True Online Emphatic TD(λ): Quick Reference and Implementation Guide

Richard S. Sutton

Revised June 6, 2015

TD(λ) is the core temporal-difference algorithm for learning general state-value functions (Sutton 1988, Singh & Sutton 1996). True online TD(λ) is an improved version incorporating dutch traces (van Seijen & Sutton 2014, van Seijen, Mahmood, Pilarski & Sutton 2015). Emphatic TD(λ) is another variant that includes an "emphasis algorithm" that makes it sound for off-policy learning (Sutton, Mahmood & White 2015, Yu 2015). This document presents the implementation of *true online emphatic TD(λ)*, an algorithm that combines the true-online idea and the emphatic idea, patterned after the combination, by van Hasselt, Mahmood, and Sutton (2014), of the true-online idea and the gradient-TD idea (Maei 2011, Sutton et al. 2009).

1 Setting and requirements

We consider the setting of general value functions, or GVFs (Maei & Sutton 2010, Sutton et al. 2011, White 2015, Sutton, Mahmood & White 2015). Here we present these ideas without assuming access to an underlying state (as in Modayil, White & Sutton 2014).

The algorithm is meant to be called at regular intervals with data from a time series, from which it learns to make a prediction. The time series includes a feature vector $\boldsymbol{\phi}_t \in \mathbb{R}^n$ and a cumulant signal $R_t \in \mathbb{R}$. The prediction at each time is linear in the feature vector. That is, the prediction is of the form

$$\boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t = \sum_i \phi_t(i)\theta_t(i)$$

where $\boldsymbol{\theta}_t \in \mathbb{R}^n$ is the learned weight vector at time t, and $\phi_t(i)$ and $\theta_t(i)$ are of course the ith components of the two vectors. The learning process results in the prediction at each time t coming to approximate the outcome, or target, that would follow it:

$$\boldsymbol{\phi}_t^\top \boldsymbol{\theta}_t \approx \sum_{k=t+1}^{\infty} R_k \prod_{j=t+1}^{k-1} \gamma_j$$

if actions were selected according to policy π, and where $\gamma_t \in [0,1]$ is a sequence of discount factors. We see from this equation why the signal R_t is termed the "cumulant"; all of its values are added up, or *accumulated*, within the temporal envelope specified by the γ_j. In the special case in which the cumulant is a reward and the γ_j are constant then the GVF reduces to a conventional value function from reinforcement learning.

To make the GVF problem well defined, the user must provide π and the γ_j. The policy π is not provided directly, but in the form of a sequence of importance sampling ratios

$$\rho_t = \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)},$$

where S_t and A_t are the state and action actually taken at time t, and $\pi(A_t|S_t)$ and $\mu(A_t|S_t)$ are the probabilities of A_t in S_t under policies π and μ respectively. The policy π is called the *target policy*, because it is under it that we are trying to predict the outcome, as stated above, and μ is called the *behavior policy*, because it is it that actually generates the behavior and the time series. Because only the ratio of the two probabilities is required, there is often no need to work directly with states or action probabilities. For example, in the on-policy case the target and behavior policies are the same, and the ratio is always one. The discount factors are often taken to be constant, but are allowed to depend arbitrarily on the time series, as long as $\prod_{j=t+1}^{\infty} \gamma_j = 0$ for all t.

In some publications concerning general value functions there is also specified a fourth sequence pertaining to the prediction problem—the "terminal pseudo reward" z_t—to specify a final signal to be added in with the cumulants at termination. More recently its has been recognized that this functionality can be included with just the cumulant R_t by appropriately setting the discount sequence γ_t (see Modayil, White & Sutton 2014). For example, if one wanted a terminal pseudo reward of z_t only upon termination, then one would use a cumulant of $R_t = (1-\gamma_t)z_t$.

In addition to the time series of the feature vectors and cumulant signals, the user must provide three sequences characterizing the nature of the approximation to be found by the algorithm:

- $I_t \in \mathbb{R}^+$; the *interest sequence* specifies the interest in or importance of accurately predicting at time t. For example, in episodic problems one may care only about the value of the first state of the episode; this is specified by setting $I_t = 1$ for the first state of each episode and $I_t = 0$ at all other times. (Or, as suggested by the work of Thomas (2014), one may want to use $I_t = \gamma^t$.) In a discounted continuing task, on the other hand, one often cares about all the states equally, which is specified by setting $I_t = 1$ for all t.

- $\lambda_t \in [0,1]$; the *bootstrapping* sequence specifies the degree of bootstrapping at each time.

- $\alpha_t \geq 0$; the step-size sequence specifies the size of the step at each time. One common choice is a constant step-size parameter, e.g., $\alpha_t = 0.1/\max_t \boldsymbol{\phi}_t^\top \boldsymbol{\phi}_t$.

Another common choice is a step-size parameter that decreases to zero slowly over time.

2 Algorithm Specification

Internal to the learning algorithm are the learned weight vector, $\boldsymbol{\theta}_t \in \mathbb{R}^n$, and an auxiliary shorter-term-memory vector $\boldsymbol{e}_t \in \mathbb{R}^n$ with $\boldsymbol{e}_t \geq \mathbf{0}$. In addition, there are the scalars $M_t \geq 0$ and $F_t \geq 0$. The emphasis M_t and the TD error δ_t are purely temporary variables. The true online emphatic TD(λ) algorithm is fully specified by the following equations:

$$\delta_t = R_{t+1} + \gamma_{t+1}\boldsymbol{\theta}_t^\top \boldsymbol{\phi}_{t+1} - \boldsymbol{\theta}_t^\top \boldsymbol{\phi}_t \tag{1}$$

$$F_t = \rho_{t-1}\gamma_t F_{t-1} + I_t, \qquad \text{with } F_{-1} = 0 \tag{2}$$

$$M_t = \lambda_t I_t + (1-\lambda_t)F_t \tag{3}$$

$$\boldsymbol{e}_t = \rho_t\gamma_t\lambda_t\boldsymbol{e}_{t-1} + \rho_t\alpha_t M_t(1-\rho_t\gamma_t\lambda_t\boldsymbol{\phi}_t^\top\boldsymbol{e}_{t-1})\boldsymbol{\phi}_t \qquad \text{with } \boldsymbol{e}_{-1} = 0 \tag{4}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \delta_t\boldsymbol{e}_t + (\boldsymbol{e}_t - \alpha_t M_t\rho_t\boldsymbol{\phi}_t)(\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1})^\top\boldsymbol{\phi}_t \tag{5}$$

3 Pseudocode

The following pseudocode characterizes the algorithm and its efficient implementation in C++. First the `init` function should be called with argument n (the number of components of $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$):

```
init(n):
  store n
  e ← 0
  θ ← 0        (or arbitrary)
  F ← D ← γ ← 0
```

On each step, $t = 0, 1, 2, \ldots$, the `learn` function is called with arguments $\alpha_t, I_t, \lambda_t, \boldsymbol{\phi}_t, \rho_t, R_{t+1}, \boldsymbol{\phi}_{t+1}, \gamma_{t+1}$:

```
learn(α, I, λ, φ, ρ, R, φ′, γ′):        ; α thru ρ are at t, the rest are at t + 1
  δ ← R + γ′θ⊤φ′ − θ⊤φ                  ; or, do all 3 inner products in a single loop
  F ← F + I                             ; F was ρ_{t−1}γ_t F_{t−1}; now it is F_t
  M ← λI + (1 − λ)F
  S ← ραM(1 − ργλφ⊤e)                   ; scalar S saves computation
  e ← ργλe + Sφ                         ; this + next 3 lines can be done in a single loop
  Δ ← δe + D(e − ραMφ)                  ; D here is (θ_t − θ_{t−1})⊤φ_t
  θ ← θ + Δ
  D ← Δ⊤φ′
  F ← ργ′F
  γ ← γ′
```

Finally, to obtain a prediction based on the learned weights, pass a feature vector to the `predict` function:

```
predict(φ):
  return θ⊤φ
```

If the task is episodic in the classical sense, then the terminal state should be represented as a special additional state at which $\gamma = 0$, $\boldsymbol{\phi} = \mathbf{0}$, and with outgoing transitions to the distribution of start states. As far as `learn` is concerned, there is still just a single sequence.

4 Code

Implementations that closely follow the pseudocode are provided for various programming languages in separate files. Where we have seen it as convenient and non-obfuscating, the implementations are in an object-oriented style in which one creates an instance of the algorithm that contains all of its internal variables.

Acknowledgements

The author gratefully acknowledges the assistance of Hado van Hasselt, Harm van Seijen, and A. Rupam Mahmood in preparing this guide.

References

Maei, H. R. (2011). *Gradient Temporal-Difference Learning Algorithms.* PhD thesis, University of Alberta.

Maei, H. R., Sutton, R. S. (2010). GQ(λ): A general gradient algorithm for

temporal-difference prediction learning with eligibility traces. In *Proceedings of the Third Conference on Artificial General Intelligence*, pp. 91–96. Atlantis Press.

Modayil, J., White, A., Sutton, R. S. (2014). Multi-timescale nexting in a reinforcement learning robot. *Adaptive Behavior 22*(2):146–160.

Singh, S. P., Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22:123–158.

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

Sutton, R. S., Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, Cs., Wiewiora, E. (2009). Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 993–1000, ACM.

Sutton, R. S., Mahmood, A. R., White, M. (2015). An emphatic approach to the problem of off-policy temporal-difference learning. ArXiv:1503.04269.

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., Precup, D. (2011). Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, pp. 761–768.

Thomas, P. (2014). Bias in natural actor–critic algorithms. In *Proceedings of the 31st International Conference on Machine Learning*. JMLR W&CP 32(1):441–448.

van Hasselt, H., Mahmood, A. R., Sutton, R. S. (2014). Off-policy TD(λ) with a true online equivalence. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, Quebec City, Canada.

van Seijen, H., Sutton, R. S. (2014). True online TD(λ). In *Proceedings of the 31st International Conference on Machine Learning*. Beijing, China. JMLR: W&CP volume 32.

van Seijen, H., Mahmood, A. R., Pilarski, P. M., Sutton, R. S. (2015). An empirical evaluation of true online TD(λ). In *Proceedings of the 2015 European Workshop on Reinforcement Learning*.

Yu, H. (2015). On convergence of emphatic temporal-difference learning. In *Proceedings of the Conference on Computational Learning Theory*.

White, A. (2015). *Developing a Predictive Approach to Knowledge*. Phd thesis, University of Alberta.