedures like TD(1).
Condition for Convergence: Repeated training procedure was based on a fixed number of epochs than a convergence on weight updates. The updates were run for a fixed number of times on each training set, which allowed different α and λ values being bench-marked against without regards to the time taken during the learning process.
Figure 2 and Figure 3 closely resemble to the original results in [1]. Figure 4, differs from original implementation, possibly due to the assumptions discussed above. But, it still indicates that the values of λ very close to 0 and very close to 1 produce larger errors.

6 References

1. Sutton, R.S. Learning to predict by the methods of temporal differences. Mach Learn 3, 9–44 (1988). https://doi.org/10.1007/BF00115009