CS7642 Reinforcement Learning Project 1: TD(λ)

Jinxin Wang

February 19, 2021

Abstract

Sutton (1988) introduced the creative methods of temporal difference for the problem of learning to predict. To better understand the underlying theory of TD methods, this report will replicate the results of Sutton's two experiments for the random-walking problem. From the replications, we can find how TD method outperforms Widrow-Hoff rule in terms of efficiency, accuracy and stability, and how learning rate affect the result of TD method.

1 Introduction

Sutton (1988)[1] introduced the methods of temporal differences as a better procedure than supervised learning for *learning to predict* problem. According to his theory, in comparison to Widrow-Hoff rule, the main advantages of TD methods are the speed and incremental computation. Except for the Mathematical proof, the example of random-walking problem also provides significant evidence for his theory.

In this report, I will replicate his experiment result in the well-known random-walking problem. The description of the problem will be introduced at first with some important notations for further implementation. The design of the experiment will follow the structure of Sutton's. Any difference between his result and mine will be analyzed and explained. In the final section, I will share my experience and some difficulties encountered in the replication.

2 Random-walk Problem

2.1 Introduction of random-walk

To replicate the results, this report will also focus on the random-walk example. In this example, a system with a state sequence is generated and all future procedures are based on the system. Among all states in the sequence, the first and the last states (or edge states) are the boundaries and also absorbing states. A walk starts from the center state and takes random steps to the left or to the right until reaching the boundary which is the termination of the walk. In this report, the same 7-state sequence from A to G is used as illustrated by Figure 1.

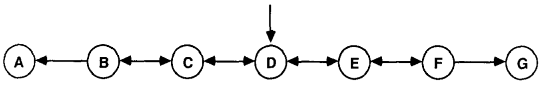

Figure 1: Illustration of the random walk system

In this system, a walk always starts from state D and ends in state A or G. To simplify the problem, the outcome of absorbing states are clarified. If the walk stops at A, the outcome z is 0. If the walk stops at G instead, the outcome z is set to be 1. The purpose of the study is to predict the value of each state from A to G (while the values of A and G are already known).

Before any further investigation on the problem, I would like to introduce some parameters or notations. In this particular example, the value we want to predict for each state is also the probability that from this state a walk will end at G. The predicted values are denoted as P_t for all step/time t in an episode. Since it's a learn to predict procedure, we would like to update our predictions during or after the episodes. To realize the updates, another vector parameter w is introduced. In each step of the episode, the current state is denoted as x_t which is a unit vector. For example, if the state at time t is at point F which is the 6th in the state sequence, we use a vector $x_t = [0, 0, 0, 0, 0, 1, 0]$ to represent the state. The advantage of such denotation will be explained later.

The random-walk problem is linear in terms of space and time, so that we can consider P_t as a function of both x_t and w, that is, $P_t = w^T x_t = \sum_i w(i)x_t(i)$. It has two important features. First, since only the i-th element of x_t is 1, the prediction can be simplified as $P_t = w(i)$. Second, the partial derivative of P with respect to w can also be easily solved as $\nabla_w P_t = x_t$. These two features will strongly help the implementation.

One of the most important characteristic of TD method is the focus on successive changes in each episode instead of the error between each prediction and final outcome. A new factor λ is then introduced to represent the decaying speed of each incremental step. The updates will follow the equation as follows (which is the same equation is Sutton's paper):

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

In this equation, α is the learning rate which determines the step size of each updates. $P_{t+1} - P_t$ represents the successive change. The sum part which contains the derivative represents the direction of the change or update. The two replicated experiments will follow this equation to compute the updates for w.

2.2 Design of experiments

The report will replicate Sutton's two computational experiments based on the random-walk system. Similar to Sutton's work, to obtain statistically reliable results, the training datasets are generated in advance. In both experiments, we have 100 datasets, each consisting of 10 episodes. Each episode was generated randomly without any random seed to represent a walk from D to one of the absorbing states.

The two experiments have some major differences. Firstly, although the updates of weight factor w follow the same equation, the time of such updates is different. In the first experiment, the weights of states are updated only once after all episodes, and each dataset has to be trained repeatedly until the weighs converge. In the second experiment, however, each dataset is only trained once while the weights are updated every time an episode terminates.

Another major difference is the selection of parameters. In experiment one, we arbitrarily select the α value and explore the prediction errors with difference values of λ. The motivation is that we only care the possibly difference results between TD method and Widrow-Hoff rule.

3 Experiment 1

3.1 Implementation

The first experiment concerns where TD methods can provide a better prediction result than Widrow-Hoff rule. Since Widrow-Hoff rule is the same as TD(1) in this case, we can apply the same algorithm to different values of λ. The value of α is set to be very small ($\alpha = 0.01$ in this replication) so that the algorithm can converge. Once converged, the RMS error can then be computed.

Algorithm 1: Experiment 1

```
Input  : training dataset;
Output: $w$;
Initialize $w$, for all nonterminal state $w_i$=0.5;
while not converging do
    for each episode do
        for each step $t$ do
            $e_t \leftarrow \lambda e_{t-1} + \nabla_w P_t$;
            $\Delta w \leftarrow \Delta w + \alpha(P_{t+1} - P_t)e_t$
        end
    end
    $w \leftarrow w + \Delta w$
end
```

3.2 Result

Following the algorithm above, the experiment was replicated with Python. As figure 2 shows, the curve is almost identical to the Figure 3 in Sutton's paper. Widrow-Hoff method has the highest prediction error than any other λ values. In fact, the curve is incremental for increasing λ values from 0 to 1. The standard error in Sutton's first experiment is around 0.01. The number is also successfully replicated. In my experiment, the standard error is 0.009.

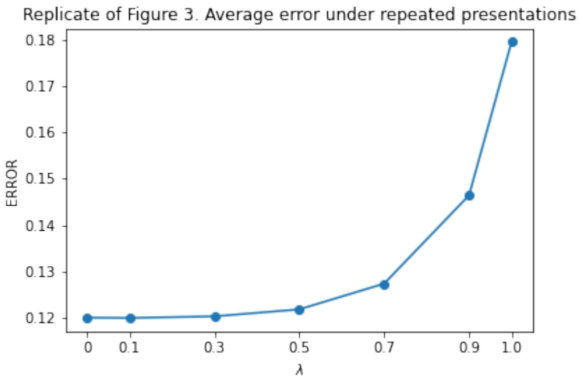

Figure 2: Replication of Figure 3 in Sutton's paper

3.3 Analysis

Well, honestly the curve is not actually "identical" to Sutton's figure 3. The most significant difference is the range of errors. In my replication experiment, the errors are approximately in the range from 0.12 to 0.18 while Sutton's errors are from 0.19 to 0.25. In other words, the prediction errors in the replication is much lower than the original errors. One of the possible reason could come from the training dataset. Although Sutton has demonstrated his mathematical theory and experiments with details, he failed to tell the audience if the training data was generated randomly or with some interference. By altering the random seed could change the range of errors in some degrees and narrow the distance from Sutton's.

The margin in errors could also be caused by the value of α which determines the learning rate of temporal distance method. In the original experiment, the value α was not provided. A higher learning rate in TD method could lead to a larger Δw in each step, which may cause the inaccuracy or even non-convergence.

4 Experiment 2

4.1 Implementation

The major differences between the two experiments have been explained above. Therefore, we only need minor change to the first algorithm so that we can implement it to explore the role of learning rate. In this experiment, values of λ and α are not assigned in advance. Instead, different pairs are tested to clarify different scenarios.

Algorithm 2: Experiment 2

```
Input  : training dataset;
Output: $w$;
for each training dataset do
    Initialize $w$, for all nonterminal state
      $w_i$=0.5;
    for each episode do
        Initialize $\Delta w = \mathbf{0}$;
        for each step $t$ do
            $e_t \leftarrow \lambda e_{t-1} + \nabla_w P_t$;
            $\Delta w \leftarrow \Delta w + \alpha(P_{t+1} - P_t)e_t$
        end
        $w \leftarrow w + \Delta w$
    end
end
```

Once we have the results from all the pairs, it is possible for us to find the best *alpha* which lead to the lowest error for each λ. This implementation can help us understand how learning rate interact with λ in TD method.

4.2 Result

Figure 3 presented us the average errors for difference learning rates and λ. The graph is not exactly the same as Sutton's. One of the most significant differences exists on results with higher learning rates. In my experiment when learning rate α is large, the prediction errors grow much quickly than Sutton's result. For example, for the TD(0) curve, it was surging after the inflection point around $alpha = 0.25$. However, in Sutton's figure 4, TD(0) method only has an average error around 0.5 when $alpha = 0.6$. One possible reason could still be the training dataset as stated previously.

The replicated result becomes better in terms of Sutton's figure 5. Both Sutton's and my figure shows an infection point around $\lambda = 0.2$ or 0.3. On the other hand, TD(1) has a much higher prediction error than any other value. In fact, TD(1) is the worst in most cases while TD(0) is not stable when the learning rate becomes larger. A more comprehensive method such as TD(0.3) could produce a better prediction.

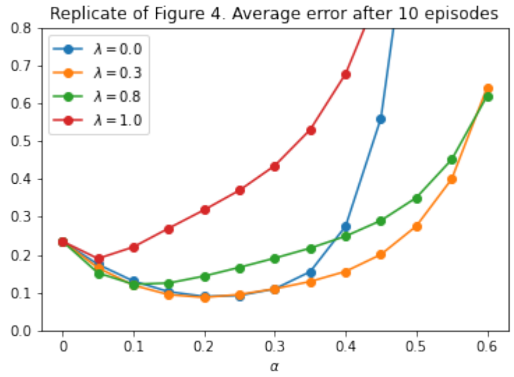

Figure 3: Replication of Figure 4 in Sutton's paper

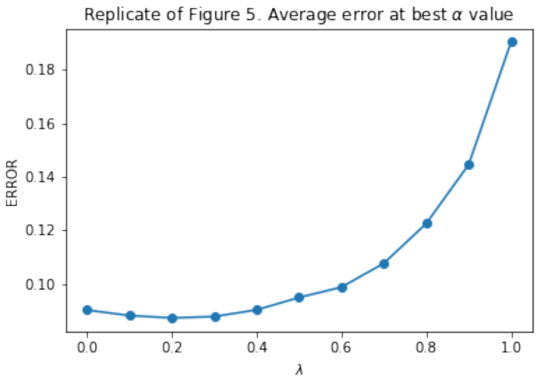

Figure 4: Replication of Figure 5 in Sutton's paper

In practice when the problem becomes much more complicated and we need to take the computational efficiency into consideration, TD(1) method has the disadvantages of low efficiency and low accuracy. When the learning rate is unknown and needs to be tuned, $\lambda = 0.3$ could be a good start point where the stability and accuracy can have a decent balance.

4.3 Analysis

Although the replication of Sutton's figure is not perfect, the conclusion that can be derived from the graph remains the same. In this experiment, we would like to understand how the learning rate and λ value change the prediction error. From this perspective, it is possible for us to find that a suitable and relatively small learning rate with a suitable λ value is the best combination. That is, neither TD(0) and TD(1) is the most effective method for the random-walk problem.

5 Difficulties in Replication

The algorithms in this report are implemented with very straightforward codes. The most difficult part of the replication stays in the understanding of what Sutton said and used. More specifically, the notations and glossary used by Sutton are quite different from the lectures and event his book. I spent most of time working to understand how his equations and theories can relate to the lectures. But once understanding the equations, especially equation (4), the algorithm becomes clear to me and the code implementation only costs quite small amount of time.

The best part of the replication is to help us understand what the learning rate and decaying rate play the role in TD method. It is also inspiring to compare TD method with supervising method on the same learn to predict problem.

References

[1] RICHARD S. SUTTON. Learning to Predict by the Method of Temporal Differences. Machine Learning 3 (Aug. 1988), pp. 9–44.