

Project 1: Desperately Seeking Sutton

Ran Chen
email: ranchen@gatech.edu

Abstract

This report replicates the experiments generating Fig 3, 4, and 5 from *Learning to Predict by the Methods of Temporal Differences*¹ by Sutton, RS (referred to as the "Sutton Paper" in this report). Code used to run experiments described in this report is hosted here: <https://github.gatech.edu/rchen350/cs7642summer2018p1>, which is also linked in the header as required

1 Problem: Random Walk

Here is an simple example described in the Sutton Paper: as shown in **Fig 1**, we have game of bounded random walks with 7 states, in which the starting point is state D, and the probability of moving either right or left is 0.5, and the game ends when either of the edge states A and G is reached. The value of the game, or the outcome of the random walks, is defined as 0 if the end state is A, and 1 if the end state is G.

To formalize this random walks game as an Markov Decision Process (MDP), we define a state X_S in the form of a one-hot encoded vector of length 7, with the component corresponding to the state being 1, and the rest being 0, e.g., $X_A = [1, 0, 0, 0, 0, 0, 0]$, $X_D = [0, 0, 0, 1, 0, 0, 0]$, and $X_G = [0, 0, 0, 0, 0, 0, 1]$. The value prediction of a particular state S , is defined as $P(S) = \omega^T X_S$, which is simply the corresponding component of vector ω since only the corresponding component in X_S is 1.

In this particular game, since the game out come is defined as 1 for terminating at X_G and 0 for terminating at A, we can define the value of a state X_S to be the probability of terminating at state G, or equivalently the expected value of the game outcome, if the game is started from state X_S , this $P(X_A) = 0$ and $P(X_G) = 1$, which means the vector ω should take the form of $[0, \omega_B, \omega_C, \omega_D, \omega_E, \omega_F, 1]$ with all values of ω components being the probabilities of terminating at X_G . We can compute the ideal values of these probabilities: $\omega^* = [0, \frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{5}{6}, 1]$, and we defined the error between ω generated from simulated data using TD learners and the ideal values ω^* to be the Euclidean distance between the two vectors divided by $\sqrt{5}$, so that we only take into account the non-terminal states (B to F).

Following derivation from the Sutton Paper, to iteratively improve our estimation of ω , at step t , the update is

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

let

$$e_t = \sum_{k=1}^t \lambda^{t-k} \nabla_{\omega} P_k$$

then

$$\begin{aligned} e_{t+1} &= \sum_{k=1}^{t+1} \lambda^{t+1-k} \nabla_{\omega} P_k \\ &= \nabla_{\omega} P_{t+1} + \sum_{k=1}^t \lambda^{t+1-k} \nabla_{\omega} P_k \\ &= \nabla_{\omega} P_{t+1} + \lambda e_t \end{aligned}$$

Note that when $\lambda = 0$: $\Delta\omega_t = \alpha(P_{t+1} - P_t) \nabla_{\omega} P_t$, and when $\lambda = 1$: $\Delta\omega_t = \alpha(P_{t+1} - P_t) \sum_{k=1}^t \nabla_{\omega} P_k$. Also since $P_k = \omega^T X_k$, $\nabla_{\omega} P_k = X_k$. Thus the update from step t to $t+1$ becomes $\Delta\omega_t = \alpha(P_{t+1} - P_t) e_t$ with $e_{t+1} = X_{t+1} + \lambda e_t$.

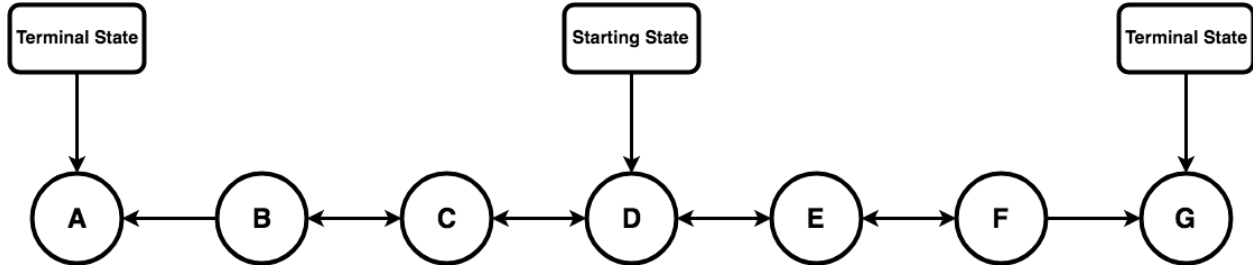


Figure 1: A generator of bounded random walks. This Markov process generated the data sequences in the example. All walks begin in state D. From states B, C, D, E, and F, the walk has a 50% chance of moving either to the right or to the left. If either edge state, A or G, is entered, then the walk terminates.

2 Experiments

Here we replicate 2 experiments performed in the Sutton Paper. These experiments utilizes simulated random walk data. First we generate 1000 random walk sequences, all of which start from X_D , and end in either X_A or X_G . These 1000 sequences are then divided into 100 training set, each containing 10 sequences.

2.1 Experiment 1

The first experiments iterate through all 10 sequences from a training set repeatedly until convergence. $\delta\omega_t$ accumulated over the steps within the sequence, and through all 10 sequences within a training set, ω is updated after one full iteration of the entire training set. Multiple α values are tested, mostly the algorithm converges for $\alpha < 0.1$, and always converge to the same values when it does. Here the learning rate α is set to be 0.01 in all cases.

The general algorithm is shown below:

Algorithm 1 $TD(\lambda)$ on training set TS until convergence

initialize: $\omega = [0, 0.5, 0.5, 0.5, 0.5, 0.5, 1]$

repeat

$\Delta\omega = [0, 0, 0, 0, 0, 0, 0]$

for all sequences in TS **do**

X_0 is the first step

$e = [0, 0, 0, 0, 0, 0, 0]$

$P_0 = \omega^T X_0$

for all steps in a sequence **do**

current step is X_k

$P_0 = \omega^T X_k$

$e = e + X_{k-1}$

$\Delta\omega = \Delta\omega + \alpha(P_1 - P_0)e$

$P_0 = P_1$

$e = \lambda e$

end for

end for

$\omega = \omega + \Delta\omega$

until $\Delta\omega < \epsilon$

This algorithm is then run on all 100 training sets, and for each training set, a root mean square error (RMSE) is calculated, then averaged over 100 training sets to obtain the average RMSE. This procedure is performed with a range of λ values, and compared (**Fig 2**).

2.2 Experiment 2

The second experiments iterate through all 10 sequences from a training set just once. $\delta\omega_t$ accumulated over the steps within the sequence, and ω is updated after one full iteration of each sequence.

The general algorithm is shown below:

Algorithm 2 $TD(\lambda)$ on training set TS once

```

initialize:  $\omega = [0, 0.5, 0.5, 0.5, 0.5, 0.5, 1]$ 
 $\Delta\omega = [0, 0, 0, 0, 0, 0, 0]$ 
for all sequences in  $TS$  do
     $X_0$  is the first step
     $e = [0, 0, 0, 0, 0, 0, 0]$ 
     $P_0 = \omega^T X_0$ 
    for all steps in a sequence do
        current step is  $X_k$ 
         $P_0 = \omega^T X_k$ 
         $e = e + X_{k-1}$ 
         $\Delta\omega = \Delta\omega + \alpha(P_1 - P_0)e$ 
         $P_0 = P_1$ 
         $e = \lambda e$ 
    end for
     $\omega = \omega + \Delta\omega$ 
end for
```

Again this algorithm is then run on all 100 training sets, and for each training set, a root mean square error (RMSE) is calculated, then averaged over 100 training sets to obtain the average RMSE. This procedure is performed with a range of λ and α value combinations, and compared (**Fig 3**). The lowest average RMSE achieved at different λ is then plotted and compared (**Fig 4**).

3 Results

The results from the two experiments described above are presented in **Fig 2**, **3**, and **4**

3.1 Experiment 1

The shape of the curve in **Fig 2** generated from experiment 1 matches almost identically to Fig 3 in the Sutton Paper, except errors at all λ values are consistently lower compare to Sutton's results. This is probably due to the difference in the quality of randomly generated data.

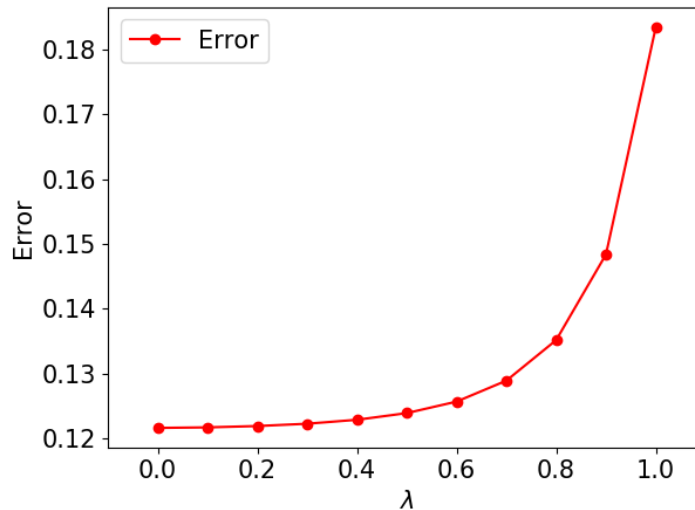


Figure 2: Average error on the random walk problem under repeated presentations.

3.2 Experiment 2

Again the shape of the curve generated from experiment 2 shown in **Fig 3** matches those in Fig 4 from the Sutton Paper: for each λ , the lowest error is achieved at some α value between 0 and 0.6. $TD(\lambda = 1)$ performed worst, while $TD(\lambda = 0.3)$ performed best. However the rate of increasing error with increasing α is higher at some λ values compared to Fig 4 from the Sutton Paper. Again this is likely due to the quality of some training sequences. Since this result is generated from only one iteration of a training set, and with some high α values, the TD algorithm could diverge, thus giving higher error after being shown the sequences just once.

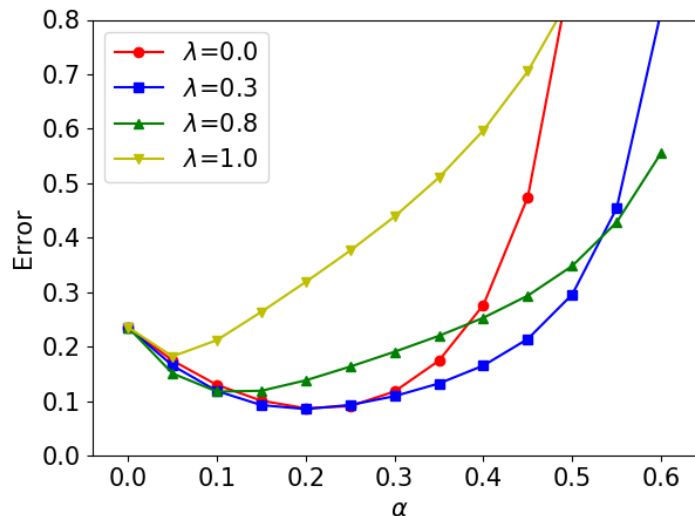


Figure 3: Average error on random walk problem after experiencing 10 sequences.

Experiment 2 is then run again with more λ and α values, at each λ the lowest error achieved by the best α is plotted against λ to show the best result achievable for each of them (**Fig 4**). This result is also very similar to Fig 5 from the Sutton Paper, with the exception of $\lambda = 0.2$ having the lowest error instead of

$\lambda = 0.3$ from Sutton's result. Again this is due to the differences in training sets, also the error from $\lambda = 0.2$ and $\lambda = 0.3$ in Sutton's Fig 5 is actually very close.

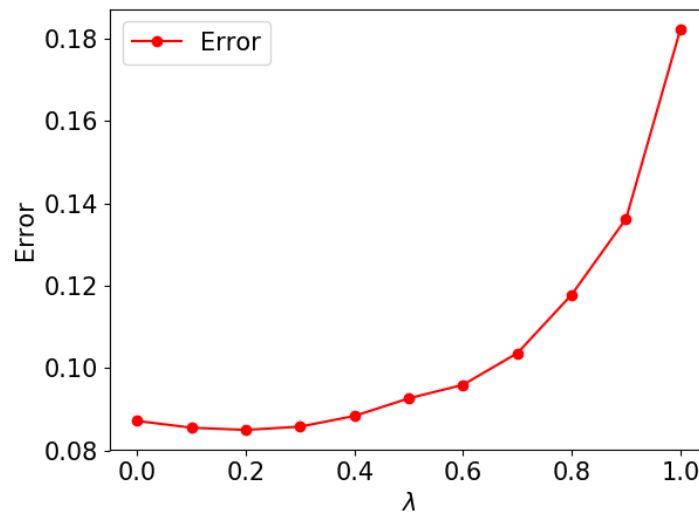


Figure 4: Average error at best α value on random walk problem.

4 Difficulties

- Experiment 1
 - The TD algorithm is not described very clearly in the paper, while the algorithm described in the lecture needs some modification to conform with the paper.
 - Choosing the right α : Since the data is presented repeatedly until convergence, α can be small yet still converge. However if α is too larger, for the algorithm could diverge. A few tests was needed to figure out the proper α .
 - Choosing the right ϵ as convergence criterion, and determining how ϵ is calculated. After a few trial, it was eventually decided to calculated as the average of the latest 20 updates toward vector ω , then averaged over the all intermediate states.
- Experiment 2
 - Figuring out the reason for the slight difference in the plots, as some of the error shoots up faster as λ goes up. It turns out it is just due to the quality of the simulated game data.
 - Cooling Rate: [0.15, 0.35, 0.55, 0.75, 0.95]

References

- ¹ Richard S. Sutton. Learning to Predict by the Methods of Temporal Differences. *Machine Learning*, 3(1):9–44, August 1988.