An exploration of Sutton - 1988 - Temporal Difference Learning

Sam Farren
February 18, 2018

Introduction
Sutton's 1988 paper on Temporal Difference Learning (TD) opened the door to other possibilities outside of supervised learning approaches in the machine learning community. He states that the TD methods used prior to his paper weren't fully understood, and that his paper was to help show applications where TD could be utilized and offer a proof to these applications where they perform better than traditional supervised learning methods at the time. This paper offers a replication of Sutton's experimental results on a specific problem called a "Bounded Random Walk" as well as an in depth analysis as to why Sutton's experiments are so important.

Initial Problem Analysis and Approach
The problem presented in Sutton's paper consisted of a dynamic system with two terminal states: A and G, as well as five states in between. The entire system was essentially: A-B-C-D-E-F-G. The agent always entered at state D for each sequence start, and ended in A or G. Landing in G gave the agent a reward of 1, and 0 for every other state. There was a 50% probability of going to either the left or the right for each action no matter where the agent was. This will be discussed later, but it is worth mentioning this inherent probabilistic structure of the system. It is key for the performance of TD, as it essentially allows state values and estimates to propagate out over time. To produce data for training, 100 sets of 10 random walk sequences were produced and this same data was used for figures 3,4, and 5. However, each of the three figures shows how TD can be altered in order to improve or diminish performance. Specifically with the rate of learning used, the chosen value of lambda, and how training is performed. I believe Sutton structured his paper in a specific order to get these main points across. These alterations and comparisons are covered below.

Generalized Pseudo-Algorithm for Figures 3, 4, and 5
As a side note I would like to briefly discuss a generalized algorithm for how RMS errors were produced for each graph. The algorithm implemented for figure 3 was as follows with differences in the algorithm for figures 4 and 5 discussed below it:

(0)*Initialize weight vector to .5's, alpha=.01,lambdas=[0,0.1,0.3,0.7,0.9,1]*
(1)*For each Lambda:*
(2)For each train set in train data:
(3)While not converged:
(4)For each sequence in train set:
(5)For each time step in sequence:
(6)Calculate Pt and Pt+1 and $\Delta \boldsymbol{w}$ using *Pt and Pt+1:*
*(6a)Pt = transformed weight vector * Observation vector at time step t*
*(6b)P t+1 = transformed weight vector * Observation vector at time step t+1*
(6c)Calculate $\Delta \boldsymbol{w}$ *from equation above where alpha = .01 and the summation being a k-step estimator through time t of all previous observation vectors with current selected lambda*

(4)After entire training set presented, update weight vector and check previous Δw with current Δw for convergence. This is true if Sum((Δwt-Δwt-1)^2)<.000001(assumption made here)

(4)If true: Return weight vector

(3) Calculate RMSE for this specific train set and d*o with all other training sets in Train Data*

(2)Average all RMSE's and get single Averaged Error for lambda value, this is plotted

(1)Plot RMSE's for each lambda

While this is for figure 3, there is a slight variation in the algorithm for figures 4 and 5, where the weight vector update ($w = w + \Delta w$) is moved inside to step 6 (this is the incremental approach) and there not being a convergence check. In addition, additional alphas are provided to calculate RMSE's across varying learning rates, so there ends up being another loop inside the lambda loop for each alpha. I should also note that alpha was chosen to be .01. All Sutton said was that alpha had to be small for the change in weight to converge. This was an assumed value in my reproduction to receive the appropriate errors.

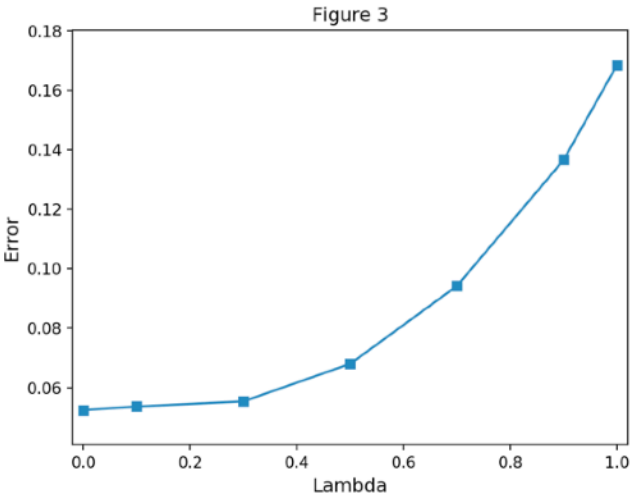
Figure 3

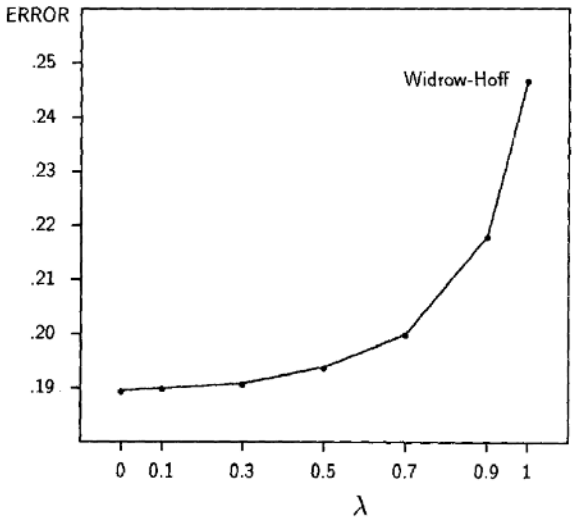
The main point Sutton wanted to demonstrate with the third figure was the idea of how differences in training approaches can affect the error of predictions. He was essentially simulating more noisy data with this approach. In particular, figure 3 used a training process called a repeated presentations training paradigm where a sequence was repeatedly presented until convergence of Δw was reached, in my case, until the sum of the squared differences of Δwt and $\Delta w(t\text{-}1)$ was < .000001 where

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

This was one of the assumptions I made for convergence of the weight vector. I compared the previous weight change value with the current weight change value to see if there was a minimal difference. This type of convergence check on training sets allows for all states of that sequence to propagate it's value to the entire environment(all other time steps within the sequence). Most interestingly is with lambda=0 because this is a one step look ahead. Therefore with repeated presentations, eventually the value of the end state will propagate toward the very first state in the sequence, since multiple iterations are performed, thus allowing weight changes in the beginning of the sequence.This allows the value of state G, if encountered, to propagate value all the way to all other states in the sequence. Take for example the sequence: D-C-B-C-D-E-F-G. Repeated presentations of this would lead G to propagate value to B, which we know from the ideal predictions, B has a very low probability of getting to the reward state (G). While this can still happen in TD(λ) with incremental updates, the effect a sequence like this has on B is less noisy. For this case, B is the closest to G it can be: five time steps away, so $\lambda^{t\text{-}k}$ becomes λ^5 giving the weight a much lower value and reducing the noise of a B leading to a G state and obtaining reward. In contrast with repeated presentations, B is given more weight than it should and leads to a larger error and creates more noise. This is why errors for repeated presentations in figure 3 is greater than figure 5 with an optimal lambda. The optimal lambda gets rid of a lot of the noise created from bad sequences and leads the weights to the true dynamical probabilistic structure of the system.

Comparing results between Sutton's figure and mine, you will notice a similar shape however each point is shifted in the negative y direction by about .13 at TD(0), and .06 at TD(1). One

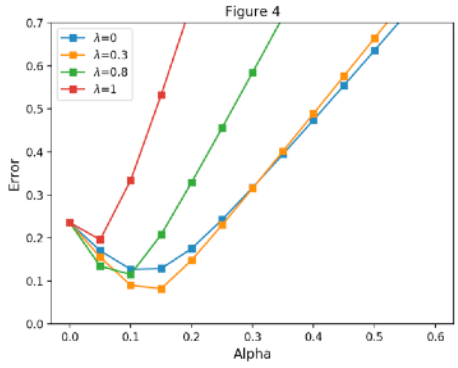


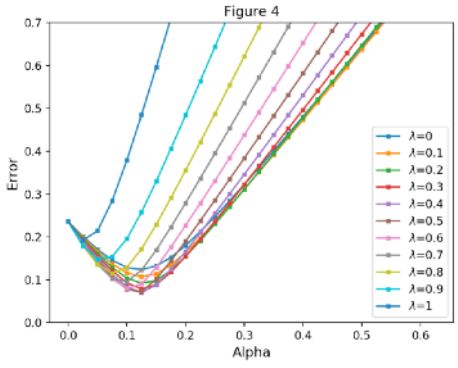


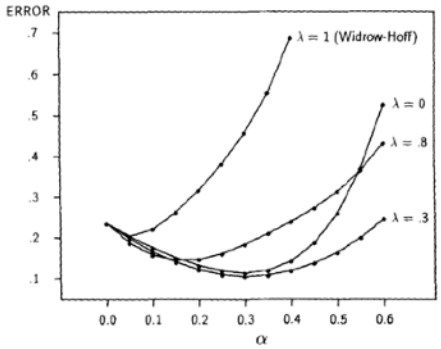
possible cause for this is due to increased floating point precision with modern architecture of computers as to Sutton's limitations on a 32 bit system back in 1988. Another one may lie in my assumption of what he actually meant by convergence in each repeated training presentation. I assumed it meant the sum of $\Delta \boldsymbol{w}$ over all states didn't change (converged) but Sutton could have meant the change in RMS error between repeated presentation, or difference in error in general. There was some gray area that needed to be experimented with.

Figure 4

For figure 4, Sutton demonstrated the importance of a learning rate for TD implementations. In addition to varying alpha rates, training was done differently as well. An incremental TD approach was used rather than a repeated presentations method. More specifically, the weight vector was updated after each sequence rather than each training set. Therefore, the associated alpha and lambda chosen had a much larger affect on the error. The same equation as above was used to calculate $\Delta \boldsymbol{w}$. You will notice that larger lambdas, specifically λ=.8 and λ=1, resulted in worse errors relative to smaller lambdas. Again going back to the previous figure in reference to noisy sequences, larger lambdas don't decay as quickly when summing over the previous observed sequences for the change in weight. Therefore, observed sequences retain more weight as time passes along in the sequence which can create more noise.



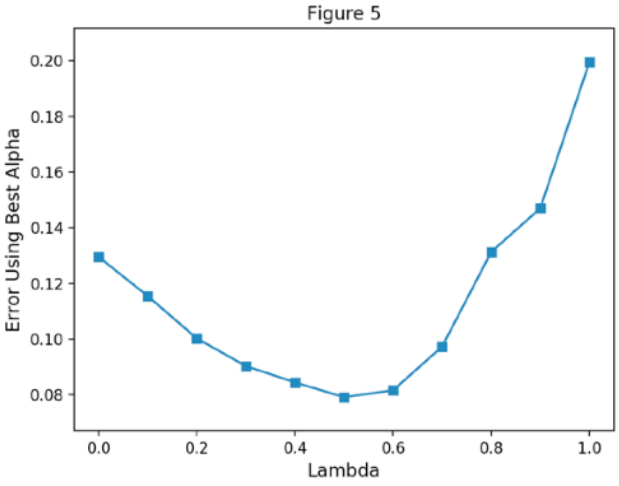


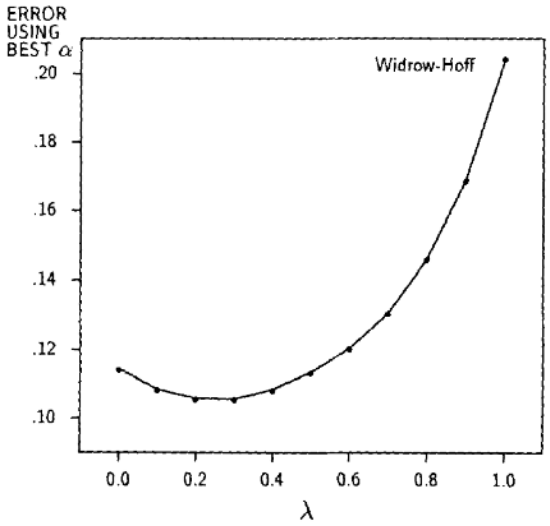


Comparing the graphs for figure 4, you will notice alpha=0 start in the same spot as Sutton, however mine start to exponentially increase at a much lower alpha, around .1-.2 for every alpha as compared with Sutton where alphas between .8 and .3 yielded much slower and steadier increases. My errors were also not spread out as far, and shifted to the left.

Figure 5

For figure 5, Sutton wanted to show the optimality of what temporal difference learning was capable of. He was basically saying, once you figure out an optimal learning rate, you can then find an optimal lambda that filters noise, but retains enough weight to converge toward the true structure of the system the agent is interacting with. In order to do this, the same algorithm was run as in figure 4, but the alpha that produced the smallest RMS error averaged over the training sets was used for each lambda iteration. For my experiment, those values relative to the lambdas in increasing order are: [0, 0.15, 0.15, 0.15, 0.15, 0.1, 0.1, 0.1, 0.1, 0.05, 0.05]. This can be seen at the minimum of the graph in the figure 4 charts, where most lambdas had a minimum error around .1-.15. The shape of each graph is similar however my lowest error was achieved with λ=.5 while Sutton's lowest error was achieved with λ=.3.





Why RMS Error

It should be noted that rather than just taking the difference in probabilities between ideal predictions and estimated predictions, Sutton chose the Root Mean Squared Error. This means he wasn't just trying to show the error, but also how far the magnitude of the error was spread out from the true prediction. Averaging the square of the residuals and taking the square root will show this spread and is one of the more popular metrics used in statistics.

Conclusion

While my results weren't perfect, in order to get an exact reproduction of Sutton's experiments, a lot of assumptions had to be made, and with the advances in technology compared with 1988, mostly in relation to figure 3, a more precise convergence was able to be calculated. All in all this paper helped the machine learning community make a lot of progress and push for other approaches outside of supervised learning methods.

Citations:

Sutton, R.S. Mach Learn (1988) 3: 9. https://doi.org/10.1007/BF00115009xs