Project 1: A Sutton 88's replica

Latest hash commit: d0100f3e0d125ff2fcbff210258930c08d320f06

Author: Poujon Adrien

I. Introduction

In the following analysis, I rely on the study of an example simple enough for conclusions to be drawn upon the efficiency of different TD(λ) methods, and in comparison with more classical approaches - at the time Sutton's paper was written- such as Widrow-Hoff procedure. I start by describing the problem and the significance of the results to be acquired (Section II). I then present the generated figures and discuss their implementation (Section III). The paper ends with an analysis of the results in comparison with Sutton's 88 paper (Section IV).

II. Problem description & significance

A. Introducing random walk

The random walk is a simple Markov process. It takes place without any take of action from the agent because travels happen at random and is characterized by a reward depending on the outcome of the walk. Basically, the agent starts from a given state and goes around at random. Each state -except for terminal states- gives access to two adjacent states, and the process could be seen the discrete moving of the agent along a straight line. In Sutton's paper, the nodes are labelled by alphabetic letters, and so is done in this paper. From left to right, I thus call the states A, B, C, D, E, F, G, the initial state being D. The two ends of the line (A and G) are thus the extremes states for the random walk, they make the walk terminate. By convention, the terminal state on the left of the line if a losing one. Therefore, all rewards are set to zero, except for the terminal state on the right which rewards +1 to the agent.

The aim of this work is to provide a way to evaluating the expected reward when the agent is in a particular state. As there is no negative reward, this is equivalent to computing the probability of the agent ending its walk in G when being in each state. As this is a random walk, I can already say that the value associated to D will be 1/2, because there is the same probability to end at A or G starting from the middle point D.

For knowing the probability value of all the states, I can define $A = (a_{ij})$, a matrix representing the probabilities of transitions from state n°i to state n°j (where A,B,...,G are indexed 0,1,...,6). Iterating A over A ad infinitum, I can obtain the probability of ending in state G starting from any state by reading the last column of A^∞, and I thus have the theoretical solution of the presented problem: $0, 1/6, 2/6, 3/6, 4/6, 5/6, 1$.

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{and } A^\infty = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5/6 & 0 & 0 & 0 & 0 & 0 & 1/6 \\ 2/3 & 0 & 0 & 0 & 0 & 0 & 1/3 \\ 1/2 & 0 & 0 & 0 & 0 & 0 & 1/2 \\ 1/3 & 0 & 0 & 0 & 0 & 0 & 2/3 \\ 1/6 & 0 & 0 & 0 & 0 & 0 & 5/6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

B. Introducing TD learning

The TD lambda method is a method for learning from the online evolution of the states of a system. The principle is predicting an outcome from a given state and comparing this prediction with other prediction(s) from previous state(s). This comparison leads to weight updates leading to the retrospective update of the predictions for next iterations. The important part of the algorithm is thus the weight update, describe by equation (1) and (2).

$$w \leftarrow w + \sum \Delta w \tag{1}$$

$$\Delta w_t = \alpha(P_{t+1} - P_t) * (\nabla_w P_t + \lambda e_{t-1}) \tag{2}$$

In equation (2), from which TD(λ) owns its name, the term e_t correspond to the considering of the series of previous predictions. Depending on the value of λ, the importance of previous predictions in the update of weights changes. Basically, when $\lambda = 0$, the update only considers the current and next states, whereas for $\lambda = 1$, all previous predictions are also considered. In this last case, equation (2) becomes equation (3) below,

$$\Delta w_t = \alpha(P_{t+1} - P_t) * \sum(\nabla_w P) \tag{3}$$

and is equivalent to the Widrow-Hoff procedure, or Least Mean Square, which is known for converging to the actual solution [3]. This work will show that this Wirdow-Hoff unbiased procedure is not the best of all $TD(\lambda)$ procedures. Similarly, I will show that the maximum-likelihood procedure embodied by $TD(0)$ is not always the best either.

When studying the equation, conjectures can be made. The value of λ will affect the method in modifying the

kinetic of the weights update. For $\lambda = 0$, the weights are slowly updated after each random walk: at least at the beginning of the procedure, only the last non-updated weight at both ends will be updated. On the contrary, for higher λ values the weights could propagate from the two extremities of the weight vector with a lower number of random walk sequences.

III. Graphs & implementation

A. Reproducing Sutton's 88 figures 3, 4, 5

I propose to reproduce 3 figures from Sutton's 88 paper. The first of these figures (Fig.1.b) is a graph representing the average error obtained for different λ for a given α-step. It has been obtained by following the procedure detailed in pseudo-code 2: weights are computed through 100 trainings consisting of repetitions 10 sequences after which the weight are updated, until their convergence. The error is then computed between the obtained predictions and the references as calculated in section II. I see that the worst $TD(\lambda)$ is TD(1) and the best is TD(0). Sutton shows that TD(0) converges to the optimal estimate for matching future experience.

The second figure (Fig.2.b), represents the average error of different $TD(\lambda)$ techniques depending on the α-step parameter. Here, the procedure also consist of 100 trainings, but this time each training set consists of only 10 sequences and the weights are updated after each sequence, not waiting for the 10 sequences to end. Again, TD(1) is shown to be the worst learning procedure for every α. TD(0) gives one of the best performances for α-step among the interval $[0.2; 0.3]$. The graph also illustrates that intermediary values for λ also give good RMS errors.

The last figure (Fig.3.b) compiles the information from the previous graph and displays the lowest average error for each lambda. It clearly shows that the best λ values are among the interval [0.1;0.4]. Note that the best λ is here λ=0.3 as in Sutton's paper, however, as will be discussed, the stochasticity of the results does not allow to conclude that one should always choose $\lambda = 0.3$ as a parameter for $TD(\lambda)$ method.

B. Discussing implementation

In discussing implementations, I should start by saying a few words about the two last figures (Fig.2.a & Fig.3.a). Reproducing these two figures was much easier than the first one. The details provided by Sutton concerning the process for their computation are complete and sufficient for implementation. The principle is described in pseudo-code 1 for the computation of the weights.

The first Fig.1.a was slightly harder to reproduce. I found the details of implementation vague and blurry. First, the update of w is not done after each sequence, but at the end of a given training set. This is not in adequacy with the method and this is given without prior explanations (one can find that it is for being able to prove convergence for α sufficiently small in the following parts of the paper). Moreover, Sutton simply says that:

> "Δw's were accumulated over sequences and only used to update the weight vector after the complete presentation of a training set." [1]

The pseudo-code thus becomes pseudo-code 2.

Algorithm 1 Experiment for figures 4 & 5

```
1:  for 100 trainings do
2:      create a set of 10 sequences
3:      fix λ and α    ▷ You may loop over predefine sets
4:      initialize w, e_init = [00000]
5:      for each sequence do
6:          initialize Δw = [00000]
7:          for the whole considered sequence do
8:              e_t ← x_t + λ e_{t−1}
9:              Δw ← α ∗ (P_{t+1} − P_t) ∗ e_t
10:         end for
11:         w ← w + Δw                          ▷ update w
12:     end for
13:     use w for updating errors
14: end for
```

Algorithm 2 Experiment for figure 1

```
1:  for 100 trainings do
2:      while w has not converged do
3:          create a set of 10 sequences
4:          fix λ and α▷ You may loop over predefine sets
5:          initialize w, e_init = [00000]
6:          for each sequence do
7:              initialize Δw = [00000]
8:              for the whole considered sequence do
9:                  e_t ← x_t + λ e_{t−1}
10:                 Δw ← α ∗ (P_{t+1} − P_t) ∗ e_t
11:             end for
12:             accumulate Δw    ▷ Not clearly defined in
    Sutton 8
13:         end for
14:         w ← w + Δw                          ▷ update w
15:     end while
16:     use w for updating errors
17: end for
```

It must be underlined that the line 12 of pseudo-code 2 is not very explicit. In fact, Sutton avoids explaining in full extent the details how the Δw's are accumulated. Several solutions are here possible. I considered two: a simple accumulation by summing the Δw's obtained over the 10 sequences; and what I call a normalized accumulation for which the Δw's are summed over a sequence and the accumulation is the mean of the final Δw's over the 10 sets. This lack of information does not result in actual difference between the obtained results as shown in Fig.4.

Finally, Figure 3 from Sutton's 88 was also difficult to plot, because of the difficulty of adjusting λ and α

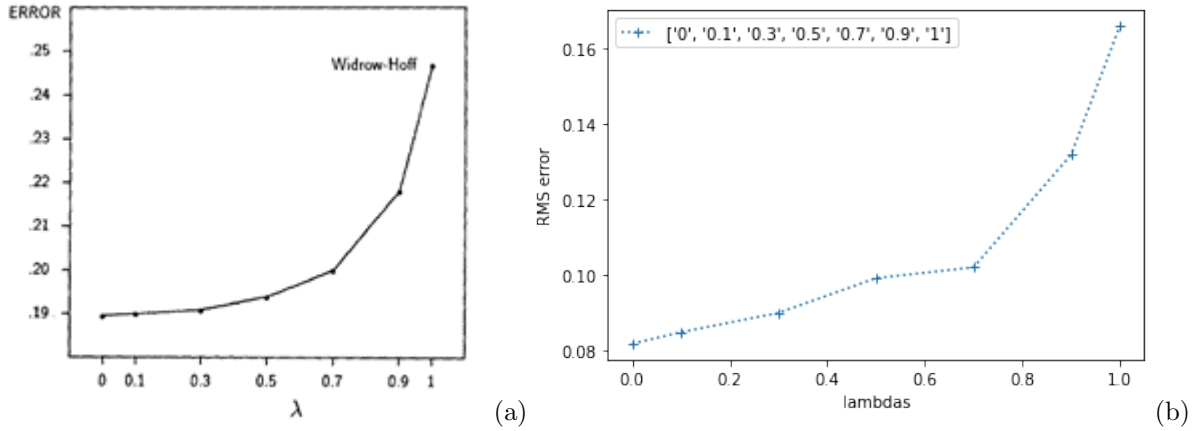

Fig. 1. Sutton's 88 Figure 3 (a) reproduction (b)

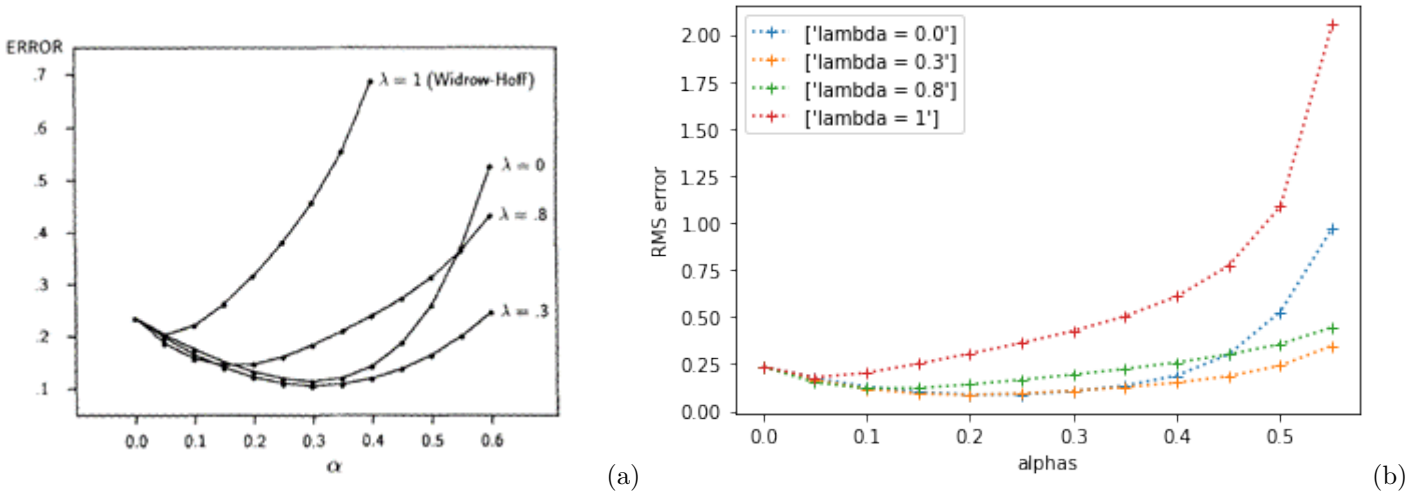

Fig. 2. Sutton's 88 Figure 4 (a) reproduction (b)

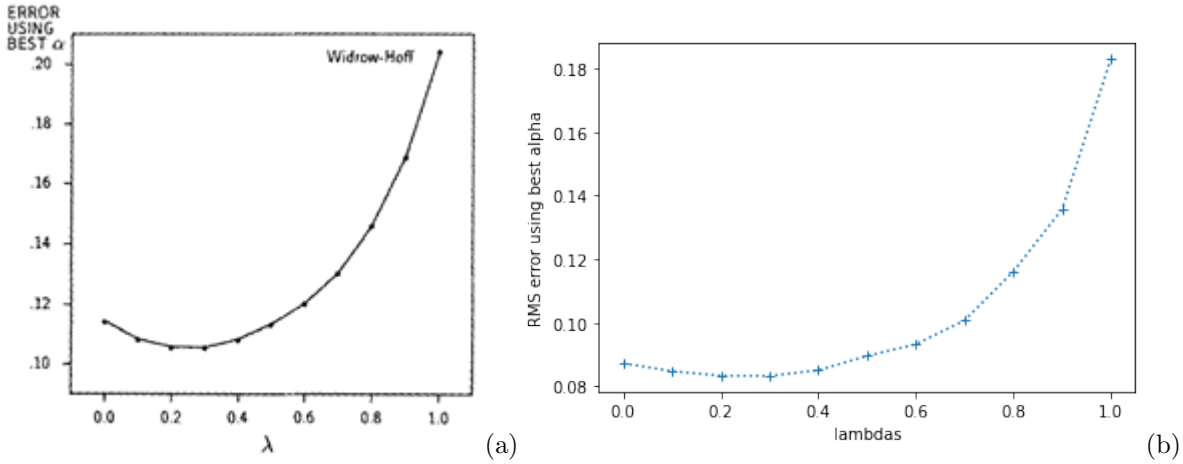

Fig. 3. Sutton's 88 Figure 5 (a) reproduction (b)

parameters. I have tried many different configurations before obtaining this final result. The adjustment of the parameters is made harder in that case by the possibility of diverging configurations of the pair (α,λ). If α is taken too big relatively to λ then the procedure could diverge and if one wants to prevent divergence by selecting very small values, the procedure might end up in not doing anything at all for most λ.

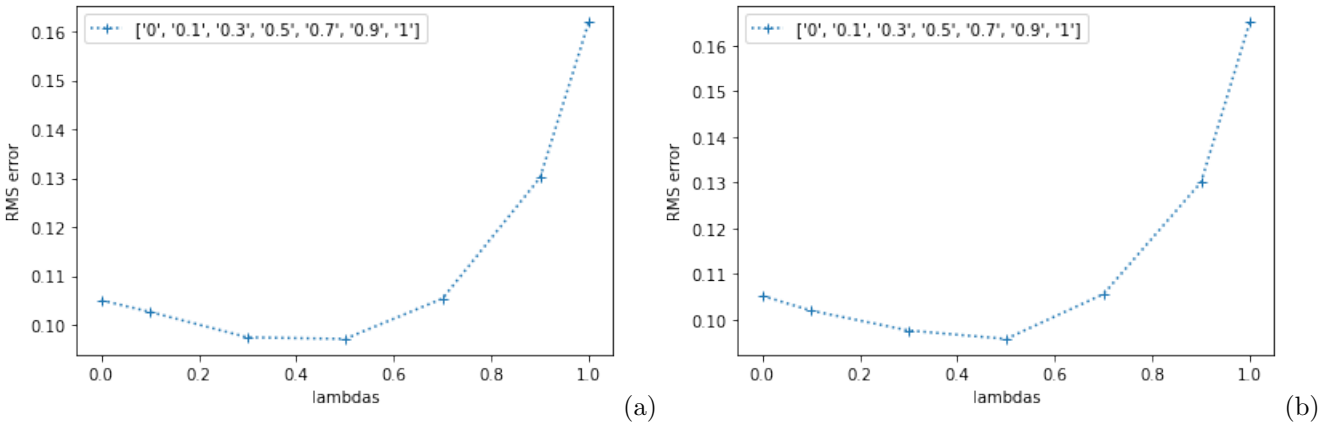

Fig. 4. Average of 20 experiments 1 for normalized (a) and simple (b) accumulation

IV. Analysis of results

A. Figure 3

Let's talk first about the reproduction of figure 3. It is the worst case among the three figures. Where Sutton's paper gives a window of error from 0.19 to 0.25, I am providing error within the interval 0.09 to 0.16, displaying a -0.1 offset. This difference is important, but I must underline that it is only a quantitative one and not a qualitative one: the overall shape of the curve stays unaltered.

What can explain that difference? It is possible that Sutton utilizes some kind of specific sequence generation that is not exactly random, giving rise to a bias and a consequently higher error. The other possibility could lie in the Δw accumulation which I already pointed as an unclear part of the paper.

B. On TD(1) & TD(0)

As explained in the section II, the λ value modifies the kinetic of the weight update. What is shown by Fig.4.b is that it is the $\lambda = 1$ method that gives the worst results. This is a bit surprising as it was though to exhibit the faster updates and, on could think, consequently, the best convergence. However, it appears that it is in fact this rapid kinetic of updates which handicaps the TD(1) method. In fact, it makes it very dependents on the training set, and in a sense, the TD(1) method overfit the proposed training set and would have difficulties to apply correctly to new sequences. Eventually, however, with a very large training set, it will converge to the optimal solution. This is typical of least mean square estimators and frequentist statistics as I will discuss later.

On the contrary the TD(0) method is really slow for updating weights. That is why when α is too big it has difficulty to perform correctly: the updates are slow but imply really big modifications in the weight vector, modifications which might need to be corrected and refined by latter iterations, but the update being slow, this process of refinement is really gradual and almost overtaken by the huge α-step.

Following Sutton's statement [1], it is clear that the problem of the update of TD(0) could be overcome by uploading from a backward browsing of a random walk sequence: if we start from a winning end of +1, then this value could propagate through all the states visited since the beginning of the sequence. As he points out, this is not in line with the idea of an online learning, but it could by considered for some problems.

C. Stochasticity of results

After running the different experiments multiple times, one can only be struck by the randomness that characterizes them. In fact, depending on the nature of the sequences which are generated at random, the efficiency of the different learning methods can greatly vary. Note that, on my figure, I decided to plot all λ through all α, no holding at $\alpha = 0.4$ for $\lambda = 1$, because I did not saw the point of truncating $\lambda = 1$ values except for best display of other curves.

For figure 4, I launched an analysis for $TD(\lambda)$ errors at $\alpha = 0.3$ and $\alpha = 0.55$. I computed the errors over 100 experiences. I plot on Fig.5 the gaussian distributions extrapolated from the mean and standard deviation values of the obtained series of results for four lambdas.

It shows that there is a huge variability of lamba = 0 and lambda = 1 method, especially with high alpha values. On these graphs, the average qualifies the capacity of a given method to give reliable results, and the standard deviation traduces the consistency of these results. What is interesting to note is that, even if TD(0) is slightly better in average for high values of alpha, its standard deviation is really important and the results are thus less consistent than those from TD(1).

Moreover, we also see that for a lower value of alpha ($\alpha = 0.3$), the TD(0) is really better than TD(1), both in term of reliability and consistency of the results it provides. The figure also shows that, as seen in figure

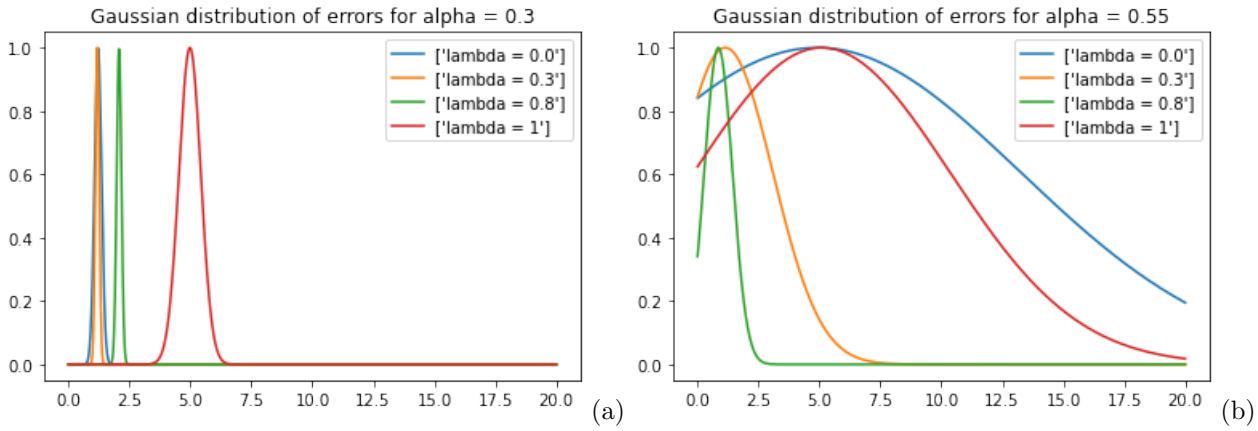

Fig. 5. Gaussian distributions of errors for $\alpha = 0.3$ (a) and $\alpha = 0.55$ (b)

5 from Sutton, the TD(0.3) is also really good, with even higher consistency than TD(0).

D. Learning with bias and a priori

From these results, I see some remarkable features. Firstly, the failing of TD(1) is not without recalling the failing of frequentist and unbiased methods for certain estimation problems such as seismic reflection. Similarly, even if TD(0) converge to the maximum-likelihood after multiple repetitions, it does not always give the best results. Only intermediary values of λ lead to bests results. This means that learning it more efficient when one takes into account previous experiences along the way and mitigates them with new information to outcome his predictions.

The analogy with differences between frequentist and bayesian approaches of statistics estimation seems to me very interesting here as we demonstrate that plugging a priori in our prediction leads to better outcomes and high efficiency. What has been observed on the dependence of results over the values of λ and α is also in adequacy with the Bayesian views on hyper-parameters and the necessity to develop stochastic algorithms to obtain working values for such parameters.

V. Conclusion

Through these experiments, I have been able to show that TD(1) is definitely not the best method for making prediction for systems that can be dealt with TD learning: systems that evolve through time and for which the probability of appearance of a future state depends on the actual occurring of certain previous states. I have shown that TD(0) is really good, which is in phase with it being consistent with the maximum-likelihood. However, as the outcomes of the method heavily relies on the data of the set and the temporal order in which they occur, I have also illustrated that intermediary values of lambda would, depending on the problem configuration, provide better outcomes. These values are about λ=0.3. Finally, as in most machine learning problems, the learning rate alpha must be adjusted -here again depending on the system structure- to provide best performances.

References

[1] Richard S. SUTTON, Learning to Predict by the Methods of Temporal Differences, Machine Learning 3:9 44, 1988.
[2] Richard S.SUTTON and Andrew G. BARTO, Reinforcement Learning, An Introduction (2sc Edition), The MIT Press, 2020
[3] Widrow, B., & Stearns, S. D. (1985). Adaptive signal processing. Englewood Cliffs. NJ: Prentice-Hall.