CS7642 - Project 1: Desperately Seeking Sutton
Git SHA1: 8b0081d2a4588d3f639f1e5a88b8147f7ac1ecf2

Ceslee Montgomery
cmontgomery38@gatech.edu

Abstract

Sutton introduces a set of learning procedures, known as temporal-difference (TD) methods, that make several improvements over existing supervised learning methods. Principally among them: (1) TD methods are more incremental and therefore easier to compute and (2) by more efficient use of experience they converge faster and produce better predictions. These claims are investigated in more detail in this paper.

1 Introduction

1.1 TD Learning

Prediction learning, or predicting future behavior from past experience, is one of the most ubiquitous types of learning applications. For example, one might learn to predict tomorrow's weather based on today's weather. In conventional outcomes-based methods, learning doesn't happen until the moment an outcome is observed, and is therefore, driven by the error between predicted and actual outcomes.

Crucially, however, under a temporal-difference framework, learning is driven by *temporally successive predictions*. For example, while conventional methods might predict tomorrow's weather only once, by employing TD methods one might predict the weather every hour between today and tomorrow. In this way, TD methods take advantage of intermediary predictions (the hourly forecast) and their subsequent correctness, and thus learn prior to knowing the final outcome. Critically, TD methods assume problems are *multi-step*, meaning the subject of prediction is multiple steps in the future, with information possibly revealed on each step. The TD family of methods include an eligibility trace parameter, λ, which allows credit assignment to decay with recency.

1.2 The Random Walk Problem

To illustrate the advantage that TD methods have over conventional supervised methods, Sutton states that the system have one required characteristic: it must be a dynamical system, such that its state evolve over time. The simplest among these, the **Random Walk Problem** presented by Sutton describes a problem modeled as a Markov Decision Process (MDP) where the transition probabilities and rewards are known. This allows for a simulation in which various methods' performance can be evaluated against known model dynamics. The problem is described as follows: a *bounded random walk* is a state sequence generated by taking random steps to the left or right until a terminal state is reached. Sutton's random walk, illustrated in Figure 1, contains 7 states, with the center state being the initial state and the left and right boundary states being terminal states. All non-terminal states have a equal probability of moving to the left or right. In order to formulate this model as a prediction problem, the outcome was defined in the right-side termination case, state G in Figure 1, to be z=1 and a left-side termination, and state A in Figure 1, z=0. Each learning method estimates the expectation of z, which, as stated, is equal to a right-side termination.

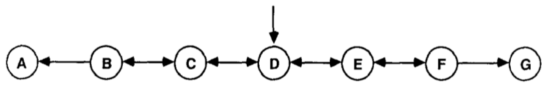

Figure 1: Sutton's MDP generator of bounded random walks

2 Experiments

2.1 Experiment 1 - Performance of TD Methods vs. Classical Supervised Learning Procedure (Widrow-Hoff) on Random Walk Problem

In order to evaluate comparative performance of TD methods, Sutton introduces a theorem that shows the linear TD(1) procedure produces the same per-sequence weight changes as the Widrow-Hoff procedure (classical supervised-learning procedure). Enabled by this theorem, the first experiment compares the performance error of various values of λ: $[0, 0.1, 0.3, 0.5, 0.7]$, against the Widrow-Hoff procedure (=TD(1)).

2.1.1 Implementation

First, 100 training datasets of 10 episodes were generated according to the aforementioned Random Walk. Then, a TD(λ) algorithm was implemented with two specific conditions: (1) weight updates were accumulated over each training set (10 episodes) and then applied and (2) each training set is repeated until convergence. By repeating the training sets until convergence – or weight updates become insignificant – the procedures are guaranteed to arrive at their best performance.

The TD(λ) algorithm was implemented as follows:

Algorithm 1 TD(λ) - w/per training set weight updates + repeated presentation

```
Fix $\epsilon$ (e.g. $\epsilon = 0.001$)
Fix $\alpha$ (e.g. $\alpha = 0.01$)
Initialize $s$
Initialize feature vector $x(s)$ (such that $x(terminal) = 0$) for all $s$
Initialize $w(s) = 0$ for all $s$
repeat
  Initialize $\Delta w(s) = 0$ for all $s$
  for all episodes in dataset do
    Initialize $E(s) = 0$ for all $s$
    for all steps in episode do
      observe $x'$, $R$
      $E \leftarrow E * \lambda + x$
      if $x' = 0$ (signaling terminal state) then
        $z = R$
        $\delta \leftarrow z - w^T x$
      else
        $\delta \leftarrow w^T x' - w^T x$
      end if
      $\Delta w \leftarrow \Delta w + \alpha * \delta * E$
    end for
  end for
  $w \leftarrow w + \Delta w$
until $\|\Delta w\|_{L_\infty} < \epsilon$
```

To perform this experiment, the above algorithm was applied for each value of λ. To reduce the statistical noise of any specific dataset, the algorithm was run over 100 such training datasets. To evaluate performance, the Root Mean Squared Error (RMSE) shown in (1) between the predictions and the "ideal" predictions (described in Sutton) was measured and averaged over the training sets.

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2} \tag{1}$$

As suggested above, a few hyperparameters must be specified. In running these experiments, $\alpha = 0.01$ and $\epsilon = 0.001$ were found to balance performance and fast computation. This experiment runs in $< 30s$ with a Python 3 (+ Numpy) setup.

2.2 Outcome

The process of implementing the code to run the experiment was relatively straightforward as the algorithm and paramaters (except the *exact* hyperparameters) were provided. The results of this experiment, shown in 2, track very similarly to [Sutton]'s. The errors from this experiment show systematically lower errors (.08 less). Runs with multiple random seeds showed that there is some variance in the error, so this isn't unreasonable. To prevent this problem in the future, with today's computational power this experiment could easily be run with 10^4x bigger datasets.

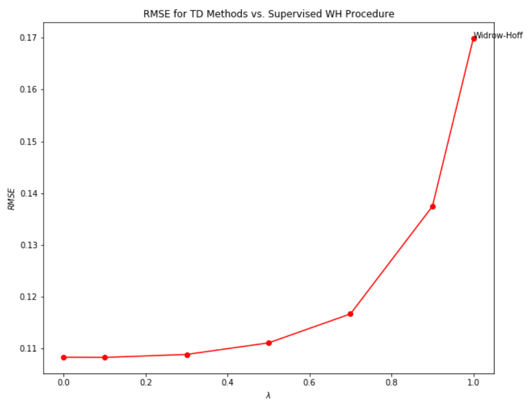

Figure 2: TD methods show better performance than the classical supervised WH procedure (see [Sutton] Figure 3)

Negligible differences between the figures aside, the outcome of this experiment support [Sutton]'s claim that TD method achieves superior performance to the classical supervised learning, Widrow-Hoff procedure. As [Sutton] mentioned previously, this points to the benefit of bootstrapping-based methods, where prior estimates and intermediary outcomes are used to improve new predictions. The next experiment examines [Sutton]'s claims that TD is not only better performing, but more compuationally efficient.

2.3 Experiment 2 - Computational Efficiency of TD Methods vs. Classical Supervised Learning Procedure (Widrow-Hoff) on Random Walk Problem

The previous experiment re-confirms Sutton's claim that TD methods can perform better than classical prediction methods – but at what cost? Understandably, Sutton follows up with an experiment into the computational complexity and memory efficiency of these methods. Thus, this second experiment compares the performance error of various values of λ, against the Widrow-Hoff procedure (=TD(1)) at various α levels with only a single presentation of each dataset. Therefore, holding computation relatively constant, allowing a direct comparison between the methods given a single opportunity to learn.

2.3.1 Implementation

First, the 100 training datasets of 10 episodes from the prior experiment were re-used for consistency. Then, a TD(λ) algorithm was implemented with two specific conditions: (1) weights are updated per sequence and (2) each training set is repeated only once.

The TD(λ) algorithm was implemented as follows:

Algorithm 2 TD(λ) - w/per sequence weight updates + single presentation

```
Fix $\epsilon$ (e.g. $\epsilon = 0.001$)
Fix $\alpha$ (e.g. $\alpha = 0.01$)
Initialize $s$
Initialize feature vector $x(s)$ (such that $x(terminal) = 0$) for all $s$
Initialize $w(s) = 0.5$ for all $s$
**repeat**
  Initialize $\Delta w(s) = 0$ for all $s$
  Initialize $E(s) = 0$ for all $s$
  **for all** episodes in dataset **do**
    **for all** steps in episode **do**
      observe $x'$, $R$
      $E \leftarrow E * \lambda + x$
      **if** $x' = 0$ (signaling terminal state) **then**
        $z = R$
        $\delta \leftarrow z - w^T x$
      **else**
        $\delta \leftarrow w^T x' - w^T x$
      **end if**
      $\Delta w \leftarrow \Delta w + \alpha * \delta * E$
    **end for**
    $w \leftarrow w + \Delta w$
  **end for**
**until** $\|\Delta w\|_{L_\infty} < \epsilon$
```

To perform this experiment, the above algorithm was applied for each value of λ used in Sutton and run over 100 such training datasets. To evaluate performance, again the RMSE (1) was averaged over the training sets.

This experiment runs in $< 30s$ with a Python 3 (+ Numpy) setup.

2.4 Outcome

Figure 3 captures the breadth of possible $\lambda * \alpha$ combinations. It's clear that λ=1 (the WH procedure) is again worse performing for every value of α. This finding suggests that even for a single pass over the data (such that computation time is somewhat equalized among methods), TD methods still largely outperform the WH procedure. Figure 4 brings this outcome more clearly into view as the best α are chosen to clearly indicate the results are unambigious.

This reproduction of Sutton's experiments align even more closely with the original. In this case, the RMSE errors are within .1 for Figure 3 and .05 for Figure 4.

References

Richard S. Sutton. Learning to predict by the methods of temporal differences. 3(1):9–44. ISSN 0885-6125. doi: 10.1023/A:1022633531479. URL `https://doi.org/10.1023/A:1022633531479`.

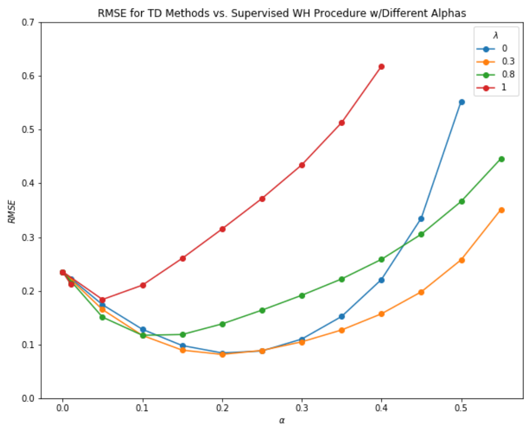

Figure 3: TD methods show better performance than the classical supervised WH procedure even with learning rate and iterations held constant (at 1 pass) (see [Sutton] Figure 4)

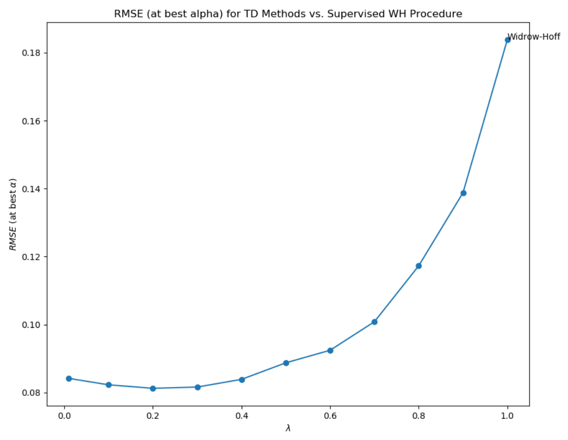

Figure 4: Combined graph showing all λ at best α (see [Sutton] Figure 5)