Project #1 Desperately Seeking Sutton

Hui Xia (Hxia40)

Georgia Institute of Technology

Git hash: 12697fbba3343c1f86182f7f7ffbaff8af4f9a8d

Abstract— **In this project, we aim to create an implementation to reproduce Figures 3, 4, and 5 from a reinforcement learning paper "Learning to Predict by the Methods of Temporal Differences". Here we describe our implementations, then compare the resulted figures with the original paper, and analyze the difference between them.**

I. Introduction

Reinforcement learning is a major category in machine learning. In 1988, Richard Sutton described a reinforcement learning method, which is temporal difference (TD) [1]. Sutton argues that for multi-step perdition problems, the TD method is superior compared with conventional supervised learning method. Sutton further reasoned that this is because the TD method could utilize the information that is delivered during the development of a temporal sequences in a multi-step perdition problems, while conventional supervised learning method cannot utilize sequential information. For example, to predict the incoming Saturday's weather, one can use supervised learning method, which estimate the weather with highest possibility using measurements collected from Monday, Tuesday, Wednesday … etc., while ignoring the sequential structure of the measurements alone the weekdays. The TD method, on the other hand, strive to collect such sequential structure and utilize it for the weather prediction.

To demonstrate the superiority of TD, Sutton proposed a simple 'random walk' model, measured the prediction error using TD over this model, and compared it with supervised learning. In this project we will firstly describe the methods taken to implement the 'random walk' experiments, then compare our results with Sutton's. By doing so, we target to gain understanding of the reinforcement algorithm TD.

II. Method

A. The Random Walk Model

As most real-world multi-step perdition problems (such as the weather forecasting example described above) are too complex to model for accurate prediction results. For demonstrative purposes, Sutton proposed a simple bonded random walk model. This model contain seven states that are connected in series, which will be referred as states A to G, respectively. State A only connects with state B, which connects to both state A and state C. That is, these states are connected in alphabetical order, and states A and G serves as two edge states of this serial connection. For each experiment, a learning agent will always begin random walking at the middle of this series, which is at state D. At each step the agent moves to a neighboring state, either to the right or to the left with equal probability. If either edge state A or G is entered, the walk terminates. A typical walk might be DEDCBA. By mathematics calculation, Sutton proved that the true probabilities of a walk ending in the right edge state, G, when the random-walk agent is located in each of the other intermediate states, i.e. states B-F, are 1/6, 1/3, 1/2, 2/3, and 5/6, respectively [1]. Here, we will implement a TD algorithm to estimate these probabilities, and compare our estimation with the true probabilities.

B. Implementing TD Algorithm

In a multi-step perdition problem, Sutton suggested using a set of weight vectors with recency to take the sequential effect of the past steps into account. The weights for each state will be updated temporally. Using the random-walking model as an example, if in the most recent random-walk, the agent reached edge-state G, the states on the agent's travel history will have their weights of the possibility of reaching state G updated. The states that the agent past 'recently' to G will be updated with a higher weight. In general, the weights are updated using the equation below:

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

In the equation above, Δw_t stands for the value estimation change in the step t of the sequence. In the random-walk example, this represents the estimated possibility of the agent will end up at the state G. α is the learning rate of the algorithm, which we will discuss below. P_{t+1} and P_t are the 'value' at step t and its next step. Especially, $P = 0$ for the last step of the sequence when the agent ends up with edge state A, and $P = 1$ when the agent ends with state G (this is because that we are estimating the possibility of the agent finish its random-walk in state G).

$\sum_{k=1}^{t}\lambda^{t-k}\nabla_w P_k$ is the eligibility vector, which decides how past observation in a past sequence will affect the weights. Parameter λ is a number between 0 and 1. The TD algorithm using parameter λ will be referred as TD(λ). When $\lambda = 0$, i.e. TD(0), the weight increment will only be determined by the prediction associated with the most recent observation. In case of TD(1), all past observation of the agent will affect the final prediction in an equal manner, which is the TD implementation of the prototypical supervised learning method. When $0 < \lambda < 1$, all past observation will affect the final prediction, but more

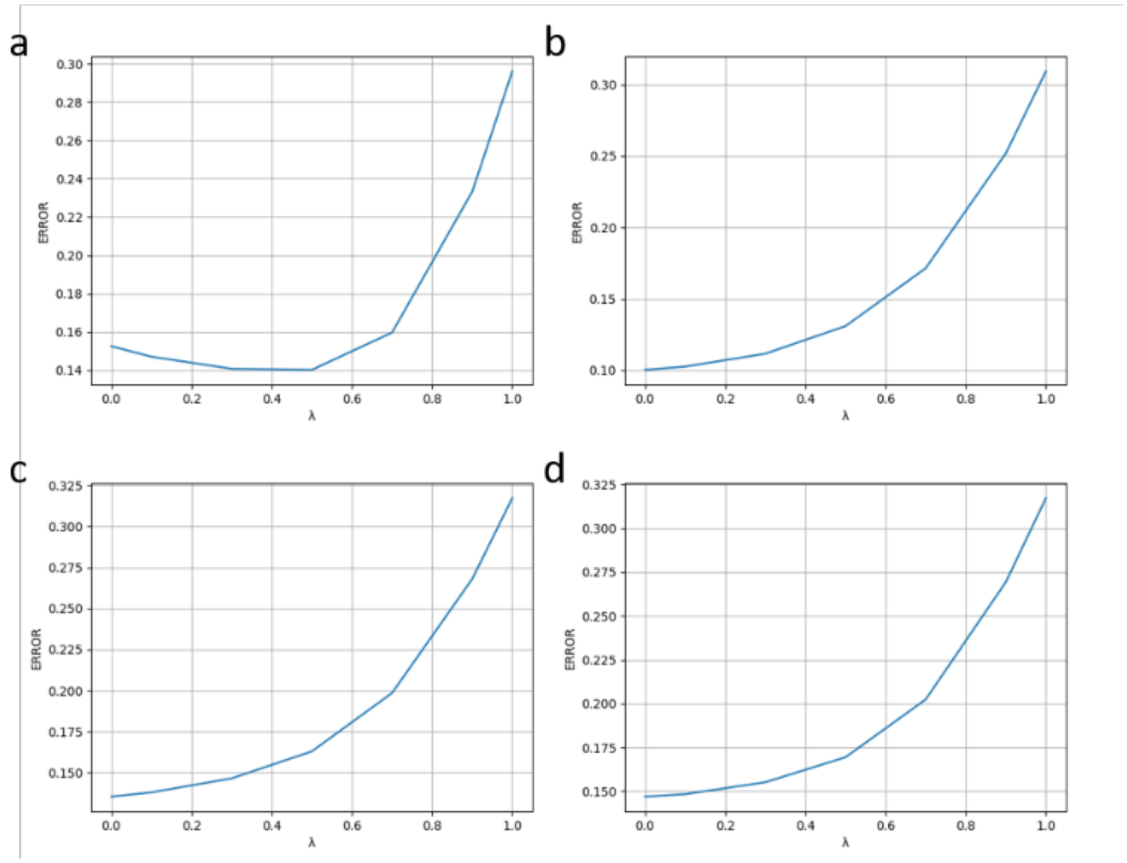

Figure 1. RMS error curve using various values of λ. (a) alpha = 0.001, delta = 0.001. (b) alpha = 0.002, delta = 0.001. (c) alpha = 0.01, delta = 0.001. (d) alpha = 0.02, delta = 0.001. Larger values of alpha will cause the algorithm to stop converging. Random seed = 1.

recent observations will affect the prediction with a higher weight, while the earlier observations will have a less effect on prediction, due to the dampening effect of multiplying λ along the sequence.

C. Reproducing Experiments from Sutton

In Sutton 1988, two computational experiments were performed using TD algorithm. The first experiment generated Sutton's Figure 3, and the second experiment generated Sutton's Figures 4 and 5. Here we will reproduce these experiments according to the paper's description. For both experiments, 100 training sets, each consisting of 10 random-walk sequences were randomly generated to perform all learning procedures. Weight increments were updated using TD(λ), which has been described above.

For the first experiment, the weight vector was updated after all sequences in a trainset has been presented and generated their respective Δw . Then the Δw from all sequences will be accumulated and used to update the weight vectors, until convergence. As Sutton did not describe in his paper on how the weight is converged, here we make **assumption (1)**, about the definition of convergence: if the root mean square error between last and this update (of the weight vectors) is smaller than a parameter delta, we would consider the weight vectors are converged. To justify this assumption, we implemented TD(λ) with various values of delta, trying to reproduce the corresponding experiment results of Sutton. Our **assumption (2)** is that the learning rate, α, obviously should be a value between 0 and 1, should ensure the convergence of the algorithm. The learning rate will decide how large steps the algorithm will take to perform updates. Larger value of learning rate could potentially make the algorithm approach to desired values faster, but if the 'step size' is too large, the algorithms may never converge.

The difference between the second experiment and the first experiment is, in the second experiment we no longer pursue convergence of weight vectors - each of the random-walk sequence will be presented to the algorithm only once, and the weight vectors will be updated after each of the sequence presentation (compared with the accumulated weight vector update in experiment 1). By updating the weight vectors only once per sequence, we can observe under what parameter condition of λ and α shall the algorithm converge faster. Thus our assumption (1) will be invalid for the second experiment, while assumption (2) remains reasonable.

As each sequence will be only presented once (rather than being presented until convergence), the total updates that applied on the weight vectors are extremely limited. For such a small sample pool, outlier case scenarios, such as an overly long random-walk sequence, could potentially bring unstable performance to the algorithm. For example, if an outlier case of the agent randomly wandered around states B and C fifty times,

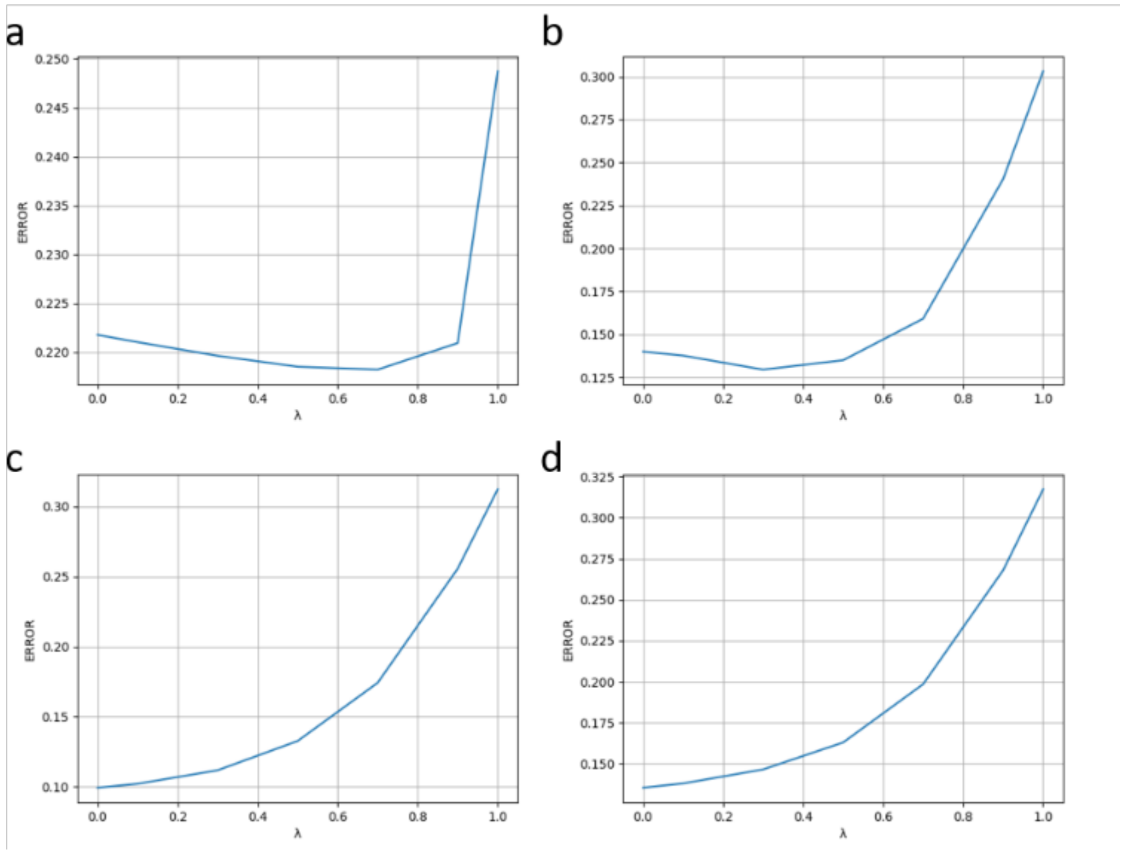

Figure 2. RMS error curve using various values of λ. (a) alpha = 0.01, delta = 0.05. (b) alpha = 0.01, delta = 0.01. (c) alpha = 0.01, delta = 0.005. (d) alpha = 0.01, delta = 0.001. Random seed = 1.

but ended up in edge state G, the weight vectors will be updated with a wrong value. This is especially true when we are using higher value of λ, such as λ =1, which treat all historical states equally. In such scenario, the deteriorative effect of data outlier will be more pronounced. Thus, we make **assumption (3)** that to reveal the performance difference among various values of TD(λ) curves, and reproduce Figure 4 of Sutton, a dataset that has a proper amount of data outlier is needed. We will discuss this topic further in the Results and Discussion section. For methods taken here, various numbers of sequences per dataset, as well as various maximum sequence length, have been investigated to find datasets have proper appearance of data outliers. We could justify this assumption if we can find such proper dataset that reproduces Figure 4 of Sutton.

III. Results and Discussion

A. Reproducing the First Experiment

To justify our assumptions for experiment 1, we implemented the TD(λ) algorithm with various values of alpha and delta. As illustrated in **Figures 1 and 2**, all combinations of parameters show a range of RMS error between 0.15 and 0.3, similar to the error range shown in Sutton Figure 3, which is 0.19-0.25. For graphs which has an alpha/delta ratio that is ≤ 1, i.e. **Figure 1a** and **Figures 2a, 2b**, the lowest error happens on a point where $0 < \lambda < 1$. For other parameter combinations when alpha/delta > 1, the lowest error on the λ curve happens on the point where $\lambda = 0$.

The observation that TD(0) does not always have the lowest RMS error is reasonable. In his paper, Sutton has reasoned that although linear TD(0) provide optimal predictions for finite training sets, it is expensive to perform such computation. That is, TD(0) is rather slow at propagating the prediction back alone the sequence, one state at a time. In scenarios when delta is relatively large, e.g. alpha/delta ratio ≤ 1 for our cases, it is very possible that the algorithm consider itself 'converged', and stopped weight updating prematurely for the points where λ is relatively small. Thus, we can justify our assumptions (1) and (2), as using proper RMS error delta and proper α, such as alpha = 0.01 and delta = 0.001, we could reproduce the same trend of the λ curve shown in Figure 3 of Sutton's paper. Thus, we reproduced Sutton's discovery that the performance of the TD algorithm improved rapidly as λ was reduced below 1, and reaches best at λ = 0.

It is noticeable that while we have reproduced the trend of the λ curve, it is however hard to reproduce the exact error range of the λ curve in Sutton's Figure 3, in which the error ranged between 0.19-0.25. We reasoned that this is due to the randomness of the dataset sample. As the random-walk datasets are generated, well, randomly, using such dataset to predict the true probability of reaching edge state G from each state could result in different errors. Using same parameters of α = 0.01 and delta = 0.001, we investigated the error range over the λ curve using different random seeds. While all of the resulted λ curves show the same trend that the performance of TD(0) is superior,

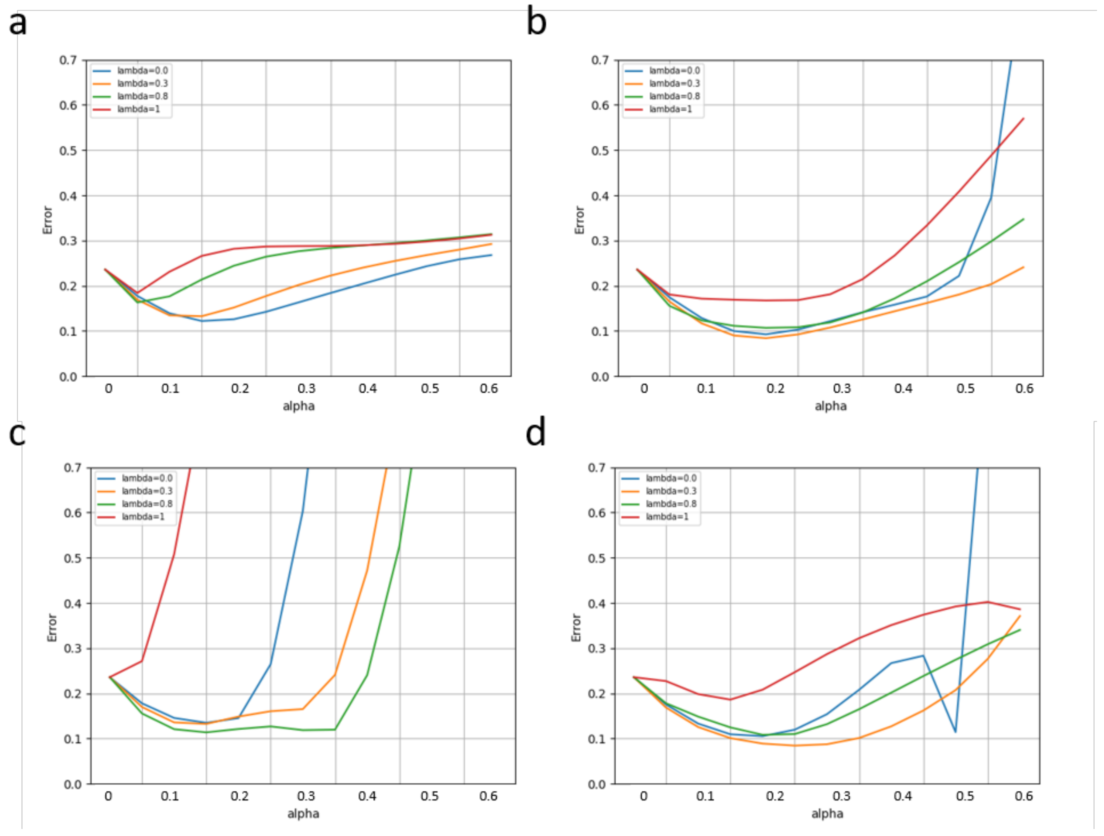

Figure 3. RMS error curve using various λ and alpha values. Ten sequences per dataset.

(a) maximum sequence length of 10 steps

(b) maximum sequence length of 30 steps

(c) maximum sequence length of 50 steps

(d) maximum sequence length of unlimited steps.

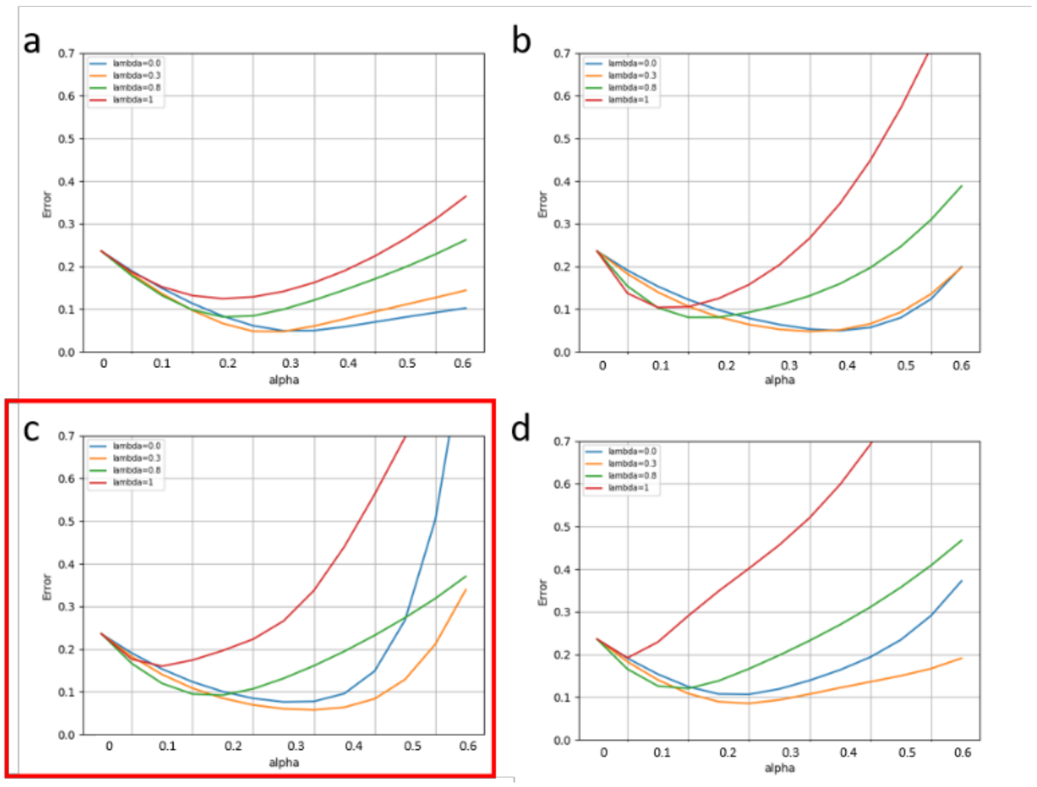

Figure 4. RMS error curve using various λ and alpha values. Seven sequences per dataset.

(a) maximum sequence length of 10 steps.

(b) maximum sequence length of 30 steps.

(c) maximum sequence length of 50 steps, which is similar to Sutton's Figure 4.

(d) maximum sequence length of unlimited steps.

the error ranges could vary. When the random seed values of 1, 2, 3, 4, and 5, the corresponding RMS error ranges between 0.12-0.32, 0.04-0.08, 0.12-0.24, 0.07-0.1, and 0.08-0.28, respectively. Thus, without knowing the exact dataset Sutton used to train his TD algorithm, it is well beyond the scope of this project report to reproduce the exact error range of Sutton's Figure 3.

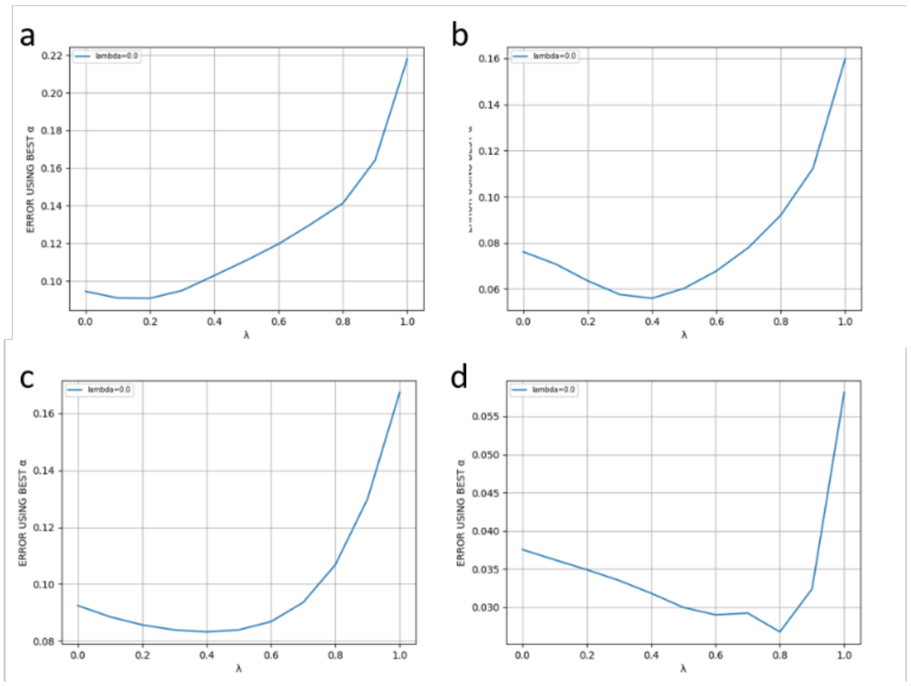

Figure 5. RMS error curve using various λ and 'best' alpha values. Maximum sequence length of 50 steps.

(a) Five sequences per dataset.

(b) Seven sequences per dataset.

(c) Ten sequences per dataset.

(d) Twenty sequences per dataset.

B. Reproducing the Second Experiment

We further implemented the procedures described by Sutton for the second experiment. Compared with the first experiment, the second experiment focus on how quickly can a TD(λ) algorithm learn. This feature is useful in many real-world problems, as data for real-world problems are usually limited, and/or expensive to obtain. To simulate this case, the data we are using here is also limited – only 100 datasets are available, and each sequence will only be presented to the algorithm once. Because the data is limited, it is important to ensure a proper representation of data outliers appearing in the datasets.

The purposes of doing so is, the dataset to be used here should have a proper number of data outliers to demonstrate the difference between curves with higher and lower λ values. The curves that are drawn with higher λ values are easier to be negatively affected by data outliers (due to the nature of high TD(λ) of distributing higher weight for older observation). Thus, the number of outliers that we use can neither too low to demonstrate no extreme error among all λ curves, nor too high to make all λ curves demonstrate high errors. We noticed that if we set the maximum threshold of the sequences length to be 20 steps, most (if not all) of the outliers in the datasets will be removed, featured by the relatively low error in all λ curves, as shown in **Figure 3a and 4a**. On the other hand, if we do not set limitation for maximum sequence length, high errors will appear in all λ curves in an incomprehensible manner, as shown in **Figures 3d** and **4d**.

After scanning through various sequence length thresholds and sequence number per dataset, our best deliverable to reproduce Sutton Figure 4 is shown in **Figure 4c** in the red frame, which use seven sequences per dataset, and has a maximum sequence length of 50 states. Thus, our assumption (3) that "to reveal the performance difference among various values of TD(λ) curves, and reproduce Figure 4 of Sutton, a dataset that has a proper amount of data outlier is needed" could be justified. More importantly, although the λ curves plotted using ten sequences per dataset (which is the what Sutton has done, plots in **Figure 3**) appear different compared with Sutton Figure 4, this result still agree with Sutton's conclusions drawn from his Figure 4. Which is, first, the best results obtained with intermediate values of α; second, for all values, the TD(1) procedure produced the worst estimates. Thus, we can conclude that our reproducing work on Sutton Figure 4 is reasonable.

Compared with Sutton's Figure 4, which is largely dependent on the appearance frequency of data outliers, Sutton's Figure 5 is easier to reproduce, as more certainty is involved. **Figure 5** demonstrate λ curves using various maximum sequence length. All of them show similar trend as Sutton's Figure 5, in which an intermediate λ value have the best performance. In Sutton's Figure 5 (which uses 10 sequences per dataset), such λ value is somewhere near 0.3. Similar to what happened in reproducing Sutton's Figure 4, our closest reproduction is still the plot drawn using 7 sequences per dataset, which has a best λ value between 0.3 and 0.4.

Conclusions

Via working on this project report, we gained understanding of TD algorithm, by justifying our three assumptions and reproducing Sutton's Figures. The major pitfalls that we have encountered in this project is realizing what a reasonable assumptions are, learning to realize how randomness can affect the figures (especially for Sutton's Figure 4), and learning to focus on the major conclusions.

References

[1] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine learning,* vol. 3, no. 1, pp. 9-44, 1988.