Project 1: Desperately Seeking Sutton

Rodrigo De Luna Lara

October 1st 2018

GitHub last commit hash: **7c02f76df42fd29c3d9caf3743486907657ff94c**

The task for this paper is to try to replicate Sutton's results from his paper *Learning to Predict by the Methods of Temporal Differences* (Sutton 1988), specifically figures 3, 4 and 5 related to his experiments on the TD(λ) algorithm in a random-walk MDP outlined in Section 3.2 of paper. The Markov process defining this exercise can be seen in Figure 1

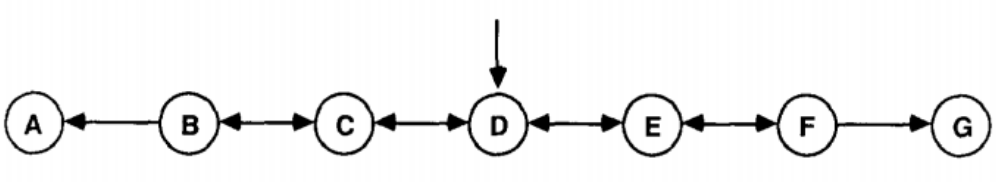

Figure 1: Markov Process for Random-Walk Experiment

The entry state is state D, and there are 2 terminal states on each side of the state diagram, states A and G. There is equal probability of going either left on right at any state. The rewards happen only in the terminal states, with state A having a reward of 0 and state G having a reward of 1.

Sutton used 100 training sets consisting of 10 sequences of random walks as defined by the Markov process in Figure 1, the construction of which was not explained in the paper. A function based on the following pseudocode was defined to create a random walk based on the aforementioned conditions:

```
Start at state 3 (D)
  While not in a terminal state
    Walk left or right (random choice)
    Append state to sequence
```

To simplify the representation of the problem, the states are renamed from their letter equivalent to an index, so state A becomes state *0* and state G becomes state *6*. The walk starts at *3* and then a random choice is taken of going either left or right, the walk continues until a terminal state is reached (states *0* and *6*). Sutton specifies that for all procedures, the weights were updated according to the TD(λ) algorithm shown in Chapter 6 of Sutton's book (Sutton and Barto (2018)):

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    $A \leftarrow$ action given by π for S
    Take action A, observe R, S'
    $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
    $S \leftarrow S'$
  until S is terminal

The function `td_lambda` in the code for this analysis is implemented based on this algorithm, receiving the values for α and λ according to the experiment as well as the sequence and the values vector.

For the first experiment, Sutton states that the weight vector is not updated after each sequence, but instead the differences in the weights are accumulated over sequences, the weight vector is updated only once the entire training set has been presented. He also states each training set was presented repeatedly to the learning procedure until it no longer produced any significant changes in the weights. Sutton doesn't mention any specific convergence criteria, nor specifically mentions what can be considered a *significant change* in the weights.

Sutton also mentions using the RMSE (root mean squared error) between the predictions from the learning procedure and the true probabilites for each of the nonterminal states ($\frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}$ and $\frac{5}{6}$ for states *B*-*F* or *1*-*5*). Finally he states using several for λ such as $0, 0.1, 0.3, 0.5, 0.7, 0.9$ and 1.0. As for the value of α sutton only mentions that it's *small*. Based on this description by Sutton, for the first experiment, it was coded in Python based on this pseudocode:

```
seeds = 69, 13
alpha = 0.01
lambdas = 0, 0.1, 0.2, ..., 0.9, 1

Loop for each seed:
  Loop for each lambda:
    Loop for each training set:
      Set values to 0
      Loop until convergence (innovation < 0.001):
        Set value of terminal states (s6=1)
        Loop for each sequence:
          Update values with TD lambda
      Calculate RMSE
```

Figure 2 shows Sutton's result alongside the result from this analysis with two different random seeds based on the above conditions. As Sutton explains, the performance improves rapidly as λ is reduced, and is best when $\lambda = 0$. So both results support this hypothesis. It is evident however that the error is significantly different with respect to Sutton's results, this could be due to the training sets generated. The two seeds that were used result in very different error magnitudes, although the behavior with declining λ remains the same.

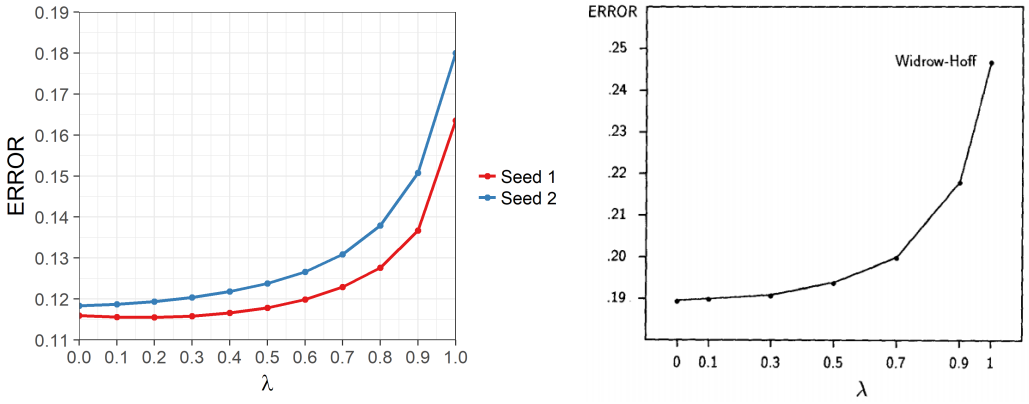

Figure 2: Results for Experiment 1

For the next experiment, Sutton explains that unlike the first experiment the training set is presented just once to each procedure, and the weights are updated after each sequence rather than after each training set. Additionally, the learning procedure is applied for several values of α for each value of λ. Finally, the weight vector is initialized to 0.50. Sutton uses $\lambda \in [0, 0.3, 0.8, 1.0]$ for the figure of the results.

Figure 3 shows the results for experiment 2 that Sutton obtained and the ones obtained for this report with the different seeds. The following pseudocode outlines how this was implemented:

```
seeds = 69, 13
alphas = 0.05, 0.1, 0.15, ... , 0.55, 0.60
lambdas = 0, 0.05, 0.10, ... , 0.95, 1

Loop for each seed:
  Loop for each alpha and lambda combination:
    Loop for each training set
      Set values to 0.5
      Loop for each sequence
        Set value of terminal states (v0=0, v6=1)
        Update values with TD lambda
      Calculate RMSE
```

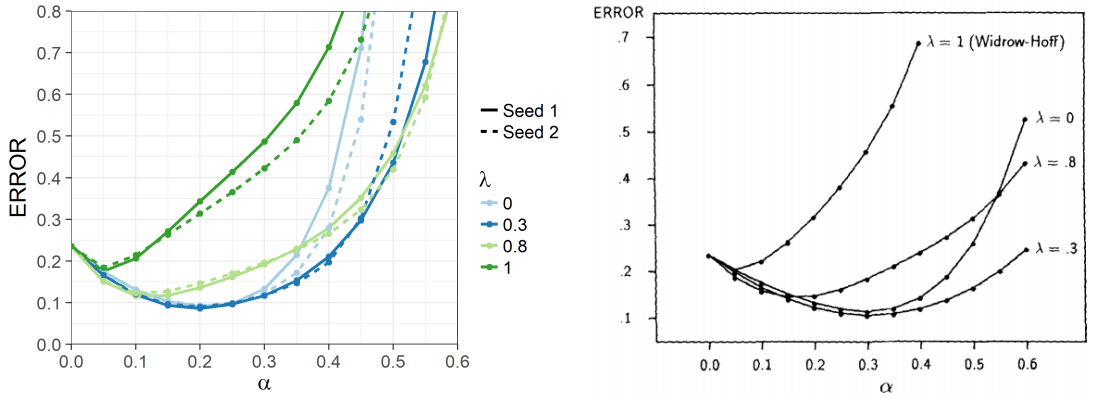

Figure 3: Results for Experiment 2

As Sutton states, the value of α has a significant effect on the performance, with the best results having intermediate values of α. Additionally, when $\lambda = 1$ the performance is worst in all cases, with lower values of λ being preferable. It can be seen here that the behavior with respect to α and λ holds true with respect to Sutton's paper, the best overall results being when $\lambda = 0.3$, although with a lower α around 0.20 instead of 0.30, also the error tends to overshoot sooner than in Sutton's results. As with the previous experiment, the randomness in the generation of the training data has a significant and visible effect on the performance, specially when $\lambda = 1$.

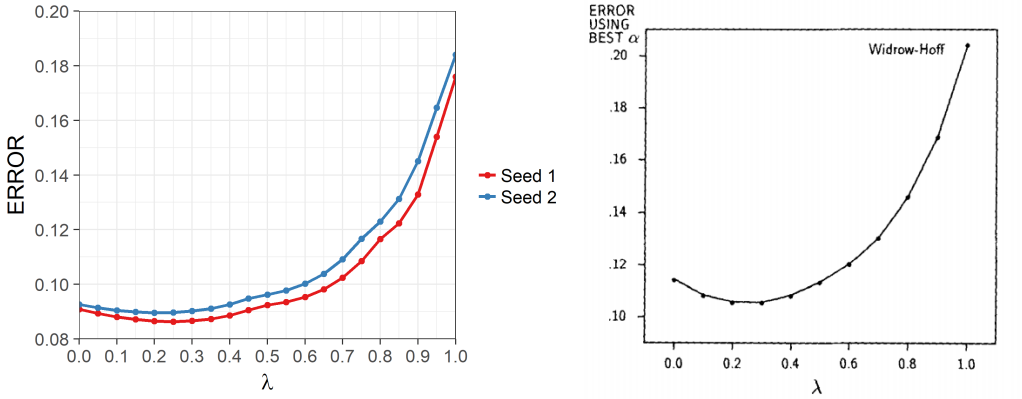

Figure 4: Best error for each lambda

Then Sutton plots the best error level for each λ, Figure 3 shows the results for this analysis as well as his paper's results. As Sutton mentions the best value for λ is around 0.3, which is also the case in the reproduction.

Having the comparison of all results with respect to Sutton's in figures 1, 2, 3 we can see that they are very different numerically, although the same conclusions about TD(λ) can be reached from both as explained in each case above.

Some of the reasons why there could be a mismatch between the results in this analysis and Sutton's may include:

1. Difference in the generation of random walks

As shown with the 2 different seeds used for generation of the training sets, the magnitude of the error is affected by the seed. Sutton doesn't provide enough information regarding how he created the random walk sequences, so the results are not exactly reproducible from that point on.

Figure 5 and Figure 6 show the distribution of the states and the length of the sequences for each seed. They have practically the same distributions in both and yet the error in figures 2, 3 and 4 is very different. They only differ in that when the seed is 13 the states tend to be slightly more to the right in the space of the walk, which is near the terminal state with the highest reward. So perhaps when the training sequence tends to reach the right side more than the left side the error is reduced, maybe Sutton's random walks weren't that random and had a tendency to go to the left side where the reward is lower.

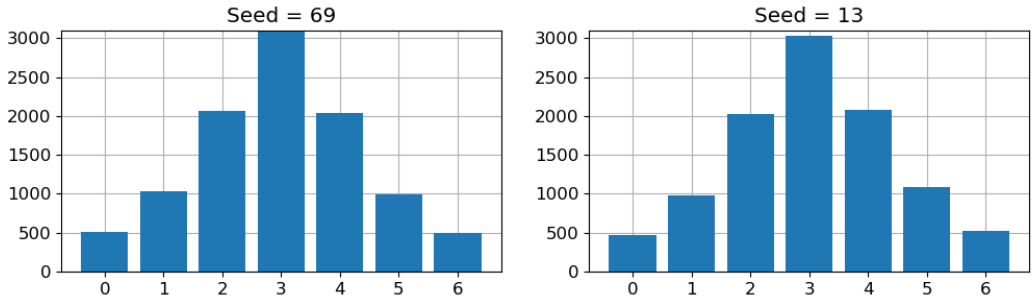

Figure 5: Distribution of Steps for Training Sets

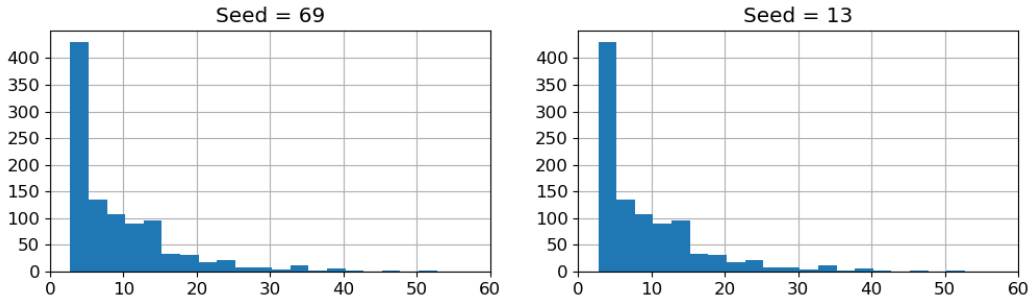

Figure 6: Distribution of Sequence Length for Training Sets

As per the length of the sequences, most of them seem to be short with less than 10 steps, there are just a handful of sequences in which the walk seems more like wandering around aimlessly within the central part of the space defined for the walk without reaching a terminal state. In both seeds there are walks that take between 30 and 40 steps. It is also interesting that despite the *randomness* most walks tend to take very few steps (less than 5) to reach a terminal state. To evaluate reproducibility, the exact training sets used by Sutton would have to be used.

2. Difference in the convergence criteria

Sutton doesn't mention some details on his implementation, as mentioned before he just says α is small for experiment 1, he doesn't provide a specific value. Also he doesn't mention exactly how he's checking for convergence in the same experiment, for this analysis the innovation is calculated as the sum of the absolute value of the weights with respect to the previous iteration, and a threshold of 0.001 is set. Sutton doesn't mention what criteria he's using to stop the training. He also doesn't mention that for high values of α there is divergence during the training.

Conclusions

The experiments in Sutton's paper are not exactly replicable, due to a lack of information provided by Sutton, like exact values for some parameters (specially α), as well as a lack of a more detailed explanation on how he considered convergence in the first experiment. Nonetheless, even if the results don't match exactly the insights they provide on TD(λ) are the same.

In this regard, TD(λ) shows preference for lower values of λ and α, as evidenced by Figure 4. For the problem analyzed (the random walk), higher values of α caused divergence in experiment 1. The preference for lower values of λ is indicative that for this problem propagation through the sequence should not be very fast, but also not too slow as $TD(0)$ shows slightly higher error.

TD(λ) seems like a very strong reinforcement learning algorithm for problems which involve temporal sequences, like the random walk exercise, and from the algorithm implemented in this analysis, are computationally cheap and efficient. Sutton states that temporal difference methods appear to learn faster than supervised learning methods, which seems an appropriate conclusion based on the complexity of some supervised learning algorithms with respect to the TD(λ) algorithm proposed by Sutton.

Sutton provides simple experiments, such as the one evaluated here for the random walk, that highlight the benefit of TD(λ) and the effect of its hyperparameters in learning. Sutton also suggests that temporal difference methods could be applied to other more complex problems like speech recognition, where learning could happen at the same time as processing. Temporal difference methods adjust predictions as each step is presented instead of once the final reward is attained, unlike other methods like Monte Carlo. This philosophy of updating estimates as new information comes in seems really powerful, and being model-free algorithms, the claim that Sutton makes that they can be applied to a wide variety of problems doesn't seem far-fetched.

References

Sutton, Richard R. 1988. "Learning to Predict by the Methods of Temporal Differences." *Machine Learning* 3: 9–44.

Sutton, Richard R, and Andrew G Barto. 2018. *Reinforcement Learning*.