TD-λ: Methods of Temporal Difference (Sutton 1988)

Chenxi Yu

Saturday, Jun 11,2017

1. Introduction

TD-λ, as a reinforcement learning algorithm, was implemented by Sutton in late 1980s to prove its superior efficiencies and accuracies in predictive learning, compared with conventional supervised learning method. By replicating TD-λ experiments on a bounded random walk problem in his paper, I managed to reproduce his experiments results and the performance of the algorithm. Below covers both batch learning and sequential weight updates by using TD-λ.

2. Experiments Overview

The data generated to experiment on the bounded random walk interconnected state model is based on a fixed locations, namely, A-B-C-D-E-F-G, where has A and G as the terminal states. While there is a 50% of chance on moving towards A or G, state A and G reward 0 and 1 respectively. Our primary goal is to quantify the probability expectation for states from B to F. For implementation simplicity, I synthesized the data of 100 training sets of 10 sequence based on directions of each move with the first letter set as D where is the initial location of all sequences(i.e. 'DAGGGG', 'DGGAGG').

Below are the TD equations (Sutton 1988) used for experiments. While the former is for weights update, the latter is for sequence learning:

$$w \leftarrow w + \sum_{t=1}^{m} \Delta w_t \qquad (1)$$

$$\Delta w_t = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k \qquad (2)$$

The standardized vectors X = [x_1, x_2, x_3, x_4, x_5] are used to represent the observed vectors x for states from B to F due to their mutual linear independence. Thus, I use [1/6, 2/6, 3/6.4/6, 5/6] to represent the true values vector on expected values for states from B to F respectively.

The value function is noted as: P = x_i*w (for simplicity, I have all the small case letters as vectors). The P function also fulfills that delta_t*P_t = w_t while I have n as the t in the equation and P_t as z. In the paper, z = 1 at state G and z = 0 at state A. I believe the above implementation takes P_t as the reward at time = t. Additionally, w is initialized to .5 for all components and λ is indeed a factor accounting for the exponential leverage on states updates like the gamma in the Bellman equation. TD-λ helps update w through observations, as the value function P(x_i,w) measures the expectation for every state during iterations. The RMSE between the predicted value P and the true values is used as the accuracy evaluation indicator to all of our models.

3. Batch Learning

Figure 3 in Sutton's paper uses batch learning updates for TD-λ. After I iterate each sequence in a training set, I use (2) continuously between steps and keep a running total of delta_w which carries over between sequences. Next, the weight update (1) is used only after the entire training set has been observed. Before the update, I continuously recheck if the weight vector reaches convergence. However, how to define "convergence" is not specified in the paper. In statistics, given some $\varepsilon > 0$, the weight vector converges if ||delta_w_t||< ε as t approaches to infinity. For this experiment, I have ε as .005. Sutton claims that all small alpha should converge to the same result, yet while I replicate his result, I find it challenging to pick an alpha that converges in a short time. After a long time exhaustive manual search, alpha = .015 as our ultimate result. I also discovered that the sum of all delta_w_t over sequences may not sustain alpha larger than .1 having individual component in w vector smaller than 1 and larger than 0.

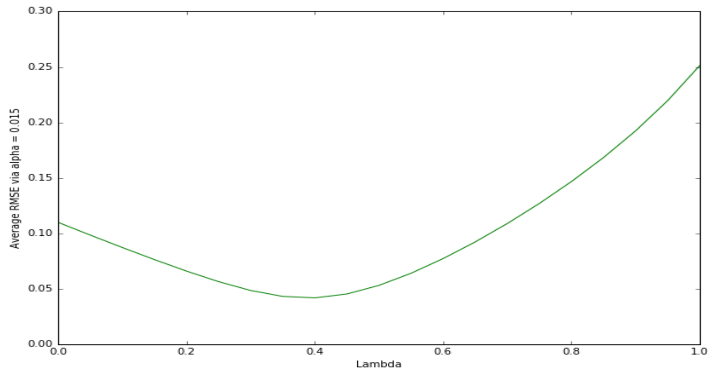


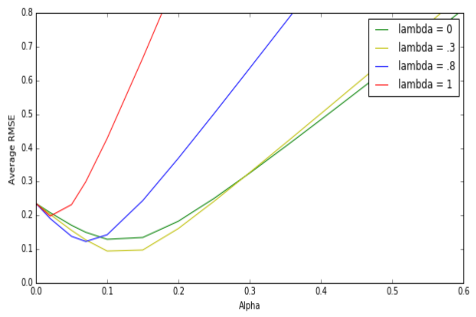
Figure 1: Average RMSE after learning training data based on repeated presentation

My result in figure 1, paralleled to figure 3 in Sutton, are similar. However, it seems that my plot seems work better than Sutton's since the convex shape and the monotonically increasing pattern of average RMSE are achieved, yet the convergence point in my result is way lower than Sutton. Compared with the Errata's result, I believe mine is more reliable because only the end state's probability change is not sufficient to have a small enough average RMSE. Thus, I suspect if the result shown in the errata exhibits a convincing graph shape for our reference. Finally, when replicating the experiment in python, mathematical operations are more precise. Yet, there is another noise factor in plate—the random generated data in the program. Though I do not close examine the effects caused by this aspect, I observe some trivial difference as data generated from time to time.

4. Sequential Weight Updates

Rather than performing update after complete representing the training data, sequential weight learning accumulate weights and perform a fresh weight update(1) after iterating each sequence in a training set. There is no convergence in guarantee in each training set. Thus, it is not uncommon to have exponential growth on weight vectors as larger alphas is expected.

It is critical to initiate each component as .5 in the weight vector since every sequence is learned for once. Sequential weight updates with various alphas and λ led to similar results to Sutton's. As alpha increases, λ =1 grows the fastest in error, followed by λ =.8 and this is where differs from Sutton's whose second fastest is λ = 0. However, λ =.3 still performs the best for alpha between .1 to .25 though is different from Sutton's results. My figure 2 grows quicker than Sutton's, yet this may be explained by a difference in training data sets. Nevertheless, I may have missed something in codes that have the lambdas = .3,.8 and 0 different from Sutton's.



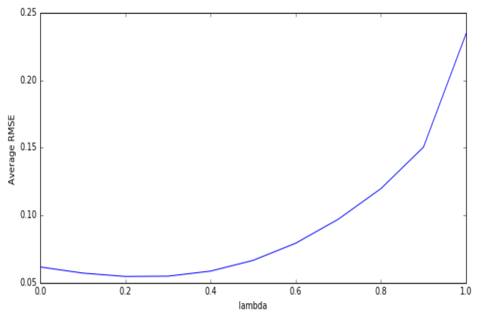


Figure 2(left): Average Error on varying alpha using sequential weights updates
Figure 3(right): Average Error of best alpha on varying lambda using sequential weights updates

In figure3, it is marginally smaller than what is shown in Sutton's figure 5, yet this is due to the fact that Sutton used alpha = .3 while mine with .2 and the training sets are different. Yet, the slope pattern is consistent with Sutton's experiment. It is interesting to note that there is a minimal changes in errors for lambda between 0 and .3. This is because the reward from P_t to P_t+1 are the same unless P_t+1 reaches G's. With a small learning rate alpha, delta_w hardly renders differences for the average RMSE.

Another key point made by Sutton's is that λ = 0 is not an optimal parameter. This is also evident in my experiment, since λ =0 could not stand out during my experiment nor intuitively absorbing sufficient information while performing sequential updates. Meanwhile, learning rate is proven to be critical since time and data could be limited. Recalled from figure1, I find that a proper lambda (.2) with a reasonable alpha is crucial in predictive learning since repeated visiting states may overlook possible adding more noise for larger λ.

Figure 3 also yields comparable results to repeated presentation training results from figure 1 while sequential update is much faster.

5. Summary

After close implementation in TD-λ, I believe this approach is more promising than a conventional supervised learning both in accuracy and speed. Furthermore, its capability to learn sequentially while saving computational space and power is potentially wonderful yet unfolded. Its simplicity in implementation could be easily scaled into multi-threading process and I believe this is vital to tune parameters as this kind of model could be constructed in multi-dimensional search.

References

Sutton, R.S. (1988) Learning to Predict by the Methods of Temporal Differences, pp. 9-44. Kluwer