Implementing Methods of Temporal Differences

Beixi Chen

Spring 2019
bchen372@gatech.edu
git hash: 1fdb17e711f7ac604337aee5c7b0eaf1ddfabe94

ABSTRACT

In paper "Learning to Predict by the Methods of Temporal Differences" [1], author Sutton introduced new prediction-learning methods called temporal-difference (TD) methods which assign credits by means of difference between temporally successive predictions. TD methods require less peak computation and yield more accurate prediction compared with conventional prediction methods which assign credit by difference between the predicted and actual outcomes. To better understand the implementation and performance of TD methods, this article replicated three experiments concerning bounded random walk in Sutton's paper.

I. Environment and Problem description

A. Environment

As stated above, the experiments conducted in Sutton's paper concern random walk, so it is necessary to introduce what is random walk and why use random walk. Random walk, a special category of Markov process [2], describes a path that consists of a succession of independent random steps on some mathematical space such as the integers. The definition reveals two important properties of random walk, first, the random walk is a succession of steps meaning it is a process; second, steps in random walk are independent and the transition probability from one step to the next step only depends on the current state and this property is also known as Markov property.

Why use random walk in TD methods implementation? In reality, many prediction problems are multi-step problems, for instance, a robot hears a sentence and decides which words it heard, it updates its prediction according to the sequence of voice, which is a multi-step process. TD methods perform well on multi-step prediction problems, therefore, the experimental data is also supposed to be data sequences, and such data sequences could be generated by dynamical system, a system which have a state that can be observed evolving over time. Random walk is exactly such multi-step process which evolves over time, and is also the simplest dynamical system, therefore easy to analyze.

A bounded random walk is needed, since unbounded random walk would explode in the course of time, then the expectation of unbounded random walk would approach infinite. Also, as stated in Theorem 2 in Sutton's paper [1], one of the sufficient condition for TD methods converge is an absorbing Markov chain.

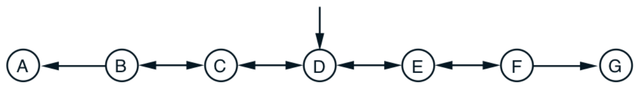


Fig. 1: Bounded Random Walk

B. Problem description

As shown in Figure 1, all walks in this paper begin in state D. States B to F are non-absorbing states, in these states the walk has equal chance of moving either to right or left, the walk terminates when absorbing State A and G are entered. There are two outcomes for one single walk, if the walk terminates at State G, outcome z is 1, if terminates at State A, outcome z is 0. For each non-terminal state z there was a corresponding observation vector x_i. The prediction denotes as $P_t = W^T \times X_t$, where W^T is the weight of non-absorbing States at step t. The ideal predictions for each non-absorbing states are $[1/6, 2/6, 3/6, 4/6, 5/6]$, the goal of experiments is to minimize the root mean square (RMS) error between the ideal predictions and those obtained by learning procedure.

II. Implementation

A. Algorithms

Sutton showed the general update rule for weight w is

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

where $\Delta w_t = \alpha(z - P_t)\nabla_w P_t$ and when $P_t = w^T x_t$ then $\Delta w_t = \alpha(z - P_t)\nabla_w w^T x_t$, this known as Widrow-Hoff rule, but it cannot be determined until the end of the sequence when z becomes known, TD methods instead can be computed incrementally:

$$z - P_t = \sum_{k=t}^{m}(P_{k+1} - P_k) \tag{2}$$

where $P_{m+1} = z$, then (1) and (2) can be combined as:

$$w = w + \sum_{t=1}^{m} \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \nabla_w P_k \tag{3}$$

In bounded random walk case, m represents mth data sequence in one training set, t represents step t in a single data sequence, since $P_t = w^T x_t$ thus $\nabla_w P_k = x_t$, where x_t is the observation vector.

B. Implementation for Experiment 1

First, created 100 training sets, and in each training set created 10 sequences, for each single sequence generated a bounded random walk as stated above. Note that the format for each state within a bounded random walk is a vector of length 5 because the absorbing states are truncated, for instance, the start state D is presented as $[0, 0, 1, 0, 0]$ and state B is $[1, 0, 0, 0, 0]$. Second, generate initial weight for all non-absorbing states, since Sutton states in second experiment that all components of the weight vector were initially set to 0.5, so here author of this article also presumes that in the first experiment there was no bias toward both ends thus also set all elements in w to 0.5. α, λ, ϵ selection would be discussed later, as long as the value of parameters have been set up, the next things to do was determine the update rule of weight vector w. As shown in Sutton's paper, Δw were accumulated over all sequences in one training set, once all the sequences in one training set have been presented and the converge limit (epsilon) has been achieved, Δw then would used to update the weight vector w. For each training set, a updated weight vector w and its corresponding RMS would be calculated, and after presenting all 100 training sets, 100 RMS would be averaged. Thus for each α, λ parameter pair, there would be one corresponding averaged RMS.

C. Implementation for Experiment 2

The second experiment has similar implementation procedure as the first one, yet this time a wider range of α would be used. Two differences need to be pointed out are: first, each training set could be used for several times in the first experiment concerning the converge limit, but now each training set was presented only once to each learning procedure; second, previously, weight vector w was updated after one training set was completed, yet now updates of w were performed after each sequence within one training set, as Sutton highlighted in his paper [1].

D. Implementation for Experiment 3

The third experiment was an upgrade version of the second experiment, it chooses the best α (α with minimum averaged RMS) of each λ value and plots the averaged RMS against corresponding λ value using best α.

III. Outcome Analysis

A. Experiment 1

1) Output and Analysis: The figure of replicating Experiment 1 and the original figure of Experiment 1 are shown in Figure 2. Both two curve were increasing, having their lowest error when using TD(0) method and having their highest error when using the TD(1) method, in other words, the conventional method. We can observe that the shape of curve

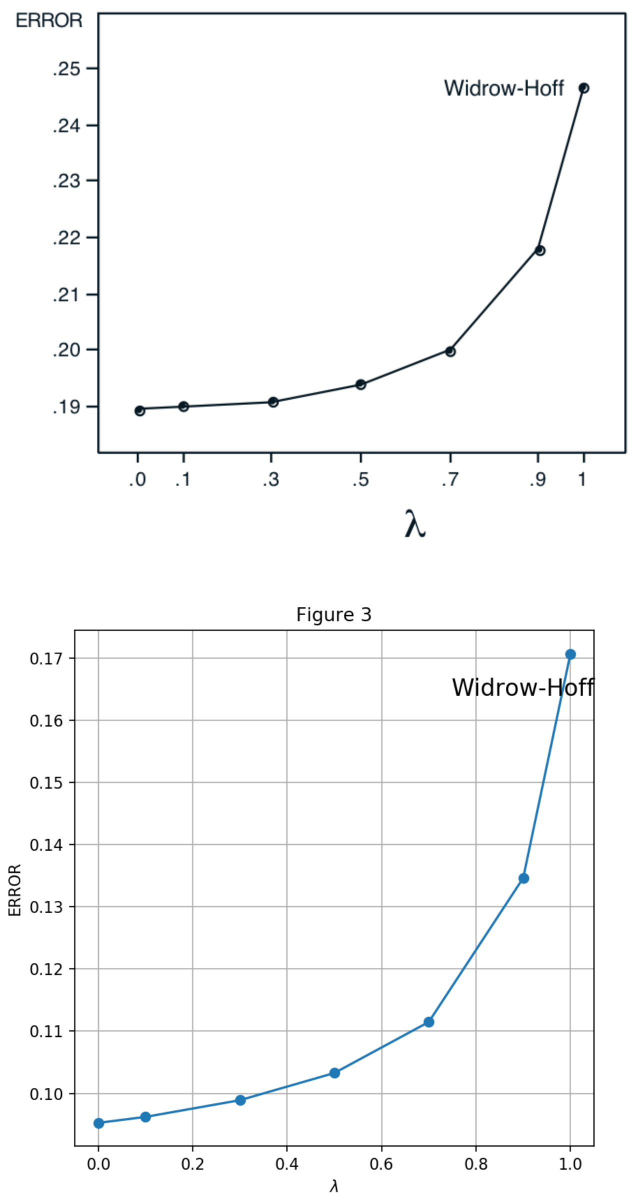


Fig. 2: Experiment 1

in both figures are similar, but if we plot them in one picture, we could find that the replicating curve was generally below the original curve, the gap between them was about 0.08. The author of this article comes up with some reasons that cause such discrepancy:

- The original experiment may use relative high *epsilon* value, making the converge time much shorter compared with its replicating counterparty.
- Given the original figure was not created until 1988 publication of Sutton's paper, the computational power of that time was not more powerful than that of nowadays. Thus, lower computational accuracy of computation that time may lead to higher error of experiment.

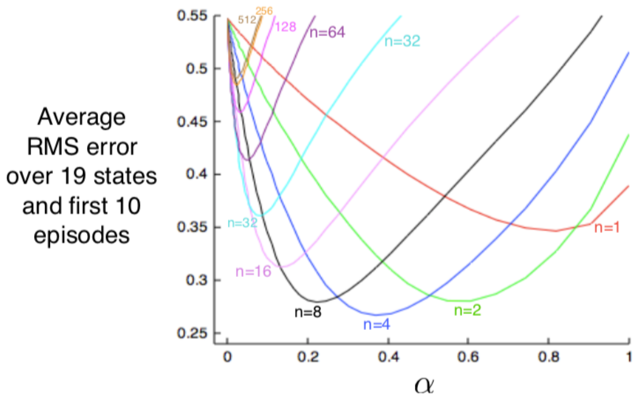
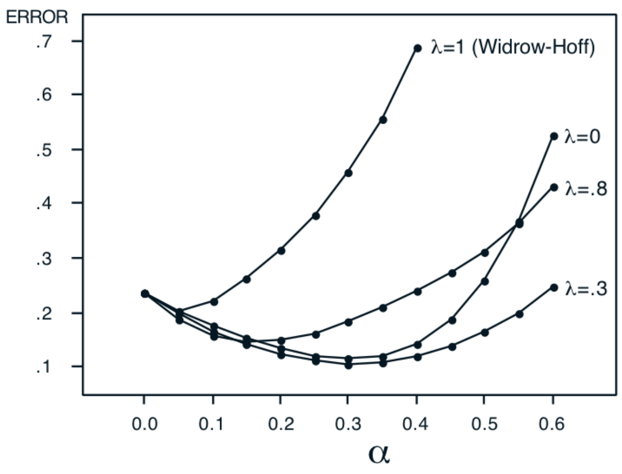


Fig. 3: Performance of n-step TD methods as a function of α Cited from "Reinforcement Learning: An Introduction"



2) Pitfalls and Solution: Note that Sutton did not mention explicitly the value of α in the first experiment, the author of this article at the very beginning used value of α about 0.2 0.3, sometimes certain training set may diverge. Thus, one of the pitfalls in replicating Sutton's paper was to find a proper learning rate α. But the solution could be found in the second experiment of Sutton's paper. In the second experiment, Sutton used small α, inferring learning rate α was highly likely has a small magnitude. Also, in Sutton's book, "Reinforcement Learning: An Introduction", [3], small α generally produced lower error as described in Fig. 3 which cited from that book. Then author of this article tried $\alpha = 0.01$ and the resulting curve yielded a similar shape.

B. Experiment 2

1) Output and Analysis: Fig. 4 and Fig. 5 are the replicating results of second experiment with two different seeds. They both had their lowest point when α was approaching 0.3, and the TD(1) method dominates other TD(λ) methods. One can observe that the replication with seed value 11 was more consistent with the original figure compared with that generated from seed 2, such consistency reflects from two aspects:

- The scale of error, the figure generated from seed 11 had error ranging from $0.2 - 1.4$ which is more closer to the original figure.
- In the figure generated from seed 11, line "$\lambda = 1.0$" was always above line "$\lambda = 0.0$", while seed = 2 figure not.

2) Pitfalls and Solution: There are several pitfalls that should be mentioned in the second experiment. When the author of this article tried to implement it the same way as the first experiment, the computer always reported an error of overflows. First author tried to turned some variable to float64 but it did not work, when author read Sutton's paper carefully, author noted that the update rule for weight vector w was no longer the same as described in the first experiment. As written in the previous part of this article, the weight vector w updated after each one data sequence and there was no search

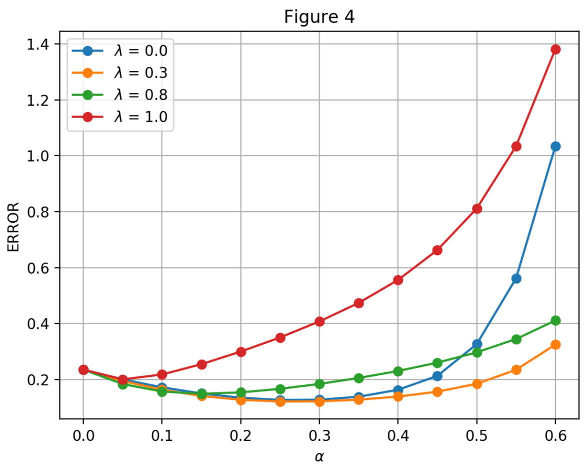


Fig. 4: Experiment 2 with seed = 11

for weight vector convergence. After changing the update rule, the resulting figure looked fine. Another pitfall is the selection of seed value, of course Sutton would not present how he generated his training data, thus it was almost impossible to perfectly replicate the original figure. As the seed value varied, the author found that the figure also varied a lot. In conclusion, the weight update rule and the choice of seed would affect the final result to some extent.

C. Experiment 3

1) Output and Analysis: As shown in Fig. 6, the replicating figure almost perfectly replicated the original figure. They both had their lowest point when λ was around 0.3. From both figure, one can observe that when using best α, the TD($\lambda < 1$) methods outperform the conventional TD(1) method. In addition, compared with the first experiment, the optimal λ in Sutton's bounded random walk example was no longer the $\lambda = 0$ (As shown in first experiment, the minimum error appeared when $\lambda = 0$), but instead, the optimal λ is around 0.3 when using best α.

2) Pitfalls and Solution: Noted that the third experiment followed the same weight vector update rule as the second

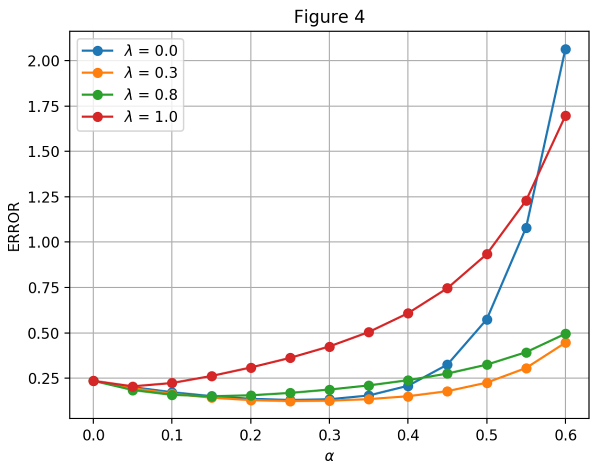


Fig. 5: Experiment 2 with seed = 2

experiment. And the initial value of all components in weight vector were also all equal to 0.5, meaning no bias to both terminations.

IV. Conclusion

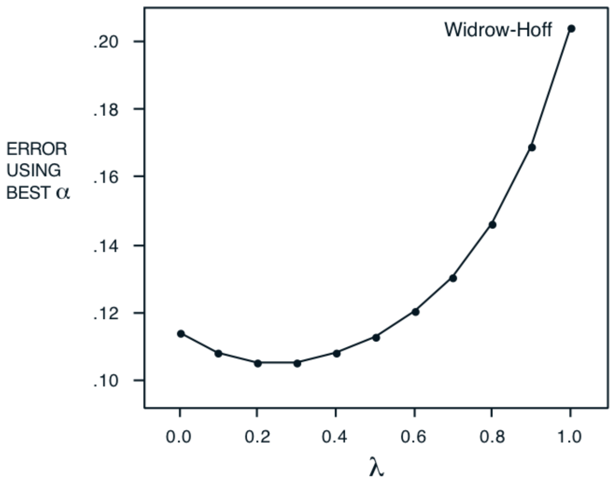
The output of this article shows a similar results of Sutton's finding, it verifies that TD(λ) methods generally outperform Widrow-Hoff (TD(1)) in prediction problems of multi-step scenarios. In the course of experiments replications, this article reveals that the update rule of weight vector w, the value selection of learning rate (α), even the value selection of seed which used to generate training data would affect final performance significantly.

References

[1] R. S. Sutton, "Learning to predict by the methods of temporal differences," p. 36, 1988.

[2] Random walk. [Online]. Available: https://en.wikipedia.org/wiki/Random_walk

[3] A. G. B. Richard S. Sutton, *Reinforcement Learning: An Introduction*. The MIT Press, 2017.



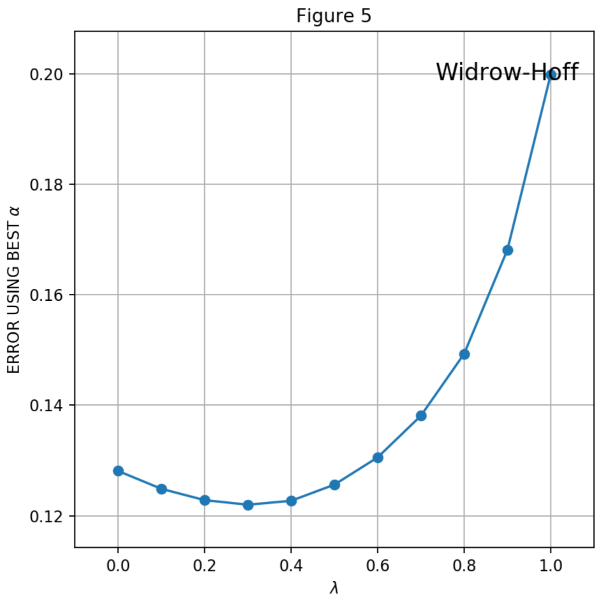


Fig. 6: Experiment 2