Reinforcement Learning Project 1

Peter Lucia
plucia3@gatech.edu

February 16, 2020

Git hash: 89e14815f5127813eca35d58bf4f4917eed9e94d

1 Abstract

In this work, I explore and compare the performance of temporal difference learning to a supervised learning based approach (Widrow-Hoff) in an absorbing Markov process. Specifically, I replicate and analyze the results originally published by Sutton in [1] for the bounded random walk problem and also compare them to Figure 7.2 from [2]. A detailed description of the experiment is presented along with an explanation of the methods. The analysis focuses on how well the replicated results match the original figures, including descriptions of any significant differences, pitfalls, and justified assumptions that needed to be made.

2 Background

The experiment, originally conceived and published by Sutton in [1], provides strong evidence of how temporal-difference methods may offer advantages over traditional supervised learning methods. Temporal difference methods seek to gain information about the value of a particular state before a final outcome is reached. Temporal difference learning is analogous to a perceptual learning method of humans, who constantly update their hypotheses about what is seen or heard in streams of input over time [1]. Temporal difference methods may also be more useful than supervised learning methods when predicting future economic outcome. For example, economic forecasts are less useful when given all at once for an entire year, and provide more useful information when viewed as a multi-step prediction problem computed on a rolling basis [1]. Consider the game of chess, where the current state of the board is updated with each player's move. With a supervised learning approach, the value of a board state is provided relative to the final outcome of the game. This information is less useful to an agent playing the game in real time. A temporal difference approach, on the other hand, may incrementally update the value of each state as the agent plays the game, similar to the way a human player would react to changes in a board state. This technique, where estimates are updated based in part by other learned estimates without waiting for the final outcome is called bootstrapping [2]. Updating the value estimates at each state lowers the likelihood of misidentifying a bad state as a good state, reduces the memory requirements, and puts milder demands on the underlying computing hardware. Therefore, a temporal difference approach reduces the overall computation required for the agent to incrementally learn which next step offers a larger eventual reward in a multi-step, non-deterministic absorbing Markov process [1].

3 Methods

The primary focus of this work is to reproduce the results from the bounded random walk example, described in Section 3.2 of [1]. The game consists of five positions as well as two terminal positions. The five positions are adjacent to one another, and the losing terminal position exists at the very left, while the winning terminal position exists at the far right. The agent can only move either left or right, such that each state is only reachable from the state to its immediate left or the state to its immediate right. At each step, the agent randomly moves either left or right with equal probability until a terminal position is reached. When the agent reaches the rightmost state, the game is won and a reward of 1 is received. When the agent reaches the leftmost state, the game is lost and the reward is 0.

There are one hundred training sets, and each has a sequence of ten episodes. An episode contains all states in the path of the random walk from start to finish, where the terminal state is reached. Episodes can vary in length, and the random walk episode generating algorithm written for both experiments accepts a minimum and maximum length for each episode it generates. While Figures 4 and 5 examine different parameters, the assumption is made that the underlying learning procedure was unchanged between them in [1].

In order to reproduce the first experiment and recreate Figure 3, an algorithm was implemented directly from equation (4) of [1].

$$\Delta\omega_t = \alpha(P_{t+1} - P_t)\sum_{k=1}^{t}\lambda^{t-k}\nabla_\omega P_k$$

In this equation, α represents learning rate. A higher value for α will correspond with greater tolerance for error in the weights while a lower value for α will take more time to converge. The decay parameter, λ, affects how many previous predictions are updated at each observation [3]. The term $P_{t+1} - P_t$ represents the difference in probability of reaching the state $t+1$ from time t. The purpose of the summation is to exponentially weight changes to predictions occurring k steps in the past [1].

While the second experiment required restricting the size of the episode lengths to less than or equal to 15 steps, no minimum or maximum episode restriction was used in the first experiment. Additionally, for both experiments, equal probability was given to moving either left or right when generating the successive states of each episode. I implemented the following algorithm to generate bounded random walk episodes:

Algorithm 1 genSequence

```
seq ← [0,0,1,0,0]
currentPos ← 2
maxColumnIndex ← 4
running ← True
while running do
    move ← randomly choose Left or Right
    if move = Left then
        currentPos ← currentPos - 1
    else if move = Right then
        currentPos ← currentPos + 1
    end if
    if terminal state then
        append reward to seq
        break
    end if
    append current state to seq
end while
if invalid sequence length then
    return genSequence(minLength, maxLength)
end if
return seq
```

If $\lambda = 1$, TD(λ) is functionally equivalent to the Widrow-Hoff supervised learning procedure and future reward is weighted the most. As λ decreases, short-term reward is given more weight. Therefore, the advantage of testing various values of λ between 0 and 1 is that we experiment with a range of TD procedures that prioritize both short and long-term reward. The supervised learning approach prioritizes only long term reward, and if we find that there are lower overall RMSE when $\lambda < 1$, then a temporal difference approach is advantageous.

To replicate Figure 3 in experiment 1, I first implemented an algorithm based on eq. (4) from [1] without substituting e_t for the sum. The implementation takes the form of the following algorithm using the ideal probabilities $P = [\frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{5}{6}]$ to calculate the RMSE:

Algorithm 2 experiment1

```
rmses ← empty array
for i from 0 → numTrainingSets - 1 do
    vs = [1,1,1,1,1]
    sequence = episodes[i]
    for j from 0 → numSequences - 1 do
        errors = [0,0,0,0,0]
        deltaW = [0,0,0,0,0]
        episode = sequence[j]
        for state in episode do
            deltaW +=$\alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_\omega P_k$
        end for
    end for
    vs += deltaW
    rmse = $\sqrt{\frac{\sum_{i=1}^{5}((P-vs)^2)}{5}}$
    add rmse to rmses
end for
return mean(rmses)
```

To replicate Figures 4 and 5 in experiment 2, weight changes were computed incrementally. Given that the error, e_t, is the value of the sum in eq. (4) of [1], Sutton shows that

$$e_{t+1} = \sum_{k=1}^{t+1} \lambda^{t+1-k} \nabla_\omega P_k = \nabla_\omega P_{t+1} + \lambda e_t$$

Therefore, the following substitution of e_t for $\sum_{k=1}^{t} \lambda^{t-k} \nabla_\omega P_k$ was made

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

and allows for an incremental approach to the experiment 2 algorithm, again using the ideal probabilities to calculate the RMSE:

Algorithm 3 experiment2

```
rmses ← empty array
for i from 0 → numTrainingSets - 1 do
    vs = [1,1,1,1,1]
    sequence = episodes[i]
    for j from 0 → numSequences - 1 do
        errors = [0,0,0,0,0]
        deltaW = [0,0,0,0,0]
        episode = sequence[j]
        for state in episode do
            errors ← $\nabla_\omega P_{t+1} + \lambda e_t$
            deltaW +=$\alpha(P_{t+1} - P_t)\cdot$ errors
        end for
        vs += deltaW
    end for
    rmse = $\sqrt{\frac{\sum_{i=1}^{5}((P-vs)^2)}{5}}$
    add rmse to rmses
end for
return mean(rmses)
```

The original Figure 4 shows the RMSE across varying α for $\lambda = 0, 0.3, 0.8,$ and 1. Therefore, each execution of algorithm 3 ran over changing values of both λ and α, with the average RMSE for each α captured and plotted for each λ.

Because the original Figure 5 shows the RMSE using the best α, each execution of algorithm 3 was run over changing values of both λ and α, with the the lowest RMSE for all α captured for each value of λ.

4 Results

4.1 Figure 3

As is evident from Figure 3, the characteristic curve of the reproduced Figure 3 matches closely with Sutton's original. The lowest RMSE occurs at TD(0), while the highest RMSE occurs at TD(1) (Widrow-Hoff), suggesting an improvement in RMSE as the weight increment is determined by more recent observations [1]. There is some variation in the RMSE at each tested value for λ. For example, the Widrow-Hoff learning procedure, TD(1), yielded a RMSE of over 0.26, while Sutton's original result was just below 0.25. Additionally, at $\lambda = 0.5, 0.7, 0.9$, the RMSE in the reproduced version was roughly 0.01 greater than Sutton's original. One possible explanation for this is that the training set originally used to produce the results in 1988 was not the same as the one generated for this experiment. Because the training set had to be regenerated using only the original instruction from [1], some variation in the generated bounded random walk episodes, as well as the resultant RMSE, is to be expected.

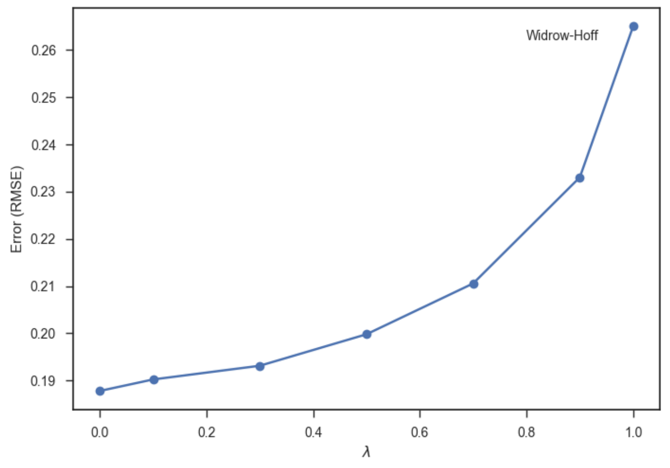

Figure 3: Replicated Figure 3

4.2 Figure 4

Overall, the replicated Figure 4 shown in Figure 4 matches closely with the original with several minor differences. For example, it shows a larger dip and a very similar characteristic uptrend of all the four λ values tested as α is increased, and exhibits a lower overall RMSE. The error curve for TD(0) crosses the error curve for TD(0.8), the Widrow-Hoff error is highest for $\alpha < 0.6$, and the curve for $\lambda = 0.3$ exhibited the lowest average RMSE for all α. To achieve similar characteristics as the original, it was necessary to restrict the episode lengths. Specifically, it was necessary to restrict the maximum sequence length to 15 steps or less. During the episode generation process, any episode generated randomly that fell outside of this range was discarded. This episode sequence restriction was not specifically mentioned in Sutton's original paper, but was necessary to dampen the quadratic error increase with increasing α for TD(0) and TD(0.3). While it is unlikely, we cannot be completely sure whether the training sets used in [1] were bounded in step size because they are not available.

The lower overall RMSE was an improvement to the original. Surprisingly, the TD(1) RMSE increased less rapidly with α than the original. At $\alpha = 0.6$, the TD(0) RMSE was lower than the original. In [1], it is suggested that the TD(0) error can be minimized by repeatedly presenting a sequence to the learning algorithm instead of only once, or by using backward propagation of prediction probabilities instead of forward propagation. In the replication of experiment 2, it was only necessary to apply a single presentation of the sequence to achieve lower error for TD(0) and TD(0.3) with a restricted episode length. It is intuitive that restricting the episode length would improve the RMSE of TD(0) because TD(0) uses only a one-step lookahead. With longer episode lengths, prediction values take longer to propagate back along the sequence. Therefore, by setting the maximum episode length to 15, prediction values would propagate back through the entire sequence within a capped amount of time, effectively reducing overall RMSE for approaches like TD(0) and TD(0.3), which have a shorter-step lookahead.

One difference between our result for Figure 4 and the original is that the lowest error occurred at $\alpha = 0.2$ instead of $\alpha = 0.3$. Also, the error for TD(0) was lower than the error for TD(0.3) for $\alpha = 0.25, 0.3, 0.35, 0.4, 0.45$. Additionally, the error curves for TD(1) and TD(0.8) both increase linearly for $\alpha \geq 0.15$, opposed to the quadratic characteristic of the original. One possible explanation for this is that we tested a limited number of values for α and λ to not only reduce the needed computation time, but also because I assumed no other values than the data points in the original figures were tested to produce the curves. This may have been an invalid assumption, and in future iterations of this work, we suggest testing a continuous array of values for λ and α, as this may produce more of the quadratic curve effect found in Sutton's original figures in [1].

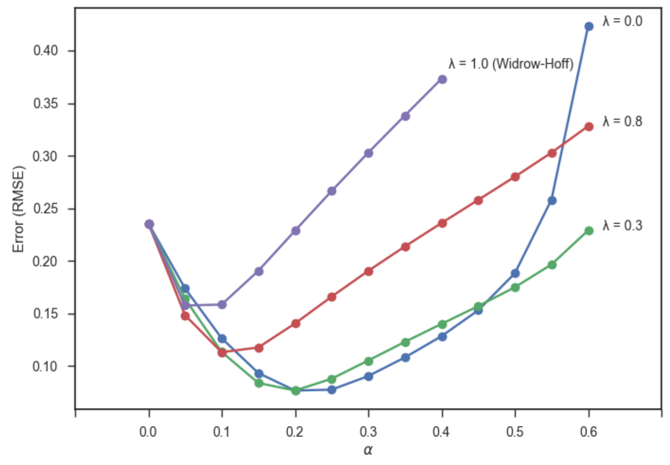

Figure 4: Replicated Figure 4

4.3 Figure 5

The reproduced Figure 5 shows the average error with the best α value against the random walk problem. Interestingly, in the replicated version, the error dip between $\lambda = 0$ and $\lambda = 0.5$ is not as pronounced as the original. However, the overall RMSE in this range is improved by about 0.02. In the Sutton Figure 5, the error at $\lambda = 1.0$ is greater than 0.2, while in our replication, the error is just under 0.16. One possible explanation for this is that our training set may have been less difficult for the learning algorithm because I set an upper limit of 15 steps on the episode length. Longer random walk episodes would be more difficult for the learning algorithm, especially for the shorter n-step lookahead inherent with lower values of λ.

Because [1] does not mention any differences in the underlying procedures used to make Figures 4 and 5, I assumed no changes needed to be made to the underlying TD algorithm. Therefore, while the runner functions exercise the underlying algorithm differently, the underlying algorithm to produce Figure 4 was unchanged to replicate Figures 5. I explored some alternative approaches that may have been used to produce Figure 5 that were not mentioned, such as adjusting how often weight updates occur and altering the minimum and maximum episode lengths, but ultimately the unaltered algorithm that was also used to produce Figure 4 produced a

Figure 5 that best matched the original in [1].

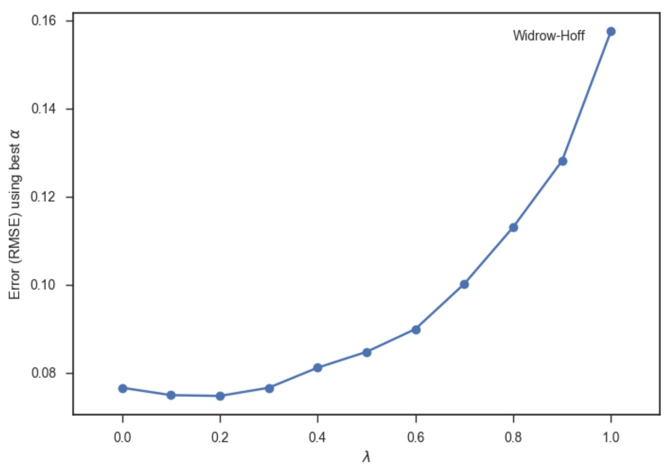

Figure 5: Replicated Figure 5

5 Discussion

In Figure 7.2 from [2], results are shown for a simple test of a larger random walk process. Specifically a 19-state bounded random walk is tested where rewards of -1 and 1 are given if the agent lands in the leftmost and rightmost states. This differs from our 5 state example, where 0 and 1 rewards are given at the leftmost and rightmost states. As the n-step lookahead increases, approximations are made of the full return with greater lookahead distance [2]. Interestingly, the result in Figure 7.2 show a similar trend to that of our replicated Figure 4. In our experiment, larger values of λ correspond with higher values of n in the n-step lookahead because with a greater trace decay we will backpropagate value estimates farther, which has the same effect of increasing our lookahead distance at each visited state. As we see in our example, high values for n and high values for λ exhibit a brief dip in error for low α, before quickly increasing quadratically (or linearly in our replicated version). This rebounding effect in the error curve is present in both the 19-state experiment as well as ours, and the lowest error in both experiments is found between $\alpha = 0.2$ and $\alpha = 0.4$. In Figure 7.2, methods with an intermediate n worked best, illustrating how generalizing TD and Monte Carlo methods to n-step methods can perform better than either extrema [2].

In additional RMSE, the mean standard error was captured for each of the experiments.

Experiment	Mean Std Error	Sutton Mean Std Error
Figure 3	0.002	0.01
Figure 4	0.008	Unknown
Figure 5	0.007	Unknown

Figure 6: Mean Standard Error Results

Comparing the results for Figure 3 with the original mean standard error published by Sutton in [1], we see improvement. Moreover, in all experiments the standard error was below 0.01, suggesting the results for each of the figures in this work are as significant in their replication as they were in Sutton's original work.

References

[1] Sutton, Richard S. "Learning to predict by the methods of temporal differences." Machine learning 3.1 (1988): 9-44.

[2] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

[3] http://www.scholarpedia.org/article/Temporal_difference_learning