Temporal Difference Learning - Sutton 1988

Hermann Hans
Department of Computer Science, Georgia Institute of Technology

February 12, 2017

Abstract

This report attempts to reproduce several of the findings that were presented in Richard S. Sutton's paper, "Learning to Predict by the Methods of Temporal Differences" (1988)[1]. In Sutton's paper, he argues that *temporal-difference methods* are superior to supervised learning methods in terms of memory, computation and accuracy for a lot of problems where both can be applied. Our reconstruction of his findings in sections two and three come close the same results from back in 1988, although minor differences do exist in all cases. A video with a quick overview of the reconstruction of these findings is available at `http://youtu.be/SJRjnWyMTw8?hd1`

1 Bounded Random Walks

Before diving into TD(λ) we quickly reiterate the simple dynamic system that Sutton uses for the rest of his paper. A *bounded random walk* is defined with states A through G, where every sequence starts in the middle state D, has a probability of 50% to move either left or right in all states, and terminates if it reaches either state A or G. The outcome of each sequence is then defined to be either 0 if it ended on the left (A) or 1 if it ended on the right (G). Using this model, TD learning methods were then applied to estimate the probability of a right-terminated outcome for each of the middle states (B through F).

Every sequence was encoded as a list of observation vectors x_i where every position was 0 and 1 at the i^{th} position. Sutton did this only for the middle states and encoded the result separately as z, where z is either 0 or 1, depending on left or right side termination of the sequence. In our implementation we actually encoded the entire walk (all seven states) in the sequence. The reasoning behind this is that the results will still remain the same as the weights of the terminal states never change. The *ideal predictions* for states B through F are then $\frac{1}{6}$, $\frac{1}{3}$, $\frac{1}{2}$, $\frac{2}{3}$ and $\frac{5}{6}$ respectively. These are the values that were then used to calculate the root mean squared (RMS) error.

2 Implementing TD(λ)

Sutton defines TD(λ) as the following function:

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

where P_t is a prediction at time t depending on the observation vector x_t and a weight vector w, α is the learning rate, and $\nabla_w P_k$ is the vector of partial derivatives of P_t with respect to each component of w. For our *bounded random walks* system it is enough to consider the simple case where P_t is a linear function of x_t and w, such that $P_t = w^T x_t = \sum_i w(i)x_t(i)$. Taking the partial derivative $\nabla_w P_k$ with respect to w also leaves us with x_k for our model. In (1) we calculate the change in weights, which is then used in conjunction with the update function of w:

$$w \leftarrow w + \sum_{t=1}^{m} \Delta w_t \tag{2}$$

which is used differently between the first and subsequent experiments by Sutton, but we'll get to that in the sections below.

3 Experiment 1 - Figure 3

A *repeated presentations* training paradigm was used by not updating the weights after each sequence like described in (2), but rather by keeping a running total of the δw and only adding it to w after a full training set (10 sequences) has been presented. This process is repeated until the vector w has converged and only then returns the predicted results.

In Figure 1 we can see that our implementation of the repeated presentations very closely matches what Sutton describes. We get the same monotonically increasing function and the relative changes from subsequent λ seem to line up as well. Our RMS error however is $\approx$.08 lower when compared to the original plot. One possible explanation that we didn't further explore was the convergence function and the value of α. The former is responsible for when to actually terminate the repeated presentations process, which Sutton only describes as "no longer [producing] any significant changes in the weight vector". It's possible that Sutton's interpretation of "significant changes" is an order of magnitude higher, citing available computing resources back in 1988 as the most plausible explanation of why this could be a factor.

Along the same lines is the chosen value of α, which is also only briefly mentioned that "for small α, the weight vector always converged." We settled on $\alpha = 0.01$ after trying smaller values which seemed to converge to the same results. No exhaustive testing was done on the value of α but even at 0.1 already showed issues where our algorithm no longer converged.

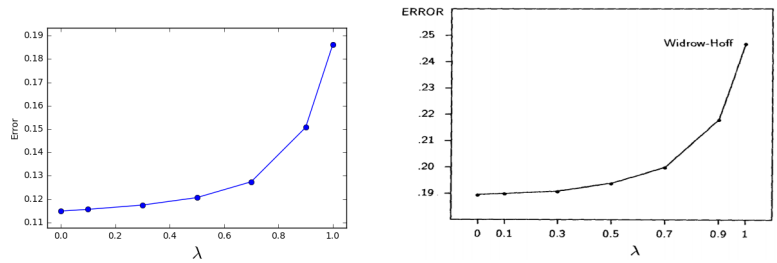

Figure 1: Comparison of average RMS error on random-walk. Our result [LEFT] Sutton 1988 [RIGHT]

4 Experiment 2 - Figure 4 & 5

In the second experiment the weight update function (2) is no longer applied as a batch update after a full training set has been seen. Instead, the weights are updated incrementally (immediately if you will) after each sequence. Also, a training set is now just presented once to the learner instead of repeatedly until some convergence property is met. In Figure 2 we plot several curves of RMS error vs. α for TD(λ) 0, 0.3, .8, and 1.

Figure 2 again shows the results of both our implementation and that of Sutton. For $\alpha = 0$ we seem to get the same value as Sutton for all λ. Looking at (1) this should be fairly intuitive since we are setting our learning rate to zero. For the case of $\lambda = 1$ our function follows a similar RMS error, but diverges slightly faster than Sutton. At $\alpha = 0.35$ we are already above the .7 RMS error cutoff that Sutton used in his plot. The other λ values follow similar shapes, but we are not seeing the crossover between $\lambda = .8$ and $\lambda = 0$. In fact, our $\lambda = 0$ performs slightly better than $\lambda = .3$!

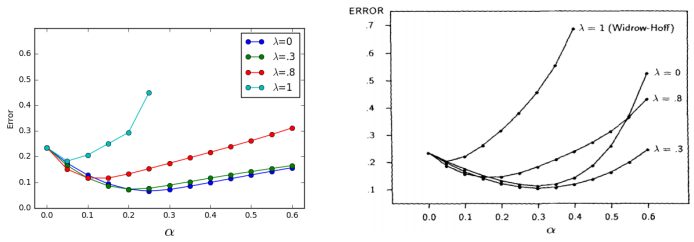

Figure 2: Comparison of learning rates for different α. Our result [LEFT] Sutton 1988 [RIGHT].

Across the board (with the exception of $\alpha = 0$), similar to what we saw in experiment 1, our RMS error is slightly lower as well.

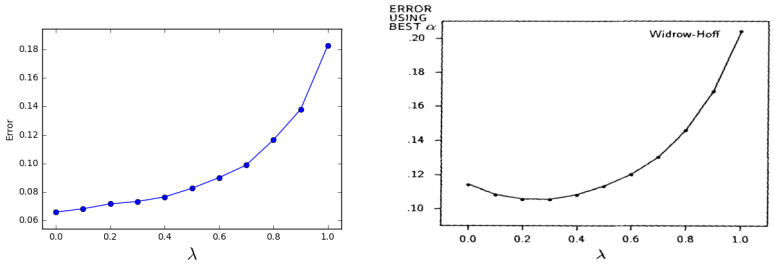

Figure 3: Comparison of RMS error using optimal α values. Our result [LEFT] Sutton 1988 [RIGHT].

Figure 3 then takes the same approach as described above, but instead of plotting the RMS error of the α values for each λ, we keep track of the best α value for each λ. Using this optimized parameter set, we then run the best α value for each λ for the same λs in experiment 1. Comparing our result to Sutton, we see however that instead of a having a slight dip and best performing at $\lambda \approx 0.3$, our result still looks similar to what we saw in the first experiment; a monotonically increasing function.

5 Conclusion

For experiment 1 we already hinted at possible differences between our results and those presented by Sutton. While in general we get very similar results, in Figure 3 especially, we see the biggest deviation. The reasons that were already cited likely factor into these differences as well. Another factor to consider is that of data types. Our implementation used Python, Numpy, and 64bit float datatypes. In 1988 it's likely that Sutton was only able to run these calculations on a 32-bit architecture, limiting his floating point precision as well. With the cumulative nature of the functions above, deviations in precision would add up quickly and also explain the greater differences in RMS error that we saw.

Other possibilities that we didn't further explore were the initial weight vectors and random training set generation. While Sutton mentions them explicitly to be 0.5 for experiment 2, no such mention is made for the experiment 1 (we used 0.5 as the initial weight vector everywhere). The latter point could also be related to computing power and available random number generators in 1988. It's possible that even with 100 training sets of 10 sequences each that there wasn't enough distribution to get good results.

To sum up, it seems that our results are strongly correlated with what Sutton described in his paper and that the deviations that we've seen could be more related to the available computing resources than the assumptions that we have made in our implementation.

References

[1] Richard S. Sutton. Learning to predict by the methods of temporal differences. In *MACHINE LEARNING*, pages 9–44. Kluwer Academic Publishers, 1988.