Git hash: 87c71f0403c3ef09d23d40a5bcec9e072ad21d89

Project 1. Replicating Figures in Sutton (1988)

Bing Yang
Email: yangbing.pku@gmail.com

Abstract—In Sutton (1988), the author introduced the TD(λ) method for incremental learning problems. Unlike classical *supervised learning* methods, which updates the parameters based on differences between observed outcomes and predictions, the *temporal-difference* method improves estimations based on differences in successive predictions, which might be more suitable for problems framed in a sequential manner. In this paper, I replicated the toy example used in the Sutton paper and compare my results to the original conclusions.

Keywords—temporal-difference, random walk, sequential learning

1. Introduction

Predictive learning refers to those problems that use past experience to predict what might happen in the future. One of the commonly used learning algorithms for predictive learning is the *supervised learning* method. The goal of supervised learning method is to find a functional form that can generate predictions from given inputs in an optimal way. Here, the inputs to the algorithm are usually some numerical values that can characterize or represent the observable features of the experience. And by "optimal way", one often means that the accuracy of the predictions can be evaluated on some training sets of experiences and it is possible to find the parameter values that have the best performance over these training sets. One of the performance measures over training data is the squared error, defined as:

$$L = \sum_{i=1}^{n} \left(z_i - P_i(w, x)\right)^2 \quad (1)$$

Here z_i is the observed outcome, and $P_i(w, x)$ is the prediction. The error is evaluated over the entire dataset. One can improve the quality of the learned functional form by minimizing the error over the training sets. One simple way for doing this is by utilizing the gradient of the error over the parameters, namely

$$w = w - \alpha \frac{\partial L}{\partial w} \quad (2)$$

By moving the parameter estimates along the opposite direction, the error can be minimized in successive iterations. The parameter α controls the rate of learning over the entire dataset and is often used to guarantee the convergence of the algorithm.

Substitute L in equation (1) into equation (2), we get

Git hash: 87c71f0403c3ef09d23d40a5bcec9e072ad21d89

$$w = w + \alpha \sum_{i=1}^{n} (z_i - P_i) \frac{\partial P_i}{\partial w} \quad (3)$$

Now consider an incremental prediction problem $(P_1, P_2 \dots P_t \dots P_n, z)$, in which the real outcome can only be observed after a sequence of predictions are made. The complete sequence can be converted into a training set consisting of n prediction-outcome pairs (P_t, z) and the regular supervised learning approach can be applied.

In Sutton (1988), Richard Sutton proposes that instead of using differences between the actual outcome and the predictions as the amplitude of change in parameter estimation iterations, one can use the differences between successive predictions. He suggests using

$$w = w + \alpha \sum_{t=1}^{n} (P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \frac{\partial P_k}{\partial w} \quad (4)$$

as the updating function. The primary rationale and detailed derivation of this formulae from equation (3) will not be discussed in this report. He called this method the *Temporal-Difference* λ method. In fact, TD(1) is exactly the same as equation (3). Based on his analysis, there are two main advantages of the TD(λ) method[1]. First, compared to the classical supervised learning method, this incremental TD(λ) method is computationally more efficient. One does not need to wait until the outcome shows up and store all the predictions along the way. Instead, by the recursive nature shown below,

$$\sum_{k=1}^{t+1} \lambda^{t-k} \frac{\partial P_k}{\partial w} = \lambda \sum_{k=1}^{t} \lambda^{t-k} \frac{\partial P_k}{\partial w} + \frac{\partial P_{t+1}}{\partial w} \quad (5)$$

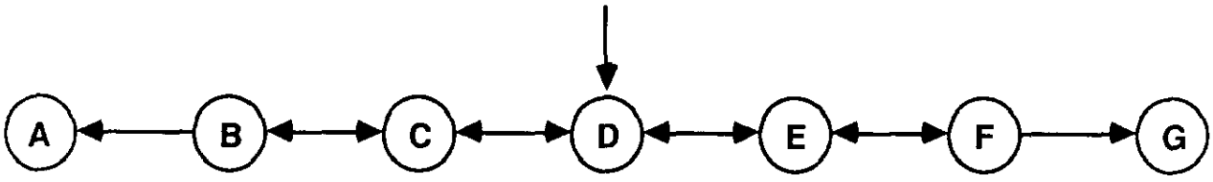
the memory needs for this algorithm is only constant. Second, the TD(λ) method converges faster than the classical supervised learning updates. The details of the proof of convergence can be queried in the original paper.

In this report, I will use the same toy model from Sutton (1988) to illustrate the strength of TD(λ) method in the context of increment learning. I will also demonstrate how the varying the super-parameters λ and α can affect the result and rate of convergence for the TD(λ) algorithm.

2. Results

2.1. The random walk

In this model, an agent does a random walk among several states, as shown below



Git hash: 87c71f0403c3ef09d23d40a5bcec9e072ad21d89

The agent always starts from position D. At each time point, the agent decides whether to go left or right by flipping a fair coin (50-50 landing head) unless he/she is already at A or G, at which time the walk stops. The outcome of the sequence is 0 if landing on A and 1 on G. At each step, the agent predicts what is the expected outcome of the sequence from that time on. Given that there are only two endpoints with outcome 0 and 1 respectively, the prediction at each time point is essentially the expected outcome of a sequence if the walk starts at the position where the agent is standing.

The input feature at each time point is completely determined by the current state. The input feature for each state is in the form of one-hot encoding. In another word, the feature is a binary vector that has the same length as the number of states with unknown expected outcome (excluding A and G, for obvious reason) and has a 1 at the position of the state and 0 for all other positions. For example, the feature vector for state D in this coding will be [0, 0, 1, 0, 0].

In this report, the functional form used for the prediction task is linear relative to the weight parameters. Given that the feature vector in encoded in a one-hot manner, the weights are essentially the expected outcome for each state. From equation (4), the updating formulae is

$$w = w + \alpha \sum_{t=1}^{n} (P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} x_t \quad (6)$$

2.2. Methods

Two simulations were performed in this report. In both simulations, 100 training sets were generated, each consisting of 10 simulated sequence conform to the model specification in section 2.1. The random walk always starts in position D for each simulated sequence. The performance is evaluated using the Root Mean Square error between each estimate and the correct expected outcome $\left[\frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{5}{6}\right]$, defined as

$$Error = \sqrt{\frac{\sum_{i=1}^{5} (est_i - target_i)^2}{5}} \quad (7)$$

The two simulations differ in when to update the parameter estimations and whether to wait for the convergence.

In the first simulation, changes in the estimates are accumulated over the 10 sequences and are added to the parameters altogether. The same training set is presented to the model repeatedly until convergence of the estimates (difference in successive estimation smaller than a predefined error term). A series of λ is used to check its effect on the final performance (λ ranges from 0 to 1, with a step size of 0.1). For each λ, the average error is calculated over the 100 training sets. Note

Git hash: 87c71f0403c3ef09d23d40a5bcec9e072ad21d89

that for each λ, a series of α are tried and the one with the smallest error is used for the final presentation (α ranges from 0 to 0.02, with a step size of 0.002).

In the second simulation, changes in the estimates are accumulated over only one sequence and are added to the parameters right before the next sequence in each training set. This is done only once for each training set (namely, 10 updates per set). A series of both λ and α are used to evaluate how they affect the rate of learning (λ ranges from 0 to 1, with a step size of 0.05; α ranges from 0 to 0.6, with a step size of 0.05). For each combination of λ and α, the average error is calculated over the 100 training sets.

2.3. Figures, results and discussions

2.3.1. TD(λ) method is better than the classical supervised learning method for the toy model

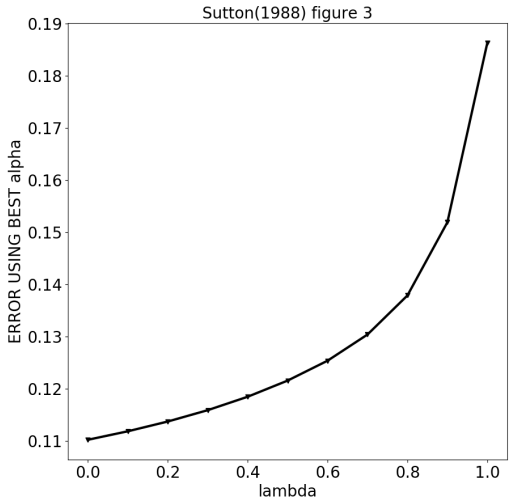


Figure 1. Average error for different λ

Figure 3 shows how the performance varies with different λ. The figure is not exactly the same as the original figure in Sutton (1988) as the smallest error occurs at $\lambda = 0$. However, this matches the statements in the paper. In fact, TD(0) is supposed to perform the best for this model since it is the maximum likelihood estimate given that the underlying process has Markov property. The random walk process is exactly a Markov process, since the state in time step t only depends on the state in time step $t + 1$.

2.3.2. The learning rate is affected by the choice of λ and α

Git hash: `87c71f0403c3ef09d23d40a5bcec9e072ad21d89`

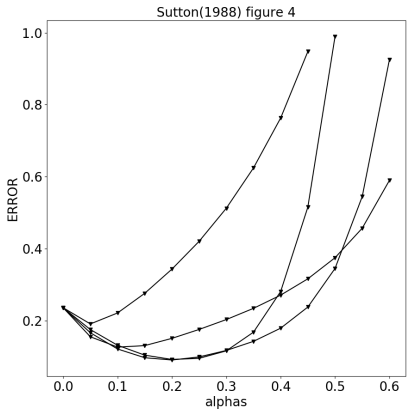


Figure 2. Replicate of figure 4 in Sutton (1988). The lambda value for each line is 1, 0, 0.3, 0.8, sorted by the right end point from left to right

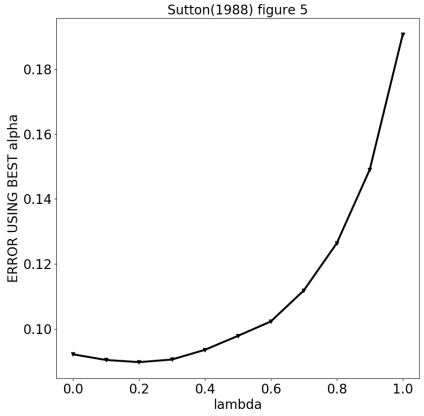


Figure 3. Replicate of figure 5 in Sutton (1988).

The major conclusions in the original paper from figure 4 and 5 are also supported in my replicate figures. As can be seen from figure 2, TD(1) is the worst algorithm for all α considered. Also, for each λ, the best α always has intermediate values. From figure 3, the TD(0) does not has the fastest rate of change when experiencing only 10 sequences, which is similar to what has been shown in the original paper (figure 5).

However, it is clear that the numerical values of each point are a little different from those in the original figures. My hypothesis is that the discrepancy is due to the different realizations of the training samples. In fact, by trying different random seeds when generate training sets, I found out that the numerical values could change across simulations (data not shown). However, the conclusion that TD(1) is the worst approach across all possible α values always hold. This suggests the robustness of the general conclusion in the original paper.

3. Conclusion

In summary, the major conclusions in the Sutton (1988) paper are qualitatively supported by my replication of the original experiments. The discrepancies in actual values in the figures might be due to the different realizations of the training data.