Temporal Difference Prediction

Scott Merrill
Georgia Institute of Technology
Atlanta, Georgia
smerrill7@gatech.edu
git hash: 921816476da34892224de7c8c54df2e47b4e3182

I. Introduction

Prediction applies to nearly every domain and helps determine the future behavior of an unknown system given the present state. While orthodox methods solve prediction problems by minimizing an error term between predicted and observed outcomes, *temporal-difference* (TD) methods minimize error between successive predictions. Prior to Sutton's 1988 paper on TD learning, the performance and statistical properties of TD methods remained unexplored [1]. Still, the technique showed promise; with impressive implementations in Samuel's checker player and Holland's Bucket Brigade a formal investigation into the new learning technique was certainly warranted. With empirical and theoretic evidence, Sutton proves not only the convergence of TD strategies, but also their optimality as compared to traditional supervised learning methods; they converge quicker, produce more accurate predictions and require less computational power. In this paper, we attempt to replicate Sutton's experiments and comment on the empirical accuracy and computational advantages of TD methods.

II. TD Learning VS Traditional Learning

A. Faster Convergence

Traditional methods to solve prediction problems take a supervised-learning approach whereby the learner is provided input-output pairs and is tasked with identifying a function that maps the input to the output. This approach works well in single-step prediction problems where all information required to determine the accuracy of a prediction is revealed in one time step. In multistep prediction problems, however, where partial information about the prediction accuracy is revealed overtime, supervised-learning approaches are slow learners; they don't learn until the final output is known. In contrast, TD methods learn through changes in predictions resulting in quicker convergence.

To demonstrate learning off changes in predictions, consider Sutton's example of a weatherman that on each day of the week, provides the probability of rain on the following Saturday. If on Monday the weatherman predicts a 50% chance of rain Saturday and then on Tuesday, he predicts a 75% chance of rain on Saturday, TD methods will learn based on this change of prediction. The supervised learning approach, however, will have to wait to observe the weather on Saturday before learning and error minimization can occur. Thus, supervised-learning strategies minimize in-sample error, whereas TD methods, as we'll see in the experiments, find the Maximum Likelihood Estimation (MLE) of the underlying Markovian Process. Another point to note is that the supervised learning approach completely ignores the structure of the problem and essentially casts a multi-step problem where information evolves over time into a single-step problem.

B. Computational Advantages

To demonstrate their memory and implementation advantages, the TD approach is compared to the Widrow-Hoff rule, a popular learning procedure often used in Artificial Neural Networks and Gradient Decent problems. Both approaches, yield the same outcome but the TD approach requires significantly less memory and therefore should be preferred.

Consider a series of input-output pairs $(x_1, x_2, .., x_n, z)$ where x_t represents a vector of observations available at time *t* and z represents the outcome. The learner will make predictions $P_1, P_2, \dots, P_m$ at each time step. For simplicity, consider a linear prediction function that's determined solely by a vector of modifiable weights, denoted w and the most recent set of observations x_t; thus $P_t(x_t, w) = w^T x_t$.

Being a supervised-learning approach, the Widrow-Hoff rule updates the weight vector by differences in the actual outcome and the predicted outcome at time t.

$$w = w + \sum_{t=1}^{m} \Delta w_t \qquad (1)$$

$$\Delta w_t = \alpha(z - P_t)\nabla_w P_t \qquad (2)$$

In (2), α is a set learning rate parameter and $\nabla_w P_t$ is the partial derivatve of the prediction with respect to the weight vector w. In our special case where $P_t = w^T x_t$, this equation simplifies to (3).

$$\Delta w_t = \alpha(z - w^T x_t)x_t \qquad (3)$$

The notable drawback of this method is the updates to w are dependent on z, which isn't determined until the end of the sequence. Predictions therefore must be carried forward and remembered to determine updates to w. TD methods evade this consequence by expressing the error in terms of the sum of prediction errors.

$$z - P_t = \sum_{k=t}^{m} (P_{k+1} - P_k) \qquad (4)$$

It can be shown that when error is expressed in this way the update rule to w becomes (5).

$$\Delta w_t = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \Delta_w P_k \qquad (5)$$

Since (5) only relies on successive predictions and the sum of all previous partial derivatives of our prediction with respect to w; TD methods can be implemented incrementally and observations need not be carried forward. Thus, for a sequence of length M, the TD procedure requires 1/M$^{\text{th}}$ of the memory as the Widrow-Hoff procedure. Additionally, learning can occur before the actual observation z occurs.

C. Class of TD methods

As discussed above, TD techniques differ from supervised learning techniques in that they learn based on successive predictions rather than the output z. The example above shows a special case where predictions at any time t update the weight vector the same. Generalizations of the TD procedure allow for the weight vector to be more sensitive to more recent predictions. Because it can also be implemented incrementally, Sutton considers exponential weighting of predictions as defined in (6).

$$\Delta w_t = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k \qquad (6)$$

It's important to note that when $\lambda = 1$, we have the special case of the Widrow-Hoff supervised-learning procedure; we will refer to this alternatively as the TD(1) procedure. When $\lambda < 1$, the result of $TD(\lambda)$ is different than TD(1). The greatest difference occurs when $\lambda = 0$ and weights are changed only by the most recent predictions. In this special case, the update rule becomes (7).

$$\Delta w_t = \alpha(P_{t+1} - P_t)\nabla_w P_t \qquad (7)$$

III. Random Walk Example

Sutton makes an important claim that any dynamic system that evolves overtime can be represented as a TD problem that is simpler and more efficient than supervised-learning methods. Sutton provides empirical evidence of these claims by implementing a very basic dynamic system; the bounded random walk. To investigate Sutton's claims, we attempt to replicate his experiment to confirm or refute his conclusions.

A. Problem Description

A bounded random walk is a state sequence produced by taking successive random steps until a terminal state is reached. Sutton's experiment considers a bounded random walk of 7 states labeled A through G. Each random walk starts at state D and moves either left or right with 50% probability. The sequence terminates when state A or G is reached. A visual representation of the problem can be seen in Figure 1.

.

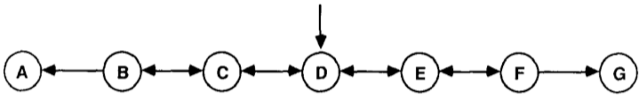

Figure 1

To cast the bounded random walk into a prediction problem, a reward of $z = 0$ is given when state A is reached and a reward of $z = 1$ is given when state G was reached. Both TD and supervised-learning approaches were used to determine the expected value of the reward in each state. With rewards defined in this way, the value of each state corresponds to the probability of the sequence ending in state G and can be shown to be $\frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}$ and

$\frac{5}{6}$ for states B, C, D, E and F respectively. With the true probabilities known, two experiments were conducted to test the performance of TD-learning and supervised-learning; the first tests the accuracy of both methods, while the second tests the speed of convergence.

B. Experiment Design

To test the desired properties of supervised-learning and TD methods, the observations from the bounded random walk had to be generated. As per Sutton's paper, 100 training sets each consisting of 10 random walk sequences was generated. The first experiment looked at seven different values of lambda; $\lambda = 0.0, 0.2, 0.4, 0.6, 0.8$ and 1.0. For each lambda, the TD(λ) procedure was used with a learning rate parameter of 0.02 to update the weight vector and identify the correct probabilities for each state. The weight vector was initialized to 0.5 for each state and each training set was repeatedly presented to the algorithm until convergence. We defined convergence as a Euclidean Distance of the change in weight vector that is less than 0.03. The changes made to the weight vector were accumulated and updated once at the conclusion of each training set. The converged weight vector was averaged over the 100 training sets and compared to the true probabilities of each state.

The second experiment contrasts the first in a few ways, but all is consistent with Sutton's implementation in his paper. Unlike the first experiment, each training set is only presented once, and weight updates weren't accumulated but instead adjusted after each sequence. As the second experiment tested the speed of convergence, multiple alpha values were chosen and compared to determine the learning rate which optimizes convergence speed. All alphas between 0 and 0.6 with increments of 0.05 were tested. And, similar to the first experiment, weights were initialized to 0.5 for each state.

C. Outcomes and Analysis

The results for the first experiment are shown in Figure 2. The x-axis shows the different values of λ tested and the y values show the Root Mean Squared Error (RMSE) between the asymptotic convergence and true probabilities. As shown in Figure 2, error was minimized for values of lambda between 0.3 and 0.5 and seemed to exponentially increase as lambda exceeded 0.6. The worst performing value of lambda by measure of accuracy occurred when lambda equaled 1 or equivalently the Widrow-Hoff procedure. The result is counter-intuitive but can be explained by the fact that the Widrow-Hoff procedure minimize the error on the training set not the underlying Markov process; TD(0) on the contrary identifies the MLE of the underlying Markovian Process and performed significantly better.

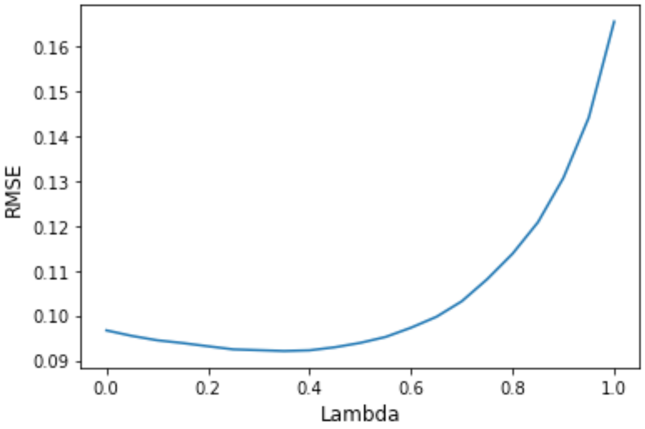

Figure 2

To highlight the difference between the TD(0) MLE approach and the TD(1) outcome-based approach consider how both algorithms would estimate the value of being in state C given only two random walk sequences:

1. $D, C, D, C, B, A, 0$
2. $D, E, F, G, 1$

For TD(1), the answer is simply 0; C has appeared two times and each time it resulted in a value of 0. The TD(0), MLE approach is more complex. In the two sequence, C was observed twice; once where it transitions to D and once where it transitions to B. Thus, the value of state C would be:

$$Value(C) = \frac{1}{2} * Value(B) + \frac{1}{2} * Value(D)$$

To solve this recursive problem, we need both the value of state B and the value of state D. By the same logic, the value of state D is given by.

$$Value(D) = \frac{2}{3} * Value(C) + \frac{1}{3} * Value(E)$$

The value of state B is simply 0 since the one time we observed state B it resulted in a reward of 0. And, similarly, the value of state E is 1. Simplifying, results in a system with two equations and two unknowns:

$$Value(C) = \frac{1}{2} * Value(D)$$

$$Value(D) = \frac{2}{3} * Value(C)$$

Solving this system of equations results in a value of state C of 0.25, which is much closer to its true value of

1/3. Moreover, minimizing in-sample error with TD(1) is a flawed approach that makes no assumptions on the underlying data generating process. TD(0), however, uses MLE to predict the underlying model for the random walk.

The results for the second experiment are shown in Figure 3. The x-axis shows different learning rates and the y-axis shows different errors. Four different series are labeled showing the relationship between error and learning rate for different values of lambda. As can be seen, the learning rate has a significant effect on each algorithms performance. The convex nature of the RMSE with respect to alpha is a theme also seen in gradient descent problems; setting the learning rate too low can drastically increase training time or cause the problem to get stuck at a local minimum whereas setting it too high results in an unstable algorithm prone to overshooting local and global minima. Regardless of the learning rate, however, TD methods seem to outperform the TD (1), Widrow-Hoff procedure. Figure 4 again plots lambda on the x-axis and RMSE on the y-axis, however, selects the learning rate for each lambda that minimizes the RMSE. Even with the best learning the supervised learning method results in the largest error.

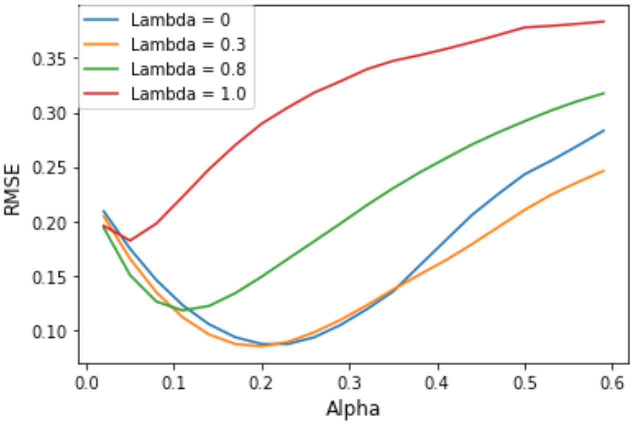

Figure 3

It's interesting to note that the optimal algorithm isn't TD(0) – the one that finds the MLE of the Markov Process – but instead occurs when lambda is 0.3. This demonstrates TD(0)'s property of slow propagation. As Sutton explains in his paper if the sequence $(x_d, x_e, x_f, 1)$ is experienced, TD(0) will only update the prediction for F. This contrasts the algorithms where λ > 0 which will also update the predictions for D and E. While perhaps a turnoff for situations with limited data and memory constraints, the slow propagation of TD(0) can easily be combatted with repeated presentations of sequences or backpropagation.

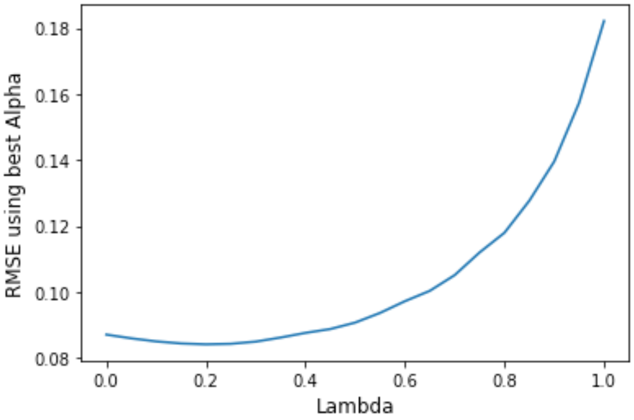

Figure 4

D. Comparison to Sutton's Resutls

Figure 5 shows Sutton's results obtained from his first experiment. The resulting figures look very similar in shape; however, Sutton's experiment indicates that RMSE is minimized with a lambda of around 0.3. While this differs slightly from our replication which indicates a lambda of 0.4 minimizes RMSE, such wasn't the point of the exercise. Sutton's larger conclusion that the TD(1) algorithm produces the largest error is clear in our results as well as Sutton's.

Nevertheless, this discrepancy may be explained by differences in training sets or by two main assumptions required to replicate the experiment. The first inference was our choice of alpha. Sutton doesn't specifically indicate a value used for the learning rate in the first experiment. Instead he suggests that the algorithm always converged for "small alpha." We noticed that for our training set, any alpha selected greater than 0.02 would result in a divergent solution. Thus, our experimentation used an alpha value of exactly 0.02.

Another source of difference was our convergence criteria. Sutton explains that convergence is achieved when there are "no longer significant changes in the weight vector." Thus, there is objectivity in not only the magnitude of such changes, but also the heuristic for identifying a change. We considered multiple heuristics and cutoff points to determine convergence and found that the Euclidian Distance Heuristic worked quite well. Convergence was determined when the distance of the changes in the weight vector was less than 0.03; this appeared to give the results most similar to Sutton's. Interestingly, we noticed, that the more times we iterate through the training set, the more the figure began to look like an exponential. We believe this is due to both the fact that TD(0) and TD(1) converge to different values and because an exponential factor was chosen to determine the weights for older predictions.

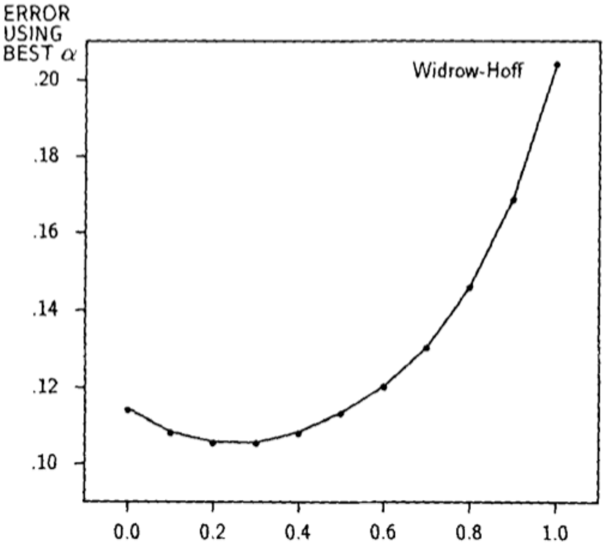

Figure 4

Our replication of the second experiment also showed very similar results to Sutton's. Figure 6 shows results from Sutton's experiment. Again, the general conclusion of the experiment remains unchanged; when λ < 1, the TD algorithms learn quicker than supervised learning techniques. Sutton's experiments are consistent with ours in that TD(0) and TD(0.3) provided the best results when considered at their optimal alpha values. However, the optimal alpha values per Sutton for TD(0) and TD(0.3) were around 0.3 whereas our experiments indicate this value to be around 0.2. In addition, the scale of the RMSE between our results and Sutton's is slightly off. Again, however, these discrepancies are minimal and don't affect the overall conclusion of the study. Furthermore, they are likely a result of the stochasticity of the training datasets.

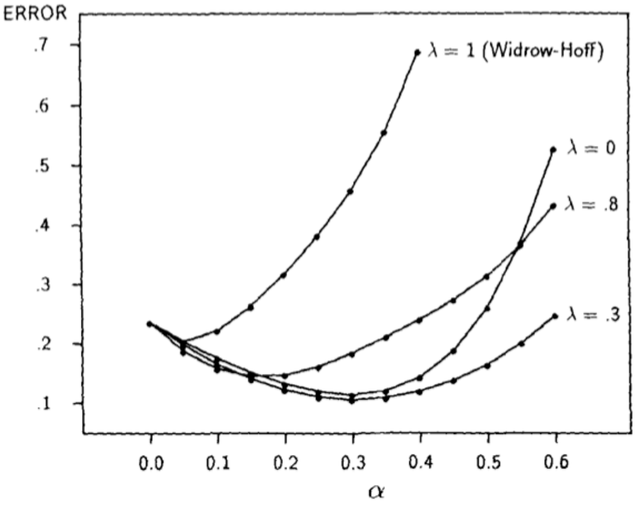

Figure 5

IV. Conclusion

Overall, Sutton's random walk experiment can be replicated easily with minimal assumptions and provides empirical proof of Sutton's claims about the convergence properties and optimality of TD methods. The simplicity of this experiment help reinforce the analytical proofs for the optimal convergence of TD(0) and suboptimal convergence properties of TD(1) Sutton provides in his paper. In addition, the results demonstrate the quicker learning properties of TD(0) and provide additional evidence for faster learning properties of generalized TD algorithms. Furthermore, with empirical proof of faster learning, empirical and analytical proof of the optimality of TD(0) under repeated presentations of data, the suboptimality of traditional supervised-learning procedures, and the intuitive incremental implementation of TD methods which saves memory, the choice between learning methods is self-fulfilling. In every aspect of learning, TD methods outperform the Widrow-Hoff rule and supervised-learning procedures and should be preferred.

References

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