CS7642: Project 1

Rory Michelen | rmichelen3@gatech.edu | Git Hash: fbf8cd0ed49472b1cd31b46c34ee5cb1a83e46c5

I. Introduction

Temporal difference (TD) learning is an iterative method of prediction that learns by comparing differences between the prediction of the previous iteration and that of the current iteration. By repeating this comparison process, TD learning improves its predictions until there are small to no differences between successive iterations.

TD learning is particularly useful for prediction problems involving multi-step observation-outcome pairs. "Observation-outcome" refers to the fact that each sequence or trial is composed of a final outcome and one or more observations . "Multi-step" refers to the fact that each trial may consist of more than one observation before the final outcome is experienced.

In his 1988 paper, "Learning to Predict by methods of Temporal Difference Learning", Sutton explains that the Widrow-Hoff procedure- a common supervised learning method, is simply a special case of TD learning. Sutton demonstrates that methods of TD learning outperform this special case for multi-step observation-outcome prediction problems. Sutton demonstrates this quality through two experiments- each highlighting a different TD learning procedure. In his experiments, Sutton compares TD learning procedures to the Widrow-Hoff procedure using a random walk example.

In the following sections, the author will explain his understanding of Sutton's experiments and procedures as well as his attempts to replicate Sutton's results

II. Experiment Environment

A. Bounded Random Walk

Imagine a person (let's call her Clare) who goes for a walk. With each step, Clare flips a fair coin. If the coin lands on heads she goes East. If it lands on tails, she goes West. This is an example of a random walk. Random walks do not need to involve people, they are models for describing many processes found in nature and industry.

Imagine that if Clare ends up 3 steps further East than her starting position, she receives a dollar and her walk is over. If she ends up 3 steps West of her starting position then she gets nothing and her walk is over. This implies that there are 7 potential states, 5 of which do not result in immediate termination. Sutton refers to these states 7 as A, B, C, D ,E, F, and G (from left to right).

Sutton's experiments concern the prediction of the value of each of these 5 non-terminal states (B through F). In other words, if Clare is in a given state, what is the probability she will receive the dollar?

B. State Representation

Throughout the rest of this report, states are represented as a 5x1 1-hot encoded vector. Terminal states were excluded since they are implied by the end of a sequence. For example, the third, non-terminal state, D, would be coded as

$$x_D = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix}^T$$

C. Implementation of a Random Walk

The pseudo-code for a random walk is trivial and is therefore excluded from this report. However,while implementing a random-walk algorithm, a question arose, did Sutton limit the maximum or minimum length of his sequences?
This question is answered throughout the rest of the report.

III. Temporal Difference Learning

Sutton proposes two TD learning procedures in his experiments. However, both procedures have the same components. These components will first be described separately and will later be used to describe the two learning procedures

A. The weight vector, w

W is a 5x1 vector. It is our prediction of the value of each of our 5 non-terminal states. For example, if $W = \begin{bmatrix}0.5 & 0.5 & 0.5 & 0.5 & 0.5\end{bmatrix}^T$, this would imply that we believe each of our 5-terminal states have an equal probability of ending with outcome 1 or 0.

With each iteration of Sutton's TD algorithm, our W vector gets closer to the true values of the states. The W vector does this with the help of the remaining components described below.

B. The prediction vector, P

The vector P is used to represent predictions in a different way. It represents the predicted value of each state in a given sequence or random walk. For example, consider the W vector, $W = \begin{bmatrix}0.1 & 0.2 & 0.4 & 0.6 & 0.8\end{bmatrix}^T$,

and the sequence D-E-F-E-F-G. Then, we can reference W in order to construct P such that it represents the predicted values at each step in the sequence. In this case, $P = \begin{bmatrix}0.4 & 0.6 & 0.8 & 0.6 & 0.8 & 1\end{bmatrix}^T$. Note that the outcome, the last entry of P, does not come from W. Instead, it is observed after completing the sequence. Sutton's algorithms also reference specific entries in P. P_t is used to denote the predicted value of the state at step t in a given sequence.

C. Partial Derivative of P

$\nabla_w P_t$ refers to the partial derivative of P_t with respect to W. In laymen's terms, this means that it is a 5x1 vector used to represent the state of a sequence at time t. For example, in the sequence D-E-F-G, $\nabla_w P_2 = \begin{bmatrix}0 & 0 & 0 & 1 & 0\end{bmatrix}^T$. This is because at time t=2, this sequence is in state E, the fourth, non-terminal state.

D. Hyperparamters

Sutton uses two hyperparameters. First is α, the learning rate. α controls how fast the procedure learns. It does this by controlling how much the information from current iteration is considered when updating W. As we will see later in this report. Selection of a proper α value is critical to the success of TD learning procedures.

Second is λ. When a reward is observed, λ controls how many steps back in the sequence the reward is attributed. Going back to the analogy from section II. A, imagine Clare performs the sequence D-E-F-G and receives a dollar. When $\lambda = 0$, Clare will only give credit to state F for the reward. She will not believe that any other states helped her receive the reward. However, if she repeated the same sequence again, she would then realize that state E got her to state F (which ultimately got her the dollar). If $\lambda = 1$ then Clare will believe that D, E, and F were all responsible for getting her that $1 on her first iteration.

E. Updating W

The algorithms in Sutton's experiments are methods of updating W to get better and better predictions. With each update, we update W like so: $W = W + \triangle W$. This update does not necessarily need to happen after each sequence, updates can be accumulated, stored and finally used once we are ready for an update. The timing of when we update W is the key difference between Sutton's two experiments.

When presented with a sequence, we can update $\triangle W$ at each step, t in the sequence like so $\triangle W += \triangle W_t$ such that so: $\triangle W_t = \alpha(P_{t+1} - P_t)\sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_t$.

Let's think of the right hand side of the equation side as two components. The first half of the equation, everything to the left of the summation, is the learning component. It tells us what we have learned in a given step of a sequence and how much of that learning we want to incorporate into $\triangle W$. From the summation onward, this is our attribution model. It tells us how much of our learning we should attribute to each state.

α plays the role of telling us how much of our learning we want to incorporate into $\triangle W$ (and ultimately W). Back to our analogy, imagine Clare performs her first random walk. At this point she doesn't know anything, so anything she learns is very important. However, after 1,000 random walks, she has a lot of experience, so new random walks should not be factored into her predictions very much. This would be the process of a decaying α value. However if α is constant, then each random walk is equally important regardless of our past experience. For constant and large α it makes sense intuitively why this will not converge. Essentially, we are biased towards recent observations and do not trust our past experience.

IV. Experiment 1

A. Training Data

For both experiments, 100 training sets were produced, each containing 10 random walks. The same 100 training sets were used for each experiment. No significant changes to the outputs were observed for different groups of training sets.

B. Description of Procedure

For experiment 1, Sutton describes the following procedure: each training set was presented to a range of TD learning procedures, each only differing in their λ and α values. The following update procedure was then applied to each λ, pair:

```
init vector, v to store W vectors
for each training set, T:
    init deltaW, W
    while not converged:
        for each sequence, s:
            for each time step, t:
                deltaW+=UpdateDeltaW()
        W+=deltaW
    v.append(W)
```

Such that the function UpdateDeltaW is equivalent to: $\alpha(P_{t+1} - P_t)\sum_{k=1}^{t}\lambda^{t-k}\nabla_w P_t$

The true value of each state is represented by the vector z and can be found analytically. For each λ, α combination, the Root Mean Squared Error (RMSE) is found. Then for each λ, the lowest RMSE among all α is selected and plotted.

C. Gap Analysis

There are several gaps in Sutton's explanation. Firstly, the range of α values is not explicitly stated. Secondly, he does not state whether he used a constant α or a decaying α (just that it is small). Thirdly, Sutton does not describe his convergence criteria. There are many different schemes, all involving the difference of W from one iteration to the next. Fourthly, he does not explicitly mention that there is no γ parameter to discount rewards. Fifthly, Sutton does not state his initial values of W for experiment 1.

D. Discussion of Results

The figure below is my attempt to replicate Sutton's experiment and reproduce Figure 3 from his paper.

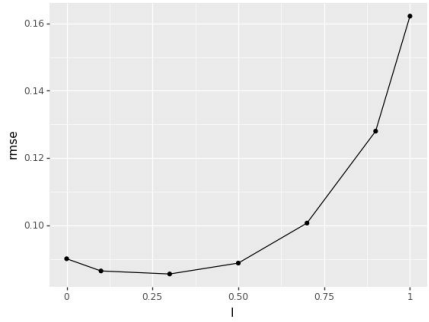

Fig. 1 Results from my replication of Sutton's first experiment. This should be compared to figure 3 in Sutton's paper.

Two differences are clear between my results and Sutton's. Firstly (1), the magnitude of my errors are smaller. Secondly (2), λ=0 produces a slightly higher error than λ=0.1 or λ=0.3 whereas Sutton demonstrates that his best RMSE value was at λ=0.

There are at least four causes of these differences. (1) My convergence methodology. A set was considered converged if d<0.05 such that $d = \sum |\, z - W \,|$. Sutton's methodology could have differed in the magnitude of d as well as the definition of d. Sutton may have squared his errors or taken a maximum instead of summation. (2) Initialization may have also impacted the results.

All states were initialized to 0.5 and while Sutton claims that initial values did not affect his results, this may not be true for my specific convergence criteria. Of course, the specific training sets (3) used could have been another source of error. 10 sequences per set and 100 training sets could be too small for complete reproducibility. The exact α values (4) used could also be a source of error.

The largest α value used in my experiment was 0.032 and it produced the lowest error for λ=0. However, α values larger than this did not converge. If I was able to successfully use larger α, perhaps this would have disproportionately decreased error for $\lambda = 0$.

Figure 1 is the result of using constant α values between 0.024 and 0.032. Values much smaller than 0.024 (e.g. 0.0001) produced larger errors for low λ values than for high λ. This is because the α was so small that the algorithm could not back propagate information to the middle states fast enough. In other words, small α meant that very small changes were made to $\triangle W$ after each iteration. Therefore, convergence criteria were satisfied with very little learning. Perhaps this could have been solved by tightening the convergence criteria.

α values larger than approximately 0.032 did not converge, though this varied by training set (which explains why Sutton used a range of αs). Intuitively, this non-convergence makes sense as explained in section III, E.

I did not limit the sequence length. This is because results with limited sequence lengths disproportionately reduced RMSE for $\lambda = 1$. When $\lambda = 1$ then for long sequences, our TD procedure will give too much credit to states visited early in the sequence even if they were close to the 0 outcome state, A.

Using a decaying α produced a curve with a much more consistent slope and less acceleration near λ =1. I believe this is because constant α introduces recency bias. With constant α, a sequence with many repeated states at the end of a training set can have much more influence on W than with decaying α. This is especially true for λ close to 1. Decaying α removes this issue and would decrease error at large λ values.

V. Experiment 2

A. Description of Procedure

In experiment 2, each training set was presented to the algorithm only once. Updates to W were performed after each sequence. At the end of a training set, the RMSE was calculated and W was reset to its initial values. After 100 training sets, the RMSE was averaged. This is repeated for several combinations of λ and α like so:

```
init vector, v to store W vectors
for each training set T:
    init deltaW, W
    for each sequence s:
        for each time step, t
            deltaW+=UpdateDeltaW()
        W+=deltaW
    v.append(W)
```

B. Discussion of Results

Figure 3 shows the results from the author's implementation of experiment 2. There are several differences between my results and Sutton's. Most notably, the RMSE for $\lambda = 0$ has a constant slope after $\alpha = 0.2$ in my results. However, Sutton's results show an increasing slope for larger α values. Additionally, for $\lambda = 1$, my results show a much faster increase in RMSE than Sutton as α increases.

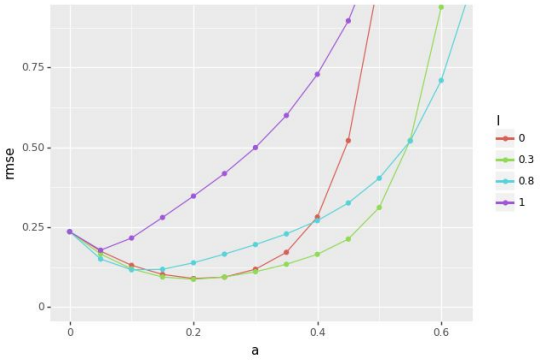

Fig. 2 Results from my replication of Sutton's second experiment. This should be compared to figure 4 in Sutton's paper.

There are many similarities between Sutton's results and the author's. Firstly both are consistent in the RMSE values for $\alpha = 0$. This comes as no surprise since $\alpha = 0$ is equivalent to no learning and thus W for all λ values are equal to their initial values. Secondly, λ values of 0 and 0.3 minimize RMSE at $\alpha = 0.3$. Thirdly, the general shape of each λ curve is approximately consistent with Sutton's figure 4.

The biggest difference from Sutton's result is that for $\alpha > 0.4$, the RMSE for λ values of 0, 0.3, and 0.8 increase at a much faster rate in the author's results. I believe the most probable difference in methodologies is that Sutton limited his maximum sequence length. Sequences with many repeated states may have increased RMSE for the author's results.

There were less gaps in Sutton's description of experiment 2 than experiment 1. He explicitly states his initial values and α values. However, there could have been a misinterpretation of his procedure or an error in the author's code. Potential sources of error include (1) Sutton updated W at each step in a sequence, not at the end of a sequence. However, this seems unlikely. (2) Sutton passed W from one training set to another whereas the author reset W after each training set. Again, this seems unlikely as training sets should be independent. (3) Training set generation could have been another source of error. If Sutton limited his maximum sequence length, minimum sequence length, or deviated from random sequence generation in any way, this could have impacted the results.

Figure 3 shows the author's replication of Sutton's figure 5. It was generated during experiment 2 (same run as figure 4). This chart shows the lowest RMSE achieved at each λ. The author's results suggest that a $$\lambda$$ value of 0.3 minimizes RMSE,similar to Sutton's. However, Sutton's results show a bigger difference between $$\lambda=0$$ and $$\lambda=0.3$$ than the author's results.

Nevertheless, these results are consistent with Sutton's that $\lambda = 1$ produces the highest error and $\lambda = 0$ does not produce the lowest error. An intermediate value is needed to take advantage of what both extreme's offer. $\lambda = 0$ does not back-propogate information fast enough whereas $\lambda = 1$ back-propagates information too quickly, consequently learning information that was actually due to randomness.

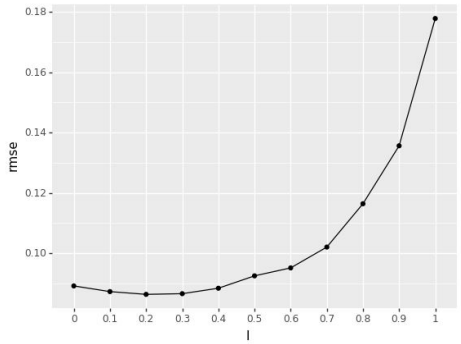

Fig. 3 Results from my replication of Sutton's second experiment. This should be compared to figure 5 in Sutton's paper.

VI. Discussion of Experiments

Both experiments demonstrate that $\lambda = 1$, or the Widrow-Hoff procedure, produced the highest error. The superiority of TD learning over Widrow-Hoff can be attributed to the fact that even if a positive outcome is achieved, not every state is equally responsible for that outcome.

Comparing figure 1 to figure 3 is interesting. Low values for λ performed worse in experiment 2 than experiment 1. This is because for small λ information is propagated backwards very slowly. This is not an issue when a sequence is reran until convergence. However, when W is reset after only 10 sequences it appears to be a big handicap.

References

[1] R.S. Sutton, Learning to predict by methods of temporal differences, Machine Learning vol. 3 no 1, pp. 944, 1988