rl class - ass2 - Convergent Temporal-Difference Learning with Arbitrary Smooth Function Approximation

Jean Harb

March 2017

1 Introduction

When using temporal difference methods, the algorithms optimize the mean-squared bellman error (MSBE), that is, the $J(\theta) = \|(V_\theta - TV_\theta)\|^2 = \|(V_\theta - (r + \gamma V_\theta)\|^2$, where T is the bellman operator. This optimization however, assumes that the target value function is representable by the function approximation, an assumption that is not necessarily true, and if not, can lead to divergence. If the target value function if not representable by the function approximation, the update will change the function in an unknown way. TDC is an algorithm which changes the objective function to now take into consideration the fact that the target might not be representable. It does so by projecting the target back to the closest function that is actually representable (as demonstrated in Figure 1). They call this the mean-squared projected bellman error (MSPBE), given by $J(\theta) = (\Pi(V_\theta - TV_\theta))^2 = (V_\theta - \Pi TV_\theta)^2$, where Π is the projection operator defined as $\Pi V = \text{argmin}_{V' \in M}(V - V')^2$. This can be seen as finding the closest value function in the function approximation space which closest (in L2 norm) to a target value function V. Also notice that $\|(\Pi(V_\theta - TV_\theta))\|^2 = \|(V_\theta - \Pi TV_\theta)\|^2$, since $\Pi V_\theta = V_\theta$ as V_θ is already in M, making itself the closest point to itself. Let's define $\Phi_\theta \in R^{|S| \text{x} n}$ where $(\Phi_\theta)_{s,i} = \frac{\partial}{\partial \theta_i} V_\theta(s)$, meaning it's a matrix of gradients of each parameter in each state and $\Pi_\theta = \Phi_\theta(\Phi_\theta^T D \Phi_\theta)^{-1}\Phi_\theta^T D$, and D is a diagonal matrix with the stationary distribution of states on its diagonal.

Note that

$$\begin{aligned} \Pi^T D \Pi &= (\Phi(\Phi^T D \Phi)^{-1}\Phi^T D)^T D (\Phi(\Phi^T D \Phi)^{-1}\Phi^T D) \\ &= D^T \Phi(\Phi^T D \Phi)^{-1}\Phi^T D \end{aligned} \tag{1}$$

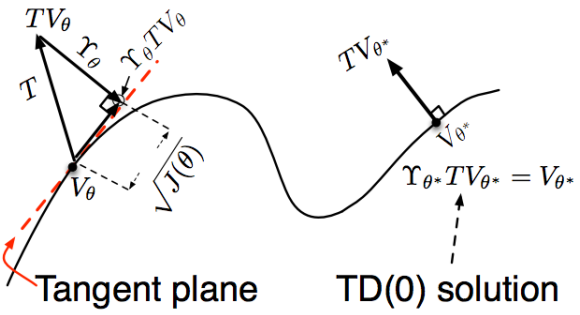

Figure 1:

And

$$\begin{aligned} J(\theta) &= \|(\Pi(V_\theta - TV_\theta))\|^2 \\ &= (\Pi(V_\theta - TV_\theta))^T D(\Pi(V_\theta - TV_\theta)) \\ &= (V_\theta - TV_\theta)^T \Pi^T D \Pi(V_\theta - TV_\theta) \\ &= (V_\theta - TV_\theta)^T D^T \Phi(\Phi^T D\Phi)^{-1}\Phi^T D(V_\theta - TV_\theta) \\ &= (\Phi^T D(TV_\theta - V_\theta))^T(\Phi^T D\Phi)^{-1}\Phi^T D(TV_\theta - V_\theta) \\ &= E[\delta\Phi]E[\Phi\Phi^T]^{-1}E[\delta\Phi] \end{aligned} \tag{2}$$

As seen in [1] we can derive the gradient of $J(\theta)$ in a straight forward way, and get the following. $-\frac{1}{2}\nabla J(\theta) = -E[\delta\phi] - \gamma E[\phi'\phi^T w] - E[(\delta - \phi^T w)\nabla^2 V(s)w]$, where $\phi = \nabla V_\theta(s)$, $\phi' = \nabla V_\theta(s')$ and $w = E[\phi\phi^T]^{-1}E[\delta\phi]$.

The only problem, is that we have a multiplication of 2 expectation, which means we can't update using the same samples. For that reason, we split the update of each (J and w) and training them at different timescales, still using the same samples to stay efficient.

Finally, this gives us the following training algorithm. We update as follows

$$w_{k+1} = w_k + \beta_k(\delta_k - \phi_k^T w_k)\phi_k \tag{3}$$

$$\theta_{k+1} = \theta_k + \alpha_k(\delta_k\phi_k - \gamma\phi'_k(\phi_k^T w_k) - (\delta_k - \phi_k^T w_k)\nabla^2 V_{\theta_k}(s_k)w_k) \tag{4}$$

2 Implementation and Results

Attached is some code which runs TD(0) and TDC, and generates plots comparing the MSE of the estimated value function with the true values, through training.

We tested TDC on the "spiral" counterexample in [2]. In this example, the value function is represented by a non-linear function approximation with one parameter. The MDP is simple, with three states, where there is no reward,

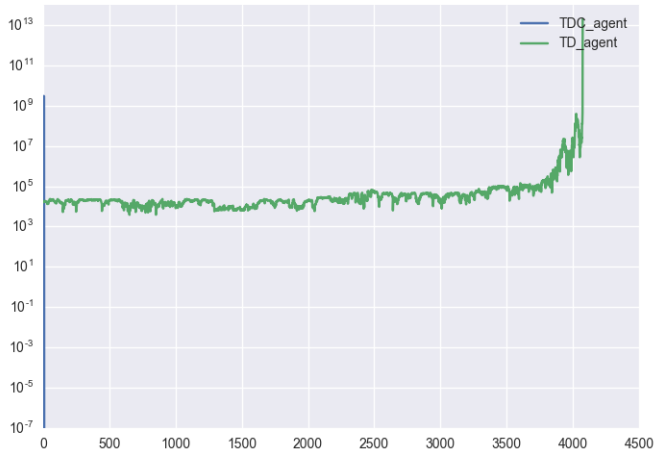

Figure 2: Comparison of TD(0) and TDC on a counterexample showing TD diverging. On the other hand, TDC converges in just a few steps.

and the agent has a 50% chance of staying in the same state and a 50% of moving to the next state (in a circular manner, so state 1 can go to 2, 2 to 3, and 3 to 1). Since there is no reward, all the agent has to learn is that the value function is 0 at all states. The approximation is given by $V_\theta(s) = \left(a(s)\cos(\hat{\lambda}\theta) - b(s)\sin(\hat{\lambda}\theta)\right)e^{\epsilon\theta}$. We start with $V_0 = (100, -70, -30)$, $a = V_0$, $b = (23.094, -98.15, 75.056)$, $\hat{\lambda} = 0.866$ and $\epsilon = 0.05$. Finally, we trained with $\alpha = 0.5$ and $\beta = 0.05$ for TDC and $\alpha = 0.001$ for TD(0). The true value function is $V = (0, 0, 0)$, and is attained when $\theta \to -\infty$. We calculate the error as the L2 norm of the distance of the estimated value function and the true value function.

As we can see Figure 2, the agent using TDC learns the correct values extremely quickly, within 10 epochs, whereas the TD(0) agent never learns and diverges.

References

[1] Shalabh Bhatnagar, Doina Precup, David Silver, Richard S Sutton, Hamid R Maei, and Csaba Szepesvári. Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in Neural Information Processing Systems*, pages 1204–1212, 2009.

[2] John N Tsitsiklis, Benjamin Van Roy, et al. An analysis of temporal-difference learning with function approximation.