Reinforcement Learning Project 1 Report Desperately Seeking Sutton - TD(λ)

Xingyan Liu (202264690069@mail.scut.edu.cn), Shiyi Wang (202230055267@mail.scut.edu.cn)
Wolin Liang (202264690304@mail.scut.edu.cn), Haoming Jia (202264690267@mail.scut.edu.cn)

***Abstract*—This project report discuss the fundamentals of Temporal Difference (TD) Learning and provides details of steps in reproducing experiment results in Sutton's paper "Learning to Predict by the Methods of Temporal Differences" using the bounded random-walk process.**

I. Introduction

Reinforcement learning aims to enable an agent to learn the optimal strategy for completing tasks through interaction with the environment, without explicit programming. One of the earliest discussions on reinforcement learning can be traced back to Minsky's *Steps Toward Artificial Intelligence*. Since then, the theory of reinforcement learning has continuously developed and found widespread applications in robotics [1], and game theory [2].

With the advancement of deep neural networks, deep reinforcement learning [3] has significantly empowered the field, providing it with more robust learning capabilities. The theoretical foundation of deep neural networks is gradient descent [4], which is also a focus of this paper. Gradient descent is not only the principle behind neural network learning but also the basis of temporal difference learning [5].

A cornerstone of this field is the concept of temporal difference (TD) learning, introduced by Richard S. Sutton in his seminal 1988 paper, "Learning to Predict by the Methods of Temporal Differences." This paper laid the groundwork for many of the advancements seen in reinforcement learning today.

Temporal difference learning represents a novel approach to prediction problems, where the objective is to forecast future outcomes based on current and historical data. Unlike traditional supervised learning, which relies on fixed target values, TD learning dynamically updates predictions by leveraging the discrepancies between successive predictions—hence the term "temporal difference". This innovative method allows for more flexible and adaptive learning processes, which are particularly beneficial in environments where outcomes are uncertain and evolve over time.

The implications of Sutton's temporal difference learning extend far beyond the initial prediction tasks it was designed to address. It has become a fundamental building block for more advanced reinforcement learning algorithms, such as Q-learning[6] and SARSA[7], which have been applied successfully in a wide range of domains, from game playing to robotics. By enabling agents to learn effectively from their interactions with the environment, TD learning has significantly advanced the capabilities of artificial intelligence systems.

In this paper, we explore the principles of temporal difference learning as introduced by Sutton, examining its theoretical underpinnings, practical applications, and impact on the field of reinforcement learning. Through this exploration, we aim to highlight the enduring relevance and potential of TD learning in developing intelligent systems capable of adapting to complex and changing environments.

II. Temporal Difference Learning Methods

Temporal Difference (TD) learning is a core technique in **reinforcement learning**, primarily used to solve prediction and decision-making problems. It updates models by comparing the **differences between consecutive predictions over time**, unlike traditional supervised learning methods. This allows TD to perform incremental learning even **without explicit labels or outcomes**.

A. *Difference between TD and Traditional Supervised Learning*

- **Supervised learning** relies on explicit labels, adjusting the model by comparing the prediction with the actual outcome. It requires a complete dataset, where each input has a corresponding output.
  For simplicity, Sutton suggests a special case where the prediction at time t is a function of the current state x_t and some learnable weights w: $P(x_t, w)$, where $w(i)$ and $x_t(i)$ are the ith dimension of w and x_t:

$$P_t = w^T x_t = \sum_i w(i) x_t(i)$$

  The weights update/learning procedure is derived as below where α is the learning rate:

$$\Delta w_t = \alpha(z - P_t)\nabla_w P_t = \alpha(z - w^T x_t)x_t$$

  For supervised learning schema, vector weights w is only updated after a complete observation-outcome sequence:

$$w_{\text{new}} = w_{\text{old}} + \sum_{t=1}^{m} \Delta w_t$$

Liu Xingyan: Responsible for code programming, results collation and analysis.Wang Shiyi: Focuses on literature research and problem modeling, specifically defining the problem environment.Liang Wolin: Handles the introduction of related work and paper analysis.Jia Haoming: Analyzes the experimental results and their significance.

Manuscript revised September 30, 2024.

- **TD learning** is different in that it does not require the final result immediately. Instead, it updates based on the **difference between consecutive predictions over time**. For instance, in a game, TD can estimate the future return of the current state without waiting for the game to end. This makes TD **more efficient in dynamic systems**.

B. Basic Concept of TD Learning

The core idea of TD learning is to update the value of a state by comparing the predictions at two consecutive time steps. The update rule is:

$$w_{new} = w_{old} + \sum_{t=1}^{m} \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \nabla_w P_k$$

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

where:

- P_t and P_{t+1} are the predicted values at times t and $t+1$,
- α is the learning rate,
- λ is the trace-decay parameter,
- $\nabla_w P_k$ is the gradient of the predicted value with respect to the weights at time k.

C. Introduction to TD(λ) Method

TD(λ) is an extension of the basic TD method that introduces **eligibility traces** and a **decay factor** λ, enabling the method to update the value of not just the current state, but also the previous states based on their eligibility.

The eligibility traces can be incrementally computed using the following formula:

$$e_{t+1} = \sum_{k=1}^{t+1} \lambda^{t+1-k} \nabla_w P_k = \nabla_w P_{t+1} + \lambda e_t$$

This mechanism allows for more efficient computation of weight updates while maintaining eligibility traces over time.

para3

III. Experiment & Results

In this section, we provide a comprehensive technical account of the reproduction of Figure 3-5 from Sutton's seminal work. Initially, the theoretical or optimal predicted values for non-terminal states in the random walk problem are derived using the methodology outlined in the paper. Two key experiments are then conducted: the first experiment focuses on repeated training sequence presentations across varying λ values, while the second experiment investigates the effect of single-sequence presentations under unbiased initialization, evaluating various λ-learning rate pairings.

A. Theoretical Values of Weights in Random-Walk

Let $\mathbf{Q}$ represent the transition probability matrix between non-terminal states, with Q_{ij} as the probability of transitioning from state i to state j, and $\mathbf{h}$ as the transition probability vector to the terminal state G. The expected outcome z for starting in state i is:

$$E[z \mid i] = \left[\sum_{k=0}^{\infty} \mathbf{Q}^k \mathbf{h}\right]_i = \left[(\mathbf{I} - \mathbf{Q})^{-1}\mathbf{h}\right]_i$$

Thus, the theoretical weights for the non-terminal states B, C, D, E, F are:

$$\mathbf{w} = (\mathbf{I} - \mathbf{Q})^{-1}\mathbf{h}$$

For the random-walk, $\mathbf{Q}$ and $\mathbf{h}$ are defined as:

$$\mathbf{Q} = \begin{pmatrix} 0 & 0.5 & 0 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 \\ 0 & 0.5 & 0 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0.5 & 0 \end{pmatrix}, \quad \mathbf{h} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.5 \end{pmatrix}$$

The resulting theoretical weight vector is:

$$\mathbf{w} = \begin{pmatrix} \frac{1}{6} \\ \frac{1}{3} \\ \frac{1}{2} \\ \frac{2}{3} \\ \frac{5}{6} \end{pmatrix}$$

Algorithm 1 TD(λ) with Repeated Presentations

```
INPUT: Python lists of $\lambda$ values
OUTPUT: Dictionary storing the mean RMSE for each $\lambda$: rmse_dict

INITIALIZE list of $\lambda$, learning rate $\alpha$, stopping criteria $\epsilon$
for each $\lambda$ do
  for each training set do
    INITIALIZE weights $w$
    while changes in $w < \epsilon$ do
      for each training sequence do
        INITIALIZE error err
        for each state do
          CALCULATE err and $\Delta w$
        end for
        ACCUMULATE $\Delta w$ across sequences
      end for
      UPDATE $w$ using accumulated $\Delta w$
    end while
    CALCULATE RMSE for the current training set
  end for
  CALCULATE mean RMSE across all training sets
end for=0
```

B. Repeated Presentations Paradigm

100 randomly generated training sets, each containing 10 bounded random-walk sequences, were prepared using the methods outlined in Section II of this report.

In the repeated presentations paradigm, for a given λ value, the learner is exposed to the 10 random-walk sequences in a training set multiple times until convergence. The weight adjustments, Δw, are accumulated across the 10 sequences, and weight updates occur only after a full epoch. This process is repeated for all 100 training sets, and the average root mean squared error (RMSE) is computed between the learned weights and the ideal weights of the non-terminal states: X_B, X_C, X_D, X_E, X_F.

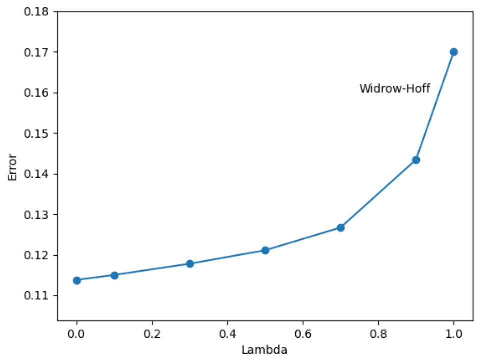

Fig. 1. Average errors on the random-walk with repeated presentations.

Figure 1 illustrates the average error between the learned weights w and the ideal weights $\left[\frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{5}{6}\right]^T$ for various values of λ. The results show that TD(1) (Widrow-Hoff) yields the highest error compared to other values of λ. This could be attributed to TD(1)'s tendency to minimize the error between predictions and outcomes within the training set, without necessarily improving future predictions. In contrast, TD(0) achieves the lowest mean error, as it prioritizes the most recent state, aligning with the maximum likelihood estimation (MLE) for a Markov process.

The reproduced results closely match those in Figure 3 of Sutton's paper, though the exact error values are somewhat lower. This discrepancy could be due to differences in the randomly generated training sets, as well as variations in initial weight settings and the learning rate α, which were not explicitly defined in Sutton's original experiment. In this reproduction, a learning rate of $\alpha = 0.01$ and unbiased weight initialization were used to ensure convergence in the TD learning process.

C. Single Presentation Paradigm

The same 100 randomly generated training sets used in the previous experiment were utilized in this paradigm. The weights were initialized with an unbiased value of 0.5 for all non-terminal states.

In the single presentation paradigm, for each combination of λ and α, the learner is exposed to the 10 random-walk sequences only once per training set. Unlike the repeated presentation paradigm, where weight updates are performed after a full epoch, in this case, the learner updates the weights w at the end of each random-walk sequence. This process is repeated for all 100 training sets, and the average root mean squared error (RMSE) is computed between the learned weights and the ideal weights.

Algorithm 2 TD(λ) with Single Presentation

```
INPUT: Python lists of $\lambda$ and $\alpha$ values
OUTPUT: Python dictionary stores the mean RMSE for each ($\lambda, \alpha$) pair: rmse_fig_4

INITIALIZE list of $\lambda$, list of $\alpha$, learning rate $\alpha$
for each $\lambda$ do
    for each $\alpha$ do
        for each training set do
            INITIALIZE weights $w$
            for each training sequence do
                INITIALIZE error err
                for each state do
                    CALCULATE err and $\Delta w$
                end for
                UPDATE $w$ using $\Delta w$
            end for
            CALCULATE RMSE for the current training set
        end for
        CALCULATE mean RMSE across all training sets
    end for
end for=0
```

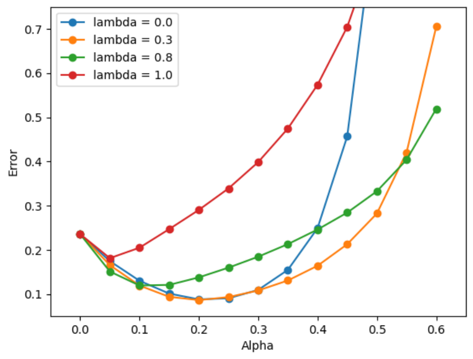

Fig. 2. Average errors on the random-walk with single presentation for $\lambda = 0, 0.3, 0.8, 1$.

Various combinations of λ and α were tested to analyze how different learning rates impact the learning outcomes.

The reproduced figure 2 illustrates the average error between the learned weights w and the ideal weights $[\frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{5}{6}]^T$ for different values of λ and α. Each line in the figure represents the mean error for a specific value of λ across different learning rates α. The learning rate significantly affects the performance of TD learning. Lower α values (approximately between 0.2 and 0.3) produced the lowest prediction errors, indicating that a well-balanced learning rate achieves both fast convergence and high-quality weight updates.

As before, TD(1) performed the worst regardless of the α value, demonstrating the limitations of the conventional Widrow-Hoff approach in the bounded random-walk problem. In contrast, TD with $\lambda < 1$ produced better predictions.

While the reproduced results closely align with figure 4 from Sutton's paper, some discrepancies in the points where the lines for $\lambda = 0$ and $\lambda = 0.8$ intersect may be due to the different randomly generated training sets used in our experiments.

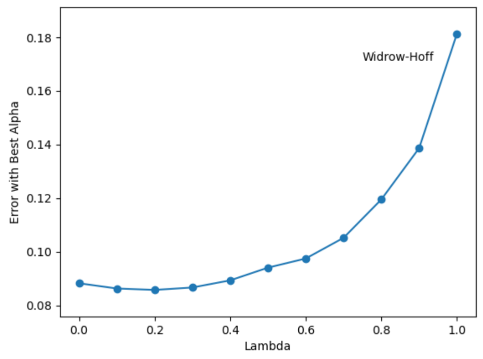

Fig. 3. Average errors at best α on the random-walk problem with single presentation for λ.

The reproduced Fig. 3 illustrates the best mean errors for each λ value, providing a more condensed view compared to Fig. 4. From the graph, it is evident that TD(1) produces the poorest predictions. The optimal λ in the single presentation experiment is approximately 0.3. Similarly, the reproduced results closely align with those of the original Fig. 5.

IV. Results Analysis

The analysis of our experimental results offers valuable insights into Temporal Difference (TD) learning, closely aligning with Sutton's foundational findings. In the repeated presentations paradigm, TD(1), similar to the Widrow-Hoff method, showed higher error rates compared to other λ values, consistent with Sutton's observations. This indicates that our setup effectively captures the essential dynamics of TD learning.

In the single presentation paradigm, TD(0) outperformed other methods, reflecting its reliance on recent states and its congruence with maximum likelihood estimation in Markov processes. This further confirms the robustness of TD learning across different conditions.

Minor deviations were observed, such as slightly lower error rates in our experiments, likely due to stochastic training set generation and variability in initial weights and learning rate (α = 0.01). These factors may have contributed to the faster convergence and improved performance seen in our setup. Discrepancies in the intersection points of $\lambda = 0$ and $\lambda = 0.8$ in our reproduced Figure 4 highlight the subtle impact of initial conditions and dataset specifics on TD learning outcomes.

These variations underscore the sensitivity of TD learning to environmental factors and parameter choices, particularly the choice of λ. The observed differences in error rates suggest that contemporary computational practices or refinements might enhance TD learning performance. Moreover, the sensitivity of TD learning to λ values emphasizes the need for careful consideration of this parameter to optimize predictive accuracy.

V. Conclusion

This report details the setup and implementation of the bounded random-walk process. Section II covers the generation of training sets, while Section III synthesizes the mathematical foundations from Sutton's paper, explaining three learning paradigms: Supervised Learning (Widrow-Hoff), Incremental Learning TD(1), and the more general TD(λ). Section IV presents the algorithms used to reproduce figures 3-5 in Sutton's work, with a focus on two learning paradigms: repeated presentations and single presentation, along with their corresponding results and analysis.

The research process, encompassing paper review, code design, implementation, and result analysis, has provided a deeper understanding of Temporal Difference (TD) Learning, particularly regarding the impact of hyperparameters α and λ on prediction outcomes.

To ensure optimal results in future implementations of TD learning, careful evaluation of prediction problems (single-step vs. multi-step) is essential, alongside selecting an appropriate learning rate to balance convergence speed and accuracy. Finally, selecting the optimal λ value based on problem specifics will leverage the strengths of TD learning, including incremental updates and improved predictive accuracy.

References

[1] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 2013.

[2] Drew Fudenberg and David K Levine. *The Theory of Learning in Games*. MIT Press, 1998.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[4] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[5] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988.

[6] Christopher John Cornish Hellaby Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, 1989.

[7] Gavin A Rummery and Mahesan Niranjan. On-line q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, University of Cambridge, Department of Engineering, 1994.