

Learning to Predict by the Methods of Temporal Differences

COMP 767 Presentation

Charles C Onu

260663256

17 March, 2017

Objective

- ▶ To mathematically illustrate the transference of TD learning (and RL methods, in general) to the supervised learning context
- ▶ To show empirical results on optimality of TD for sequence learning tasks

Primary reference

R. S. Sutton, "Learning to Predict by the Method of Temporal Differences" in Machine Learning 3, Boston:Kluwer Academic Publishers, 1988.

Supervised Learning

- ▶ Single instance prediction; e.g., image and digit present

$$x^1, y^1$$

$$x^2, y^2$$

...

$$x^n, y^n$$

- ▶ Time-series/sequence-based prediction; e.g., chess positions and outcome; heart-rate readings and clinical outcome

$$x_1^1, x_2^1, \dots, x_m^1, y^1$$

$$x_1^2, x_2^2, \dots, x_m^2, y^2$$

...

$$x_1^n, x_2^n, \dots, x_m^n, y^n$$

where $x_i, x_i^j \in R^k$

Relation to Model-free RL

Sequence prediction is similar to the model-free reinforcement learning setting

- ▶ Similarity
 - ▶ Dynamics of underlying system is not known
 - ▶ We have actual example trajectories x_1, x_2, \dots, x_m , and the final outcome y
- ▶ Difference
 - ▶ In RL: trying to figure how best to act in the system to accumulate 'good' outcomes (active learning)
 - ▶ In Supervised Learning: Given a trajectory already taken, what is the final outcome/reward - win or lose? rain or sun, survive or die

Gradient Descent

- ▶ Gradient descent update for some set of sequence data $x_1, x_2, x_3, \dots, x_m, y$ is given as:

$$w := w + \sum_{t=1}^m \Delta w_t$$

$$w := w + \sum_{t=1}^m \alpha(y - h(x_t)) \nabla_w h(x_t)$$

- ▶ Widrow-Hoff rule (Widrow & Hoff, 1960): special case of gradient descent where the hypothesis function is linear.

$$w := w + \sum_{t=1}^m \alpha(y - w^T x_t) x_t$$

- ▶ $h(x_t)$ can be non-linear (or any function indeed). $\nabla_w h(x_t)$ just has to be computable.

Deriving TD for sequence prediction(1)

- ▶ Key idea is to represent the error term as a sum of changes in successive predictions

$$y - h(x_t) = \sum_{k=t}^m (h(x_{k+1}) - h(x_k)) \quad \text{where } h(x_{m+1}) = y$$

both produce exactly same weight changes

- ▶ Putting back into gradient descent:

$$w := w + \sum_{t=1}^m \alpha (y - h(x_t)) \nabla_w h(x_t)$$

$$w := w + \sum_{t=1}^m \alpha \sum_{k=t}^m (h(x_{k+1}) - h(x_k)) \nabla_w h(x_t)$$

$$w := w + \sum_{t=1}^m \alpha (h(x_{t+1}) - h(x_t)) \sum_{k=1}^t \nabla_w h(x_k)$$

Deriving TD for sequence prediction(2)

- ▶ The expression is nice:
 1. It is incremental/online and memory efficient. Don't have to keep each example in memory until end of sequence.
 2. Makes more efficient use of data (learns faster)
- ▶ Recall that these are exactly the benefits of TD over Monte Carlo method.

$$w := w + \sum_{t=1}^m \alpha(h(x_{t+1}) - h(x_t)) \sum_{k=1}^t \nabla_w h(x_k) \quad \text{TD}(1)$$

- ▶ As mentioned earlier, this learning rule produces the same weight changes, and prediction as the standard supervised learning form.
- ▶ This is TD(1) for the sequence prediction problem

Deriving TD(λ) for sequence prediction(1)

- ▶ Key idea: include exponential weighting with recency (where $0 \leq \lambda \leq 1$)

$$w := w + \sum_{t=1}^m \alpha (h(x_{t+1}) - h(x_t)) \sum_{k=1}^t \lambda^{t-k} \nabla_w h(x_k) \quad \text{TD}(\lambda)$$

- ▶ Note that last sum term can still be computed incrementally

$$e_t = \sum_{k=1}^t \lambda^{t-k} \nabla_w h(x_k)$$

$$e_t = \nabla_w h(x_t) + \sum_{k=1}^{t-1} \lambda^{t-k} \nabla_w h(x_k)$$

$$e_t = \nabla_w h(x_t) + \lambda e_{t-1}$$

- ▶ λ effectively controls how far back into the past one should go to make updates.

Deriving TD(λ) for sequence prediction(2)

- ▶ When $\lambda = 0$,

$$e_t = \nabla_w h(x_t)$$

- ▶ e_t helps us see that for TD(0), the weight increment is influenced only by its effect on the prediction associated with the most recent observation

$$w := w + \sum_{t=1}^m \alpha(h(x_{k+1}) - h(x_k)) \nabla_w h(x_t) \quad \text{TD}(0)$$

- ▶ As $\lambda \rightarrow 1$, previous observation predictions are considered with increasing exponential weighting.

Experiment: $TD(\lambda)$ in random walk

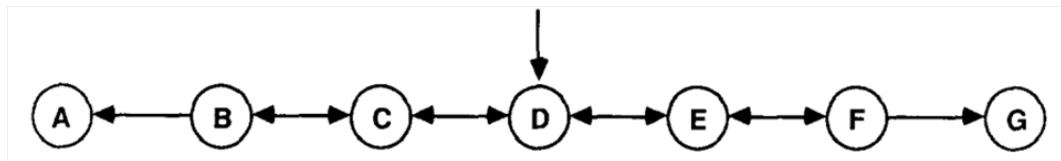


Figure 1: Random Walk Environment(Sutton, 1988)

- ▶ Always starts at D
- ▶ Policy: equiprobable random actions - left or right
- ▶ Rewards: +1 if terminate on right. 0 if terminate on left.

Experiment

- ▶ Training set of 10 sequences(or examples) was generated 100 times. E.g., [D E D E F G]
- ▶ Each visited state in a sequence was represented as one-hot encoded vector E.g., if state A, $x_i = [1 \ 0 \ 0 \ 0 \ 0]$; if B $x_i = [0 \ 1 \ 0 \ 0 \ 0]$, etc.
- ▶ So that prediction $h(x_i)$ is essentially the learned weight, \hat{w}_i :

$$h(x_i) = \hat{w}^T x_i = \hat{w}_i$$

- ▶ True values of weights w for non-terminal states B, C, D, E, F are $\frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}$ respectively
- ▶ Root mean square error (rmse) between true weight w and learned weight, \hat{w} is computed.
- ▶ Tested using $\lambda = \{0.1, 0.3, 0.5, 0.7, 1\}$ and $\alpha = [0 : 0.05 : 0.6]$.

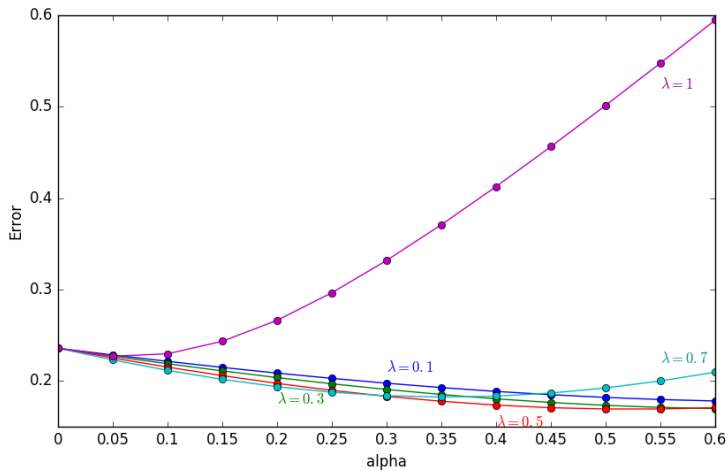


Figure 2: Average error over 100 runs

Summary

TD learning applied to the sequence prediction problem

- ▶ converges more rapidly
- ▶ makes more efficient use of experience
- ▶ makes more accurate predictions along the way
- ▶ is a lower variance method

than standard supervised learning (i.i.d) approach.

NOTE: These convergence and properties above are only guaranteed if the underlying system is Markov. Many natural and dynamical systems are.