git commit hash:5160c6a1c9c96735510c6e3fb0063c3c44e295e2

Replication of an article by Richard Sutton [1] titled *"Learning to predict by the methods of temporal differences"*

Milad Abedi
Georgia Tech
Department of Computer Science
Atlanta, USA
miladabedi@gatech.edu

***Abstract*—This document is a report of an attempt to replicate parts of Richard Sutton's 1998 paper titled "Learning to predict by the methods of Temporal Differences". At the time of publication of the article in question, state of the art in prediction problems was dominated mostly by the supervised learning methodology, where the prediction models were adjusted/trained in accordance with the observed final outcomes. In this paper, Sutton presented the Temporal Difference (TD) reinforcement learning algorithm, which in contrast to the prevailing methods of it's era allows for incremental update and learning on-the-go. In his paper, Sutton has demonstrated the feasibility and efficacy of this model through a number of simulation studies, the replication of which is the subject of the report at hand.**

I. Introduction

The task of prediction of the outcome of at least partially unknown processes has been the common theme of a multitude of disciplines, such as Machine Learning, Statistical Inference, etc. A common analytical framework sought after in the industry is to use the past experience to build insight that would allow for better decision making in the present and future. This endeavor can be, made further challenging by the non-deterministic nature of the underlying process, lack of knowledge regarding the environment, and the high expense associated with production of more experience.

For the purposes of this paper, we can divide the prediction problems in to two categories of single step and multi-step problems. In the former case, the experiences are in the form of observation-outcome pairs, whereas in the latter case the final outcome will be revealed after a sequences of observations and experience, whilst receiving feedback (positive or negative) during the intermediate cases. The single-step category can be very effectively handled by the supervised learning paradigm. However, in the multi-step category, the TD approach, due to it's ability to perform incremental updates could show superior performance.

In order to facilitate a meaningful comparison of the supervised learning paradigm with the TD learning paradigm, the author (Sutton) has presented a particular case of TD learning, using a linear state-value model. The author has further proven that theoretically, the approach in question will result in exactly the same final predictive model as a well-know supervised learning method called the Widro-Huff algorithm. However, due to it's incremental learning nature, the TD learning method in question will be more effective (faster learning) than it's counterpart.

The author then proceeds to compare a wider family of the TD learning methods, called $TD(\lambda)$. By utilizing a simulated learning problem, the author has presented an empirical comparison of the $TD(\lambda)$ method with it's counterpart in the supervised learning category, namely the Widro-Huff algorithm. Firstly, the author has compared the Widro-Huff algorithm with the family of $TD(\lambda)$ algorithm, and demonstrated that the latter can produce better prediction of the future. In a second experiment, the author has compared the ability of two approaches to learn quickly. In this experiment the family of $TD(\lambda)$ algorithms showed a distinct superiority. In the third experiment, the author has compared the accuracy of prediction for the two approaches, with each approach having been fine tuned for it's best performance. The TD(λ) algorithm, again showed superiority.

II. Results and Discussion

In this section we will replicate the experiments performed by Sutton, in the same order that was presented in his article.

A. Simulation Set-Up

The simulations performed in Sutton's paper are conducted in a simulated environment. In Fig. 1, a diagram of the simulated environment has been presented. The agent begins every experience at state D, and from there moves either to the left or to the right with equal probability of 50%. In the States B,C,D,E,F the agent will continue to make a choice to move left or right with equal probability. States of A,G are both terminal states, however a move from State F to G will yield a reward of +1, whereas a move from B to A will result in a reward of 0.

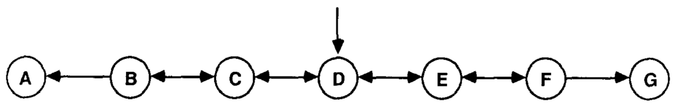

Fig. 1. The virtual virtual environment.

Each experiment has been conducted on 100 data sets, and the reported error values in the results represent an average of the 100 experiments. Each data set contains 10 sequences of an agent starting at state D and finishing at a terminal node.

The State-Value function has been represented as a linear model, as follows:

$$V_{(S)} = W^T \cdot \Phi_{(S)} \tag{1}$$

where:
W: A vector of length 5 representing the weights of the model
$\Phi(s)$: A vector of length 5 that returns the feature vector for a given state S

we define the feature vector $\Phi(s)$ to be a representation of the current states, where the elements of the vector are associates with states B,C,D,E,F. If the agent is present in any of these states the corresponding value in the feature vector will be 1, and the rest of the values will be zero. For instance, if the agent is at state B, the feature vector will be as follows:

$$\Phi_{(B)} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{2}$$

The state value for this case can be calculated by inserting the value of $\Phi_{(B)}$ in equation (1), resulting in the following:

$$V_{(B)} = W^T \cdot \Phi_{(B)} = \begin{pmatrix} W_1 \\ W_2 \\ W_3 \\ W_4 \\ W_5 \end{pmatrix}^T \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = W_1 \tag{3}$$

Similarly, it can be demonstrated that value of each state is equal to its corresponding weight, i.e.:

$$V_{(C)} = W_2;\ V_{(D)} = W_3;\ V_{(E)} = W_4;\ V_{(F)} = W_5$$

The ideal values for the weight vector can be analytically calculated (refer to section 4.1 of [1]) to be equal to the following:

$$W_{ideal} = \begin{pmatrix} 1/6 \\ 2/6 \\ 3/6 \\ 4/6 \\ 5/6 \end{pmatrix} \tag{4}$$

B. Experiment 1

In his first Experiment, Sutton has presented a comparison of the $TD(\lambda)$ algorithm for a range of values for the λ parameter. As mentioned in the previous section, Sutton has presented the proof that $TD(\lambda = 1)$ is equivalent to the Widro-Huff algorithm, thus this experiment can also be interpreted as comparison of a family of $TD(\lambda)$ algorithms with their supervised-learning counterpart.

For any given value of λ, the agent is trained on a data set consisting of 10 sequences of interactions. At the end of the training, the updated weights of the linear model for the state-value function are compared to the ideal weights using RMSE (Root Mean Squared Error) as the metric. This is repeated for 100 times, once for each data set, and the average value for RMSE's gets reported as the Error. The author has modified the original algorithm for this experiment such that the weights do not get updated after each sequence, but rather the values of weight change are summed for all sequences in the data set, and applied only after the last sequence has been processed. For each data set, the procedure was repeated until convergence of the W vector. In the replication of the study, the following condition has been used for convergence:

$$Converged : \frac{||\Delta W||}{||W||} < 0.01$$

The pseudo-code for the learning procedure is as follows:

Algorithm 1 Experiment 1

```
RMSE ← Empty Extendable List
for $training - data$ in $data - corpus$ do
    $W \leftarrow \overrightarrow{0.5}$
    while not converged do
        $\Delta W \leftarrow \overrightarrow{0}$
        for $Sequence$ in $training - data$ do
            $E_{(S)} \leftarrow \overrightarrow{0}$
            for $S, S', R$ in $Sequence$ do
                $E_{(s)} \leftarrow \lambda \cdot E_{(s)} + \Phi_{(S)}$
                $Err \leftarrow R + W \cdot \Phi_{(S')} - W \cdot \Phi_{(S)}$
                for $S''$ in $States$ do
                    $\Delta W \leftarrow \Delta W + \alpha \cdot Err \cdot E_{(s'')}$
                end for
            end for
        end for
        $W \leftarrow W + \Delta W$
    end while
    add $RMSE(W_{ideal}, W)$ to the RMSE List
end for
```

The final results of the experiment can be viewed in Fig. 2. As seen in Fig.2, the Widro-Huff method (equivalent to TD(1)) has shown the worst performance, and the TD(0) case has shown the best performance, in terms of their ability to accurately understand the underlying process and predicting state values. This is not consistent with intuition, since the Widro-Huff algorithm operates based upon the notion of RMSE minimization. However, one must keep in mind that the minimization in that case is restricted to the training data. TD(0) however, is equivalent to performing expectation maximization on the underlying Markov Process. Since the virtual environment as defined in Fig. 1 is a Markov Process, the TD(0) has an advantage of having a more robust optimization goal, and thus being less susceptible to over-fitting to the training data.

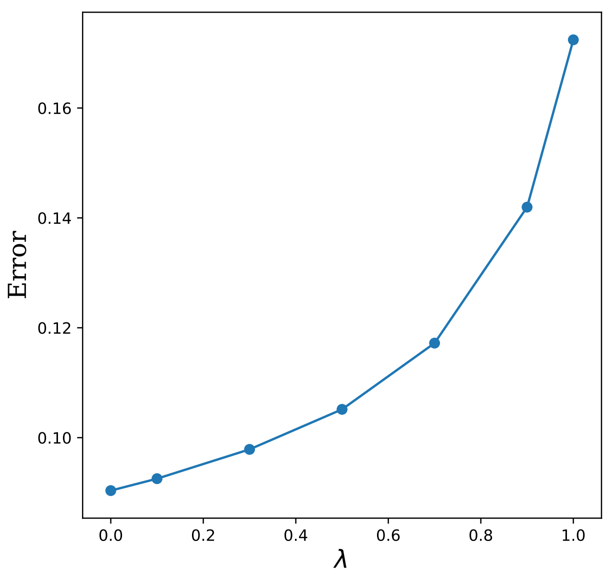

Fig. 2. The Error values between the models' prediction of the weights and the ideal weights for a range of Lambda values. Note that the case of $\lambda = 1$ is equivalent to the Widro-Huff supervised learning algorithm

C. Experiment 2

In The second experiment, the author intends to compare the performance of different procedures in terms of their training speed. To do so, the author has compared the Error for different procedures after a single update has been applied to the weights, with the assumption that the procedure that results in the greatest amount of improvement in the RMS error between the computed weights and the underlying ideal weights, is likely to have the best performance in terms of speed of learning.

For this Experiment, weights are updated after every single sequence has been observed by the update procedure. However, unlike Experiment 1, the update will not continue until convergence, and the process will be halted after the first round of observing the sequences, and the RMS Error between the updated weights and the ideal weight vector will be recorded. This procedure will be done for each of the 100 available data sets, resulting in 100 RMSE values. The average of these values will be reported as the (Expected) Error. In order to provide a more clear understanding of the performance of each procedure, the author has performed the above-mentioned procedure for a range of values for the training rate (α). The following is a pseudo-code for Experiment 2:

Algorithm 2 Experiment 2

```
RMSE ← Empty Extendable List
for $training - data$ in $data - corpus$ do
    $W \leftarrow \overrightarrow{0.5}$
    for $Sequence$ in $training - data$ do
        $\Delta W \leftarrow \overrightarrow{0}$
        $E_{(S)} \leftarrow \overrightarrow{0}$
        for $S, S', R$ in $Sequence$ do
            $E_{(s)} \leftarrow \lambda \cdot E_{(s)} + \Phi_{(S)}$
            $Err \leftarrow R + W \cdot \Phi_{(S')} - W \cdot \Phi_{(S)}$
            for $S''$ in $States$ do
                $\Delta W \leftarrow \Delta W + \alpha \cdot Err \cdot E_{(s'')}$
            end for
        end for
        $W \leftarrow W + \Delta W$
    end for
    add $RMSE(W_{ideal}, W)$ to the RMSE List
end for
```

The final results of this experiment can be seen in Fig. 3.

As seen in Fig. 3, the performance of the algorithms in terms of their rate of learning, depends both on λ and α. While the Widro-Huff ($TD(1)$) has shown the worst performance, all the $TD(\lambda)$ cases with $\lambda < 1$ have shown better performance.

D. Experiment 3

In light of the results of Fig. 3, the author has performed the third experiment to compare the ability of different algorithms in terms of their rate

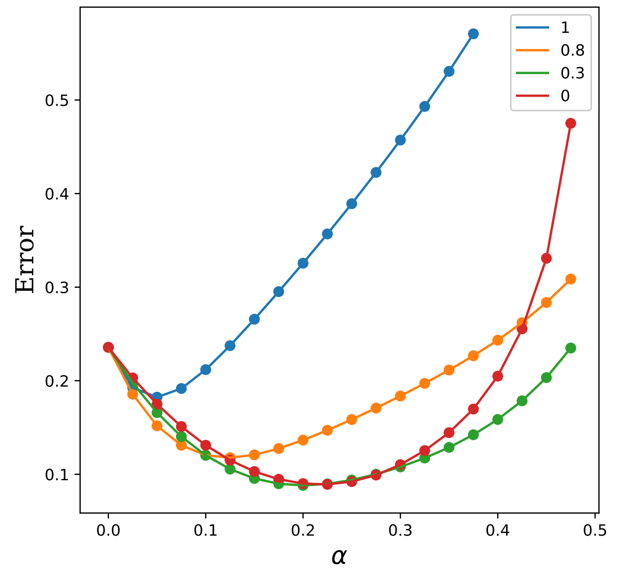

Fig. 3. The Error values between the models' prediction of the weights and the ideal weights for a range of λ and α values, after a single round of training.

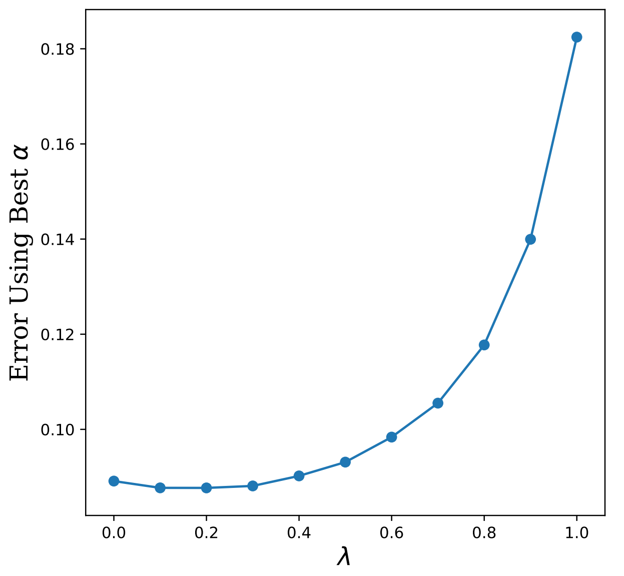

Fig. 4. The Error values between the models' prediction of the weights and the ideal weights for a range of λ values paired with their optimal α counterparts, after a single round of training.

of learning, without the interference of the influence of choosing the α values, thus comparing only the effects of λ on the learning rate. However, as seen in Fig. 3, choosing any constant value for the whole experiment will give an advantage to some procedures over others. As such, the author has assigned to each case, the value of α that is optimal for that given λ based on the results of Experiment 2.

The following is the utilized pseudo-code for Experiment 3:

Algorithm 3 Experiment 3

```
RMSE ← Empty Extendable List
for $training - data$ in $data - corpus$ do
    $W \leftarrow \overrightarrow{0.5}$
    for $Sequence$ in $training - data$ do
        $\Delta W \leftarrow \overrightarrow{0}$
        $E_{(S)} \leftarrow \overrightarrow{0}$
        for $S, S^{'}, R$ in $Sequence$ do
            $E_{(s)} \leftarrow \lambda \cdot E_{(s)} + \Phi_{(S)}$
            $Err \leftarrow R + W \cdot \Phi_{(S')} - W \cdot \Phi_{(S)}$
            for $S^{''}$ in $States$ do
                $\Delta W \leftarrow \Delta W + \alpha_{optimal} \cdot Err \cdot E_{(s'')}$
            end for
        end for
        $W \leftarrow W + \Delta W$
    end for
    add $RMSE(W_{ideal}, W)$ to the RMSE List
end for
```

Fig. 4 presents the results of this experiment. As can be seen in Fig. 4, the optimal value for λ, as far as the rate of learning is concerned, is not $TD(0)$, despite the fact that full implementation of $TD(0)$ (until convergence) produces the most accurate estimation of the W vector (See Experiment 1). This can be ascribed to the fact that $TD(0)$, in every step, only updates the weight values for the preceding state. By contrast, in cases where $\lambda \neq 0$, all the states that have preceded the current state, receive an update on their estimated values, thereby resulting in faster learning.

III. Challenges and pitfalls encountered

During this effort to replicate the results of the paper that is the focus of this report, numerous challenges were encountered. The following is a discussion of these incidents.

The first challenge that I encountered related the problems with the convergence of the algorithms. For instance,in the case of the first experiment, choice of a small value for α was paramount for the convergence of the algorithm used in that case. Another challenge was lack of a common data set, to allow for an exact replication of the study. Due to lack of access to the data set used by the author, I randomly generated the data set in accordance with the description of the virtual environment. As such, in every instance that my results differed from that of the author's, I had no means of ruling out the difference in training data as the cause of those differences.

References

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