Replication & Analysis of Sutton 1998

Evan Jones
Georgia Institute of Technology College of Computing
Git Hash:f84d95d163961cfc6bcf14e96c38ecf8cd2c4f6b
Georgia Institute of Technology
Atlanta, GA USA
evanjones@gatech.edu

***Abstract*—Despite its publication nearly thirty-three years ago, Sutton's 1988 paper "Learning to Predict by the Methods of Temporal Differences" remains one of the most foundational works on incremental learning procedures. In this paper Sutton formalizes the at the time cutting edge methods of credit assignment via differences in temporally successive predictions used by Samuel's checkers player [5] and Holland's bucket brigade [2]. In this article, Sutton's study is replicated and analyzed to better understand both Sutton's methods as well as the implications of these experimental results.**

***Index Terms*—TD Learning, Reinforcement Learning**

I. Introduction/Background

Prior to Sutton's 1988 paper "Learning to Predict by the Methods of Temporal Differences", most active research in learning problems centered on using supervised learning methods as a means of prediction. Although this approach can be very impactful, it has limited utility when applied to learning problems that involve a temporal or sequential dimension. Observing this need for a better method of "learning to predict" [4] rather than just predict Sutton proposes and formalizes the temporal difference learning paradigm. Under this learning approach, learning is accomplished incrementally through a framework of successive predictions and weight updates which overtime converge to the maximum likelihood estimate of the underlying Markov process. This method directly contrasts with the approach of supervised learning, which relies strictly on complete and definite sequences of states and rewards to form an estimate. Additionally, supervised learning methods remain optimal only with respect to training data and does not minimize error with respect to future data. This limits the class of problems that supervised learning can be applied to (without major problem relaxations), and is less efficient from both a computational and learning perspective. Through careful replication of Sutton's three random walk experiments this paper will demonstrate such differences between supervised and TD learning methods as well as their novel and surprising unification via Sutton's TD(λ) algorithm.

II. Random Walks

In order to best demonstrate and compare TD and supervised learning methods Sutton opts to utilize a bounded random walk example as this is one of the simplest dynamical systems to both implement, generalizes to more difficult problems easily, and yields an accessible format for debugging and observation. More specifically our random walk will consist of five non-terminal states B,C,D,E,F, two terminal states A,G with respective rewards zero and one. Each initialization of the random walk will begin in state D, and at each step in the environment the agent will move either one position right or one position left with a probability of $p = 0.5$. The agent will continue to take steps until the agent enters one of the terminal states A,G and receives the corresponding reward. After termination the environment is reset, and the agent begins again from state D. A graphical illustration of the process is provided in Sutton [4] and is reproduced in this paper as Figure 1. Each of the states visited during a given episode will be recorded with unit basis vectors of length five with a one representing the given state visited in that step. Given our choice of system we can clearly solve for the true transition probabilities for each state which later are used to benchmark the effectiveness of our various learning procedures. The true values of each state are a result of our reward ordering. Since left side termination has a zero reward, and we plan to utilize un-discounted state values via a constant $\gamma = 1$ the process naturally represents the probability of right side termination given the current state. These known values are as follows:

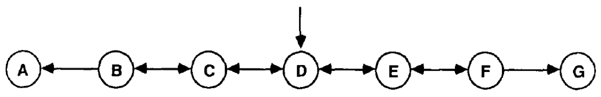


Figure 1. Graphical Representation of a Random Walk Process

$$True\ Values = \left[\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}\right] \qquad (1)$$

With the dynamics of our random walk clearly defined we now move onto the implementation of each learning procedure and sampling from this environment to conduct experiments on our learning procedures.

It is notable that the environment relies on Numpy's implementation of the Mersenne Twister pseudo-random number generation algorithm [3] to sample from the environment with respect to a Bernoulli ($p = 0.5$) distribution responsible for step direction. The Mersenne Twister algorithm was developed in 1997 which is nine years prior to the publication of Sutton's paper. For this reason it is impossible that Sutton utilized this same algorithm in his implementation of the experiment. Since Sutton does not specify a specific algorithm or seeds to initialize training sets I found it best to use Mersenne, the current standard of puesdo-random number generation. The differences in these methodologies and the nature of random sampling in general can result in some variance between our presented results. However, this should be kept to a minimum. This problem of random sampling differences will continue to prove and on going noise throughout experiments one, two and three.

III. TD-λ Implementation

The algorithm for computing TD(λ) was created in two primary forms in order to satisfy Sutton's specifications for the two different types of updates required of the algorithm in experiment one (batch updates over training sets) vs. experiments two and three (episodic updates over sequences). In order to verify the proper formulation of these algorithms I also constructed a standalone TD(0) and Windrow Hoff [7] implementation bereft of eligibility traces in order to verify that TD(λ) provided these same estimates for the cases of TD(0) and TD(1) as the coding of eligibility traces seem to be one of the more difficult aspects of the implementation. Implementation of both of these simpler algorithms also reinforced the equivalence between the Windrow Hoff [7] method in supervised learning and Monte-Carlo methods (for this markov process). In addition to the functions created for TD(λ), the replication of this paper also called for the construction of several helper functions for calculating RMSE and the generation of consistent training sets ready for use with our algorithms. Each of these functions is available for viewing in the paper's accompanying GitHub repository.

IV. Experiment 1

In Sutton's first experiment we will be sampling ten complete experience sequences or episodes from the random walk environment descried earlier. Each of these ten episodes constitutes a single training set. Given a training set, we apply Sutton's formulation of TD(λ) to the entire training set being careful to accumulate weight updates in the form of a cumulative weight update for all experience sequences within the episode, and without updating the randomly initialized state value array until the entire training set has been seen. After the first update to the state values we will continue to run the algorithm using the new updated weights array derived from the previous update. We will continue to use this approach of repeated presentations of the training set data until the changes in weights produced by successive iterations reaches a sufficiently small value so as to approximate convergence. After applying the TD(λ) algorithms to one hundred different episodes until convergence on each, we take each episodes final estimate of the state values and compare it to the true state values of the random walk expressed in (1) using residual mean squared error (RMSE) as our difference metric. Each resulting episode's metric was then averaged together with the other training set metrics to form and average estimate of RMSE for the given value of λ. This process was repeated using the same experience and training sets for each λ value by controlling random seed initialization.

In describing this methodology, Sutton leaves out some very important details that are crucial to exact replication. Firstly, Sutton neglects to specify an initial α value or the α decay schedule used within his formulation of the experiment. Although we are assured convergence given that α is "sufficiently small" [4] under both constant and decayed α due to repeated presentations (more on this in Section V) it remains to be seen what specifically Sutton had in mind. A similar lack of definiteness follows to the level of convergence before termination as well as the measure of convergence used. Typically, in practical experiments convergence in the limit would terminate in the event of the norm reaching an arbitrarily small value such as $1e-4$ (as this paper uses) or a limit on the number of iterations to avoid a non-halting problem. Through trial and error I found that the closest results to Sutton came from using the squared-euclidean norm for the convergence metric, a terminating weight delta condition of $\Delta_w \leq 1e-4$, and a terminating step count condition limiting each training set evaluation to one hundred thousand repeated presentations. However, it is notable that under no case did the algorithm implemented halt due to the repeated presentation count limit, which also demonstrates we properly converge in all cases.

With these problems in mind, the differences between Sutton's results on Figure 3 in [4] and the replicated results in Figure 2 could be due both to a difference in random sampling, a difference in convergence criterion, or even a differing definition of sufficiently small α value. However, despite these reasonable explanations it still seems odd that the graph retains the same shape as Sutton's results, and only truly differs by a small constant vertical shift upward. Despite several different implementations using various hyper-parameters, I was unable to thus directly reproduce the result so it remains an ongoing question as to how or if Sutton's implementation differs from the one set forth in this paper.

As we can see from the resulting chart in Figure 3, TD(0) performs the best among the other implementations. One of the reasons for is TD(0) is the most granular of all of the implementations above given its single step increments. This is in direct contrast to the Monte-Carlo or TD(1) approach which is limited to full episodes as observations. Thus, we can see plainly in the graph that as λ increases we lose the ability to learn from those more granular state transitions. Typically single step procedures such as TD(0) are limited in usefulness due to the far greater computation required compared to a single roll out like Monte-Carlo. However, this limited state space and implementation that allows large overhead for repeated presentations, TD(0) is able to take full advantage of the small learning increments and has plenty of time to properly propagate these incremental transitions throughout all states. It is also important to note the change in slopes of the Figure 2 as we move past $\lambda \geq 0.5$. After this

Figure 2. Replication of Sutton 1988 Experiment 1

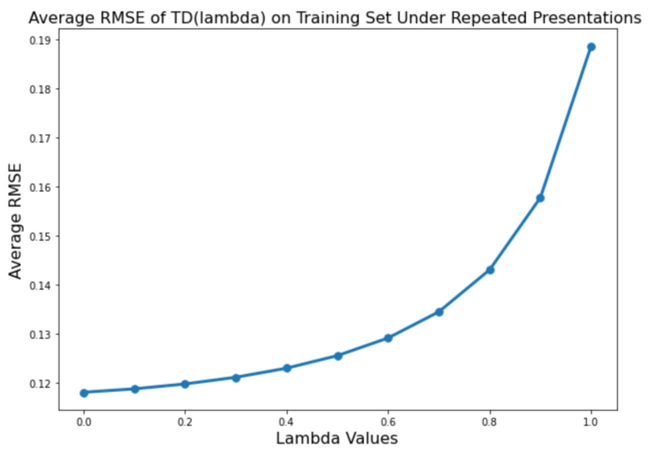
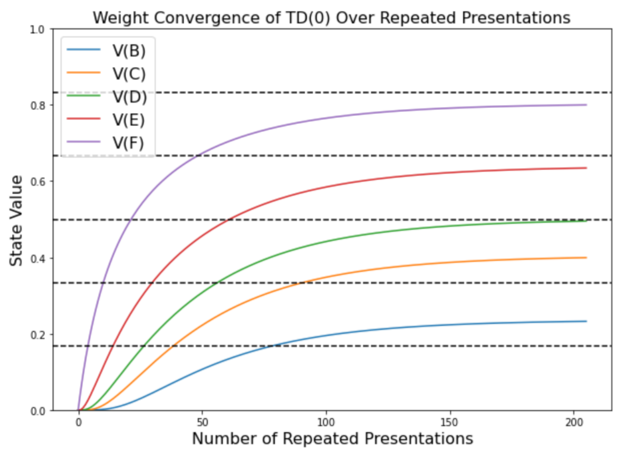


Figure 3. Convergence of TD Zero Weights Under Repeated Presentations



point the linear-like trend of increasing RSME seems to have actually turned out to be an exponential relationship. However, it is important to note that the range of the graph is also only 0.07 units so this behavior may seem worse that it really is due to the design of this particular experiment as well as the scale of the graph.

One of the more satisfying results of this experiment is that regardless of our initial weight vector and choice of α constant or using decay schedule we are guaranteed to converge to the same resulting state value estimates for a given training set as set forth by Sutton's proof of convergence [4]. The robustness of the implementation to the weight vector initialization makes sense as the specified update methods are agnostic to initial conditions. However, the α or more plainly worded step size parameter is especially important to convergence guarantees. It turns out that this result is nicely explained via an analogy to batch gradient descent and stochastic gradient descent. Updating in the method described in experiment 1 directly parallels to a batch update in the direction of the gradient. Due to repeated presentations paradigm the data we are trying to minimize with respect to the true labels remains stationary with respect to future targets. This allows utility updates to monotonically improve until we reach the optimal state value estimates for that given batch. This full convergence is best illustrated in Figure 3 where the single case of TD(0) is utilized to show how the weights change under repeated presentations.

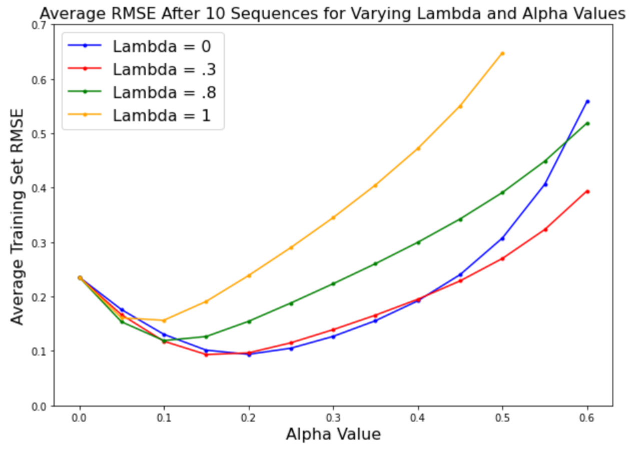
Similarly, the decayed α will accomplish this, but will likely take longer to converge given diminishing step sizes, which decay rapidly due to the additional exponent. However, this implementation with batch updates with repeated presentations is the only case we will discuss that permits both α decay and constant α convergence. As we will see in experiment two, sequential updates without repeated presentation parallel nicely to a stochastic gradient descent problem which must have decreasing α and immensely more observations to even potentially reasonably converge. Also notable is in Sutton's experiment RMSE estimates were observed to have a standard error of $\bar{SE} = 0.01$, this replication resulted in a slightly smaller standard error of $\bar{SE} = 0.0072$. The similarity in these metrics adds straight to the conclusion that our implementations were relatively similar and stable estimators for representing the true population.

V. Experiment 2

In experiment 2 Sutton begins to explore the impact of varying of α-values on different λ values and each of their impacts on the resulting RMSE. In this implementation we will not only vary the λ values for the evaluation on the training data, but will also vary the α step size parameter as well. Unlike before, we will take advantage of the incremental nature of TD(λ), and perform sequence level state value updates. Additionally, under this methodology we will also remove our convergence conditions and allow the algorithm to only see each training set of 10 episodes once each. Removing the benefits of repeated experience will allow us to better understand the speed and value of incremental learning. Unlike experiment 1 we will initialize the initial weight vector with equal weights. This is done due to the small number of episodes seen, and with lack of repeated presentations convergence is not guaranteed making initial conditions important to our prediction. In order to neutralize this effect we initialize the weights each to one half thus creating initial conditions that favor neither right or left side termination initially. Similar to experiment 1, Sutton neglects to specify a particular α decay schedule to use within the implementation of TD(λ) which is essential to replicate results properly. Building on our parallel to experiment two's update methodology and stochastic gradient descent we can put some bounds on possible decays schedules of α following convergence proofs by set forward by Shapiro and Wadi in [7] (For the SGD perspective) as well as Barto and Sutton [1] (For the TD perspective) which are as follows:

$$\sum_{n=1}^{\infty} \alpha_n = \infty; \quad \sum_{n=1}^{\infty} \alpha_n^2 < \infty \qquad (2)\ (3)$$

Figure 4. Replication of Sutton 1988 Experiment 2



With these conditions in mind and using trial and error I eventually decided on the following α decay schedule:

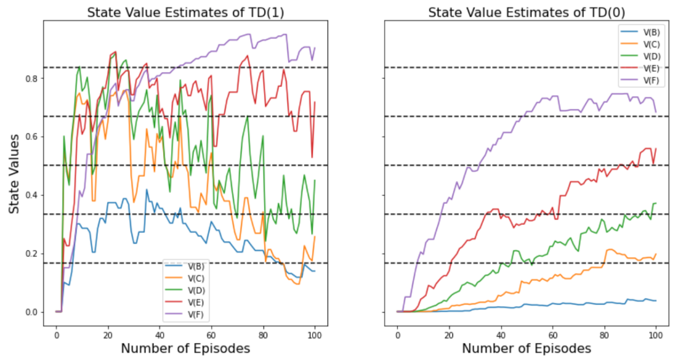
$$\alpha_{n_k} = \alpha_0/(|n_k|^{1/n_k}) \qquad (4)$$

Where $|n_k|$ = the length of the k^{th} episode and n_k = the index of the current episode being iterated. This α decay schedule led to the results displayed in Figure 4 which are nearly a mirror image of those produced by Sutton [4].

As we can plainly see from Figure 4 for small values of α it doesn't matter which value we initialize for λ we arrive at similar errors as all variants are pinned at an RMSE 0.25. With an α value this small there isn't time for the algorithm to take enough update steps large enough to have an impactful difference in RMSE. In fact, given our initial weight vector in this formulation with no updates our RMSE will be .2357 which is right around where each line is anchored to at very small α's. As α increases we can see that we start to see some benefit. The steps start to become large enough to update our weight vector enough within these ten episodes to lower the error. Notably, TD(1) is the first to see errors begin to increase again. This makes sense given our understanding of TD(1) use of full returns which already had high variance, but now due to large step size are doomed to perform poorly. TD methods with smaller values of lambda demonstrate more robustness to this problem as they create smoother estimates with more bias, but less variance.

This behavior is best illustrated by Figure 5 which plots weight updates on for each episode under both TD(1) and TD(0). It is plain to see that TD(1)'s estimates are extremely jagged and have high variance. On the other hand TD(0) shows a much less erratic progression of weight changes admittedly prone to sample bias, but far superior to that of TD(1). Together Figures 4 and 5 paint an interesting picture with regard to episodic updates and the impact of α. In Experiment 1 α was able to remain constant or non-decayed for the simple reason that we were working with the same batches of experience. This allows us to continually step in the correct direction as we have a constant target (our training

Figure 5. A Comparison of TD(0) and TD(1) Episodic Learning

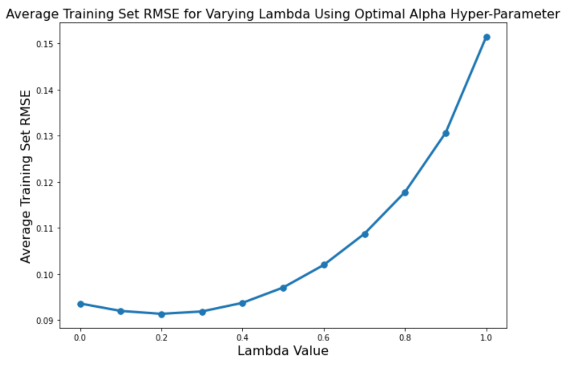


data) rather than these episodic updates which compare more to stochastic gradient descent as a means to overcome a non-stationary target of incoming new experience.

Another unexpected takeaway from the plot of experiment two is the observation that TD(0) is not always the method that results in the lowest RMSE at a given α. At first glance this seems to defy conventional wisdom that TD(0) would always be superior because it is the only method not limited by the granularity of n-step returns it can observe and account for all incremental state transitions. However, as we discussed in experiment 2 this is assuming TD(0) has time to propagate these early episode observations to later states. In experiment one TD(0) was given ample time to converge through repeated observation, in experiment two we only have 10 unique episodes presented once. This limited learning constraint thus requires us to make compromises between how far ahead we should look. If we stick with TD(0) we will not be able to fully propagate early transitions especially for small α which unfortunately is a requirement for implementing any TD(λ) method. To this end we may resort to slightly larger λ values such as $\lambda = .2$ or $\lambda = .3$ which are able to propagate experience more deeply without loss of generalization in the case of this particular experiment. This trade-off is the reason we can see some clear jockeying for the optimal λ value at a given α value.

However, overall TD(0) and TD(.3) remain quite close to one another which also makes sense given their similarity. As we would expect TD(1) performs the worst consistently with errors rapidly increasing for $\alpha > .2$. Of additional note in the plot is the interesting behavior of TD(0) as α increases from $\alpha > .4$. Initially at this level we see TD(0) provided worse RMSE values than TD(.3). As we continue to increase α TD(0) actually surpasses TD(.8) in RMSE. Again, this behavior by TD(0) seems inconsistent with our understanding of it typical performance. We can explain this divergence in expectation by considering the impact of larger step sizes. In TD(0) we use the most incremental of methods with the largest steps happening early on in the episode thus with larger α's we may introduce significant bias into our final estimates based upon this early experience. These larger steps are also applied

Figure 6. Replication of Sutton 1988 Experiment 3



more times in TD(0) than the other algorithms due to TD(0) being completely incremental in step thus this larger step in the wrong direction can be far more costly in terms of error for TD(0). Figures 4 and 5 provides an excellent example on the finer aspects to choosing a proper λ and α value pair as hyper-parameters in our learning model. Also, in the case of TD(1) universally poor performance it shows how TD methods are able to learn much faster from experience even in small sample size settings.

VI. Experiment 3

In experiment 3, we follow the same methods set forth in experiment 2 except this time we are looking to find the optimal α value for each given λ value that minimizes its RMSE. This type of hyper-parameter tuning will give us the best case scenario given a training set and λ value. Comparing such best case estimates allows us to compare the different initialization of λ's impact on the overall result. In implementing this experiment required certain practical reductions in the problem must be made in order to run in a proper amount of time. For this reason, the α values test were only in the range of [0,1] with a step size of one-tenth for a total of eleven α values. It is unclear based upon Sutton's plot and methodology if smaller more granular values of α were required to represent the result exactly. Similarly to previous experiments we also face similar lack of information regard the α decay procedure and the random seed initialization which can explain the small amount of variance in the reproduced graphs.

Moving in the analyzing the graph itself, we can see that we see a decreasing RMSE as we approach λ =.2. This again contradicts the conventional wisdom that TD(0) would always best in terms of RMSE performance with small α. However like experiment 2, given that this experiment centers on only using 10 episodes it may be that although at the limit TD(0) would be optimal and more accurately propagating the individual state transitions throughout the function under limited experience this ability is limited. On the other hand TD(.2) seems to be a comfortable level of comprise in the weighting of steps returns where over fitting is avoided, but reasonable convergence and experience prorogation is still confirmed. Beyond this point we see the RMSE steadily increases and reaches a worst case at λ =1 as expected from earlier analysis. Comparing the replicated results to Sutton's results demonstrates similar conclusions. Both the curve shape and slopes are very similar the only notable difference being Sutton's minimizing at $\lambda = .3$ instead of the replication observed minimizing value of $\lambda = .2$. The difference in these minimum values is most likely due to the inherently randomness of the problem. In fact Sutton concedes in his paper than most problems of this form will arrive at optimal λ between $0.0 \leq \lambda \leq 0.3$ a range which the replicated result surely resides within. It is notable that experiment 3 demonstrates just how important properly tuned λ parameters are to such a learning problem. we can see a nearly 80% reduction from best to worst RMSE observed λ controlling for changes in α by using the best case α the effect of λ alone is isolated and clearly visible.

VII. Concluding Thoughts

Although tedious and at times frustrating attempting to understand the original authors methods and intent, study replication such as this provides a great exercise to truly understand the finer points of an academic paper. The process of re-reading, trial, and error (much like TD learning itself) permits incremental learning and after working it fully it finally clicks (or I suppose converges). Posing answers to the questions Sutton left for the reader on α values and convergence were directly responsible for my discovery of the parallels to gradient descent methods, step sizes, and update methods which is an invaluable insight as one progresses further into the field and builds upon these valuable intuitions. For this reason I find Sutton's experimental design to be rather ingenious. They each successively prove additional properties of the TD(λ) on the surface while also building a sequentially stronger foundation in the intuition behind TD(λ), and by experiment three Sutton has made the algorithm seem both novel and an obvious improvement upon supervised learning in such temporal problems.

References

[1] Andrew G. Barto and Richard S. Sutton ”Reinforcement Learning: An Introduction” Second Edition, The MIT Press, 2020.

[2] Holland J. H. ”Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In: Machine learning: An artificial intelligence approach (Vol. 2). Los Altos, CA: Morgan Kaufmann, 1986

[3] Matsumoto M. Nishimura T. ”Mersenne Twister: A 623-Dimensional Equidistributed Uniform Pseudo-Random Number Generator” In: ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1. pgs. 3-30, January 1998.

[4] Richard Sutton “Learning to Predict by the Methods of Temporal Differences”. In: Machine Learning 3 (Aug. 1988),pp.9-44

[5] Samuel, A.L. ”Some studies in machine learning using the game of checkers”. In: IBM Journal on Research and Development, 3, 210-229 New York, NY: Mcgraw-Hill 1959.

[6] Shapiro A., Wardi Y. ”Convergence Analysis of Gradient Descent Stochastic Algorithims” In: Journal of Optimization Theory and Applications: Vol 91, No. 2 pp. 439-454. New York, NY: Plenum Publishing Corporation, 1996.

[7] Widrow B.. and Hoff. M.E. ”Adaptive switching circuts”. In: 1960 WESCON Convention Record, Part IV pp. 96-104, Englewood Cliffs,NJ: Prentice-Hall,1960.