Replicating experiments in Sutton (1988) - Learning to Predict by the Methods of Temporal Differences

SAHIL DHINGRA

CS-7642 Reinforcement Learning
Project 1
GitHub hash – *8be86c4*

1. Introduction

In this article, we would discuss replication of the following random walk sequence from Sutton (1988). A description of the walk can be found in Figure 2 below. A walk's outcome is defined to be z = 0 for a walk ending on the left at A and z = 1 for a walk ending on the right at G. The learning methods estimate the expected value of z; for this choice of z, its expected value is equal to the probability of a right-side termination. At any time t, the state of the system can be represented as a one-hot vector. For example, D would be represented as [0, 0, 0, 1, 0, 0, 0] and so on. P_t can be calculated as $w^T x_t$.

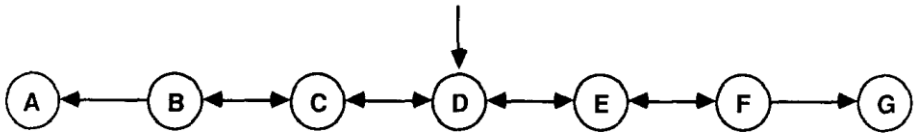


Figure 2. A generator of bounded random walks. This Markov process generated the data sequences in the example. All walks begin in state D. From states B, C, D, E, and F, the walk has a 50% chance of moving either to the right or to the left. If either edge state, A or G, is entered, then the walk terminates.

2. Experimental Design

Our example generates *bounded random walks*. Two computational experiments were performed using observation-outcome sequences generated as described above. In order to obtain statistically reliable results, 100 training sets, each consisting of 10 sequences, were constructed for use by all learning procedures. For all procedures, weight increments were computed according to TD(λ) algorithm. Different values were used for λ. These were λ = 1, resulting in the Widrow-Hoff supervised-learning procedure, λ = 0, resulting in linear TD(0), and also λ = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 and 0.9, resulting in a range of intermediate TD procedures.

2.1. Experiment 1

Referred to as *repeated presentation* training paradigm, in this experiment, for each training set, Δw's are computed and summed over all 10 sequences in the set. After the complete presentation of a training set, the weight vector is updated and Δw's reset to 0. This procedure is repeated on the same training set until convergence. A low enough α is selected (α = 0.01). Beyond a certain value of α (~0.03), the procedure never converges.

2.2. Experiment 2

The objective of this experiment is to assess how fast each procedure learns, for a given λ. Unlike in Experiment 1, we don't run this procedure until convergence. Rather, final weight vector is obtained after passing each training set only once through the procedure. It differs from Experiment 1 in a few other ways as well –

I. Weight update is performed after each sequence, rather than accumulating Δw's over 10 sequences in the training set
II. We test a range of learning-rate parameter α values: 0.01, and values in the interval [0.05, 0.6] with a difference of 0.05
III. To avoid any bias either toward termination at A or G, since we are running the procedure only once per sequence, all components of the weight vector were initially set to 0.5

3. Implementation

3.1. Algorithm

TD(λ) was implemented in Python 3.7 for both the experiments, as described below.

1. **Experiment 1 –**

Initialize $\alpha = 0.01$
Loop for each training set:
Initialize weight vector $w = [0, 0, 0, 0, 0, 0, 0]$ and $\Delta w = [1, 1, 1, 1, 1, 1, 1]$
If maximum absolute value in $\Delta w < 0.0001$ (convergence condition):
$\Delta w = [0, 0, 0, 0, 0, 0, 0]$
Loop for each sequence:
Initialize $e_t = x_1$
Loop for sequences = 1, 2….T–1:
$\Delta w = \Delta w + \alpha\ (w^T x_{t+1} - w^T x_t)\ e_t$
$e_{t+1} = x_{t+1} + \lambda\ e_t$
$\Delta w = \Delta w + \alpha\ (z - w^T x_T)\ e_t$
$w = w + \Delta w$

2. **Experiment 2 –**

Loop for each λ:
Loop for each α:
Loop for each training set:
Initialize weight vector w = [0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5]
Loop for sequence = 1, 2…10:
Δw = [0, 0, 0, 0, 0, 0, 0]
Initialize $e_t = x_1$
Loop for sequences = 1, 2….T–1:
$\Delta w = \Delta w + \alpha (w^T x_{t+1} - w^T x_t) e_t$
$e_{t+1} = x_{t+1} + \lambda e_t$
$\Delta w = \Delta w + \alpha (z - w^T x_T) e_t$
$w = w + \Delta w$

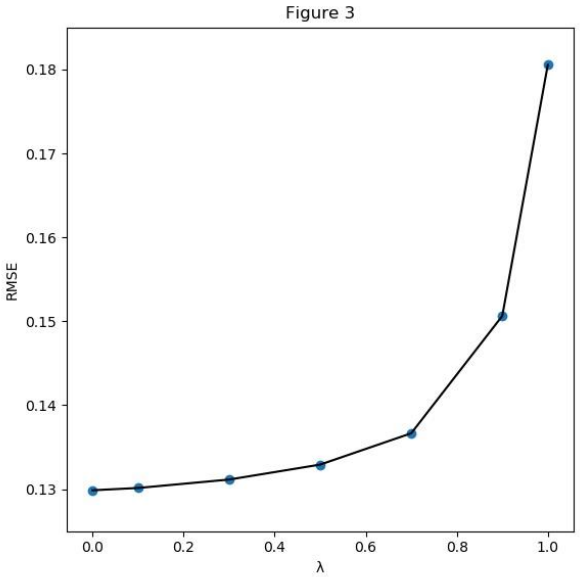
3.2. RMSE calculation

$$RMSE = \sqrt{\frac{(w_B - 1/6)^2 + (w_C - 1/3)^2 + (w_D - 1/2)^2 + (w_E - 2/3)^2 + (w_F - 5/6)^2}{5}}$$

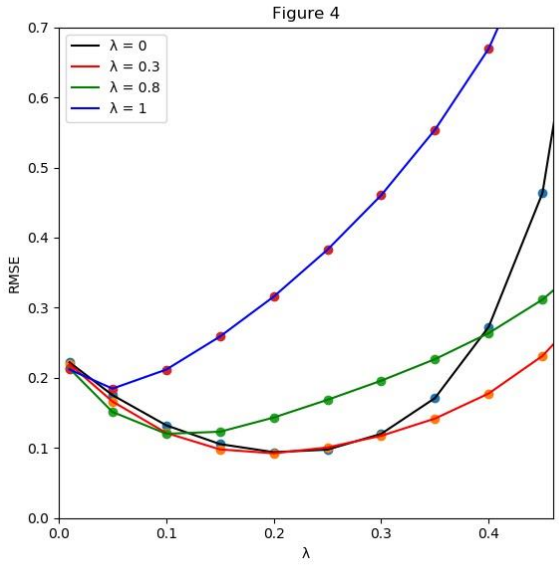
4. Results

For a small enough α (0.01 used in this experiment), the Fig.3 is replicated with an almost identical shape. However, the RMSE values are smaller for all λs.

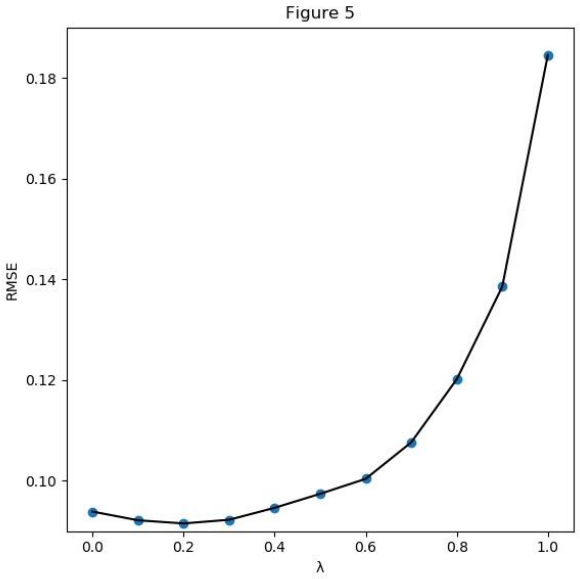
1. **Fig.3** – For a small enough α (0.01 used in this experiment), the Fig.3 is replicated with an almost identical shape. However, the RMSE values are smaller for all λs



2. **Fig. 4** – Replicated results appear identical to Sutton's version when we look at a slightly cropped version of the plot, with α values cropped until 0.5. RMSEs for all λs explode rapidly beyond this point. We will discuss the reasons for this difference in the next section.



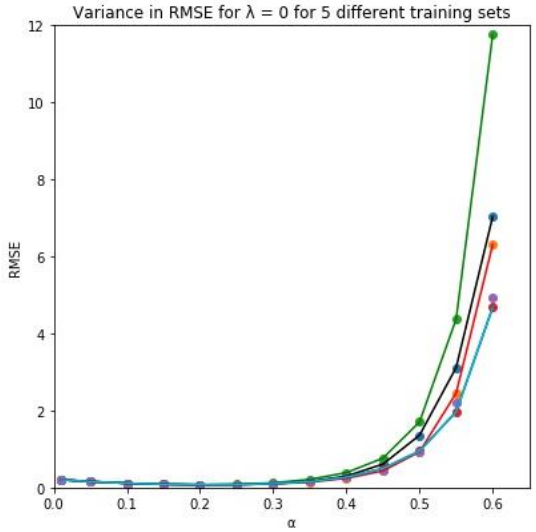
3. **Fig. 5**, the curve is not perfectly exponential, unlike the paper – slight deviation from the perfect curve can be observed for a few instances.
    a. The minima in the curve can be observed around the λ value, ~0.3
    b. However, the RMSE values are lower for each λ compared to the paper



5. Analysis of results

5.1. **Weight initialization** – In Experiment 1, the final value of the weight vector is not independent of the initial values, contrary to what is mentioned in the paper. Initialization does seem to affect the final weight values significantly and similar results can be produced with different initial weights.

5.2. **Variance in results for higher α values** – In Experiment 2**,** for all λ values, the variance in results increases exponentially with α. For higher α values, the procedure makes bigger jumps in weight updates after each sequence**.** Results are more sensitive to the training set, which explains why the replicated Fig.4 appears different for higher α values. This was validated by reproducing the figure with different training sets, which resulted in changes in the curves.



5.3. **Small α** – In Experiment 2, the corresponding small enough α was not provided in the paper. Upon randomly selecting a small value of α = (0.1, 0.2, 0.3..), the algorithm doesn't converge. Hence, even lower values were tested. Final value used was α = 0.01. For α>0.03, the solution doesn't converge.

5.4. **Convergence condition** – Further, the exact convergence condition was not mentioned. Higher values and other convergence conditions were also tested, but weights don't converge to the solution in some of those cases.

5.5. **Bias towards training data** – RMSEs for all λ values explode rapidly for α>0.5 (which cannot be verified against the plot in the paper which is limited to RMSE values lower than 0.7). For higher α values, the curves are more sensitive to the training data and result in significantly different weights for different training sets. Change in weights after complete presentation of a training set are large enough for α greater than a certain value, such that the weights would never converge.

References

Richard S. Sutton (1988), Learning to Predict by the Methods of Temporal Differences. *Machine Learning 3:9 44, 1988*