Convergent Temporal-Difference Learning with Arbitrary Smooth Function Approximation

Bhatnagar, Shalabh, et al.

COMP767 - Reinforcement Learning
Claudio Sole

École polytechnique de Montréal

March 23, 2017

Introduction

- TD(λ) is know to converge when used with **linear** functions approximators in the **on-policy** learning scenario
- The absence of these requirements can cause the parameters of the function approximator to diverge when trained with TD methods
- Purpose of the study: defining the first TD algorithms that are stable when used with smooth nonlinear function approximators

Idea

Extends the algorithms like GTD2 and TDC (linear function approximators, off-policy scenario), designed to follow the gradient of an objective function whose unique optimum is the fixed point of the original TD(0) algorithm

TD algorithms with function approximation I

- TD(0) with function approximation:
  - starts with arbitrary θ_0
  - after observing the k^{th} transition, compute the *td-error*

    $$\delta_k = r_k + \gamma V_\theta(s'_k) - V_\theta(s_k)$$

    and compute the update

    $$\theta_{k+1} = \theta_k + \alpha_k \delta_k \nabla V_{\theta_k}(s_k)$$

- gradient-descent methods for TD learning (linear approximators)

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

$$\theta_{k+1} = \theta_k + \alpha_k(\phi_k - \gamma\phi'_k)(\phi_k^T w_k) \qquad GTD2$$

$$\theta_{k+1} = \theta_k + \alpha_k \delta_k \phi_k - \alpha_k \gamma \phi'_k(\phi_k^T w_k) \qquad TDC$$

The problem

The first step is to find an objective function:

- in the linear case MSPBE: projection of the Bellman error on a natural hyperplane of the possible V_θ
- with non linear function approximation, the value function is no longer restricted to an hyperplane, but can move on a nonlinear surface
  - projected onto a nonlinear manifold is not computationally feasible
- **Assumption:** θ changes very little from one step to the next one $\implies$ surface close to linear

Idea

We can project onto the tangent plane at a given point

Definitions

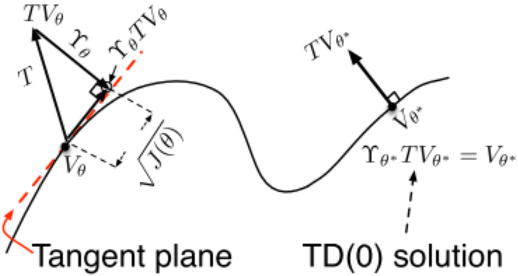
- *Tangent plane* $P\mathcal{M}_\theta$: hyperplane that passes through V_θ and orthogonal to the normal of the manifold $\mathcal{M}$ at θ
- *Tangent space* $T\mathcal{M}_\theta = \{\Phi_\theta a | a \in \mathbb{R}^n\}$ translation of $P\mathcal{M}_\theta$ into the origin. $\Phi_\theta \in \mathbb{R}^{|S|\times n}$, with $\left(\Phi_\theta\right)_{s,i} = \frac{\partial}{\partial\theta_i} V_\theta(s)$
- Projection operator Π_θ which projects a value function onto $T\mathcal{M}_\theta$:

$$\Pi_\theta = \Phi_\theta(\Phi_\theta^T D \Phi_\theta)^{-1}\Phi_\theta^T D$$

Objective function

- Objective function:

$$J(\theta) = ||\Pi_\theta(TV_\theta - V_\theta)||_D^2 \qquad (1)$$



Derivation of the algorithm (1)

- The objective function can be rewritten as

$$J(\theta) = \mathbb{E}\big[\delta \nabla V_\theta(s)\big]^T \mathbb{E}\big[\nabla V_\theta(s) \nabla V_\theta(s)^T\big]^{-1} \mathbb{E}\big[\delta \nabla V_\theta(s)\big]$$

- Let's define $\phi \equiv \nabla V_\theta(s)$ and $\phi' \equiv \nabla V_\theta(s')$. Then

$$\begin{aligned} -\frac{1}{2}\nabla J(\theta) &= -\mathbb{E}\big[(\gamma\phi' - \phi)\phi^T w\big] + h(\theta, w) \\ &= -\mathbb{E}[\delta\phi] - \gamma\mathbb{E}[\phi'\phi^T w] + h(\theta, w) \end{aligned}$$

with

$$\begin{aligned} w &= \mathbb{E}[\phi\phi^T]^{-1}\mathbb{E}[\delta\phi] \\ h(\theta, w) &= -\mathbb{E}[(\delta - \phi^T w)\nabla^2 V_\theta(s) w]] \end{aligned}$$

Derivation of the algorithm (2)

- The resulting updates are a generalization of the ones of GTD2 and TDC:

$$non - linear \;\; GTD2$$

$$\theta_{k+1} = \Gamma\Big(\theta_k + \alpha_k\{(\phi_k - \gamma\phi'_k)(\phi_k^T w_k) - h_k\}\Big)$$

$$non - linear \;\; TDC$$

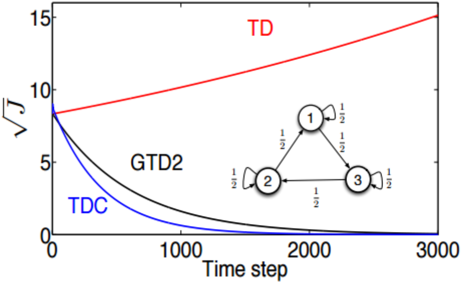
$$\theta_{k+1} = \Gamma\Big(\theta_k + \alpha_k\{(\delta_k\phi_k - \gamma\phi'_k)(\phi_k^T w_k) - h_k\}\Big)$$

- $\Gamma : \mathbb{R}^n \to \mathbb{R}^n$ is a mapping whose purpose is to prevent parameter divergence in the initial phase of the algorithm (possible due to nonlinearities)
- for the weight vector, the update is the same of the linear case:

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

Results

$$V_\theta(s) = \Big(a(s)cos(\lambda\theta) - b(s)sin(\lambda\theta) \Big) e^{\epsilon\theta} \qquad (2)$$



References I

Bhatnagar, Shalabh, et al.
Convergent temporal-difference learning with arbitrary smooth function approximation.
Advances in Neural Information Processing Systems.
2009.