Foundations of Reinforcement Learning

The deadly triad, function approximation in PG, and actor-critic

Yuejie Chi

Department of Electrical and Computer Engineering

Carnegie Mellon University

Spring 2023

Outline

TD(0) with linear function approximation

Suppose we collect a trajectory following policy π:

$$s_0, r_0, s_1, r_1, s_2, r_2, \ldots$$

The value function of π is approximated as

$$V^\pi(s) \approx \phi(s)^\top w.$$

TD(0) on a single trajectory:

$$w_{t+1} \leftarrow w_t + \alpha_t \underbrace{\left(r_t + \gamma \phi(s_{t+1})^\top w_t - \phi(s_t)^\top w_t\right)}_{\text{TD error } \delta_t} \phi(s_t)$$

Applying TD(0) to on-policy control

SARSA with linear function approximation:

- Approximate the *on-policy* Q-function with

$$Q(s, a; w) = \psi(s, a)^\top v,$$

- **Policy evaluation:** apply TD(0) to update the weight

$$v_{t+1} \leftarrow v_t + \alpha \left(r_t + \gamma \psi(s_{t+1}, a_{t+1})^\top v_t - \psi(s_t, a_t)^\top v_t \right) \psi(s_t, a_t)$$

- **Policy improvement:** ϵ-greedy policy improvement

Off-policy evaluation with function approximation

Suppose we collect a trajectory following behavior policy π_b:

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2, \ldots$$

with $a_t \sim \pi_\mathsf{b}(\cdot|s_t)$.

Off-policy evaluation

How do we perform off-policy evaluation using TD(0) with function approximation, when the policy under evaluation π is different from π_b?

TD(0) updates with importance sampling

$$J(w) = \frac{1}{2}\mathbb{E}_{s\sim d^\pi}\underbrace{\left[\left(V^\pi(s) - V(s;w)\right)^2\right]}_{=:J(s;w)} = \frac{1}{2}\mathbb{E}_{s\sim d^\pi}\left[\left(V^\pi(s) - \phi(s)^\top w\right)^2\right].$$

- Using the TD target $r_t + \gamma V(s_{t+1}, w) = r_t + \gamma\phi(s_{t+1})^\top w$, the semi-gradient is evaluated as

$$\nabla_w J(s_t;w) = -\underbrace{\left(r_t + \gamma\phi(s_{t+1})^\top w - \phi(s_t)^\top w\right)}_{\text{TD error } \delta_t}\phi(s_t).$$

- Update the weight w via

$$w_{t+1} = w_t - \alpha_t \underbrace{\frac{\pi(a_t|s_t)}{\pi_{\mathsf{b}}(a_t|s_t)}}_{=:\rho_t}\nabla_w J(s_t;w) = w_t + \alpha_t\rho_t\delta_t\phi(s_t).$$

Q-learning with linear function approximation

Q-learning with linear function approximation:

- Approximate the *off-policy* Q-function with

$$Q(s, a; w) = \psi(s, a)^\top v,$$

- **Policy evaluation:** using *Q-learning target* to update the weight

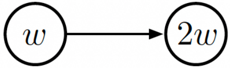
$$v_{t+1} \leftarrow v_t + \alpha \left(r_t + \gamma \max_a \psi(s_{t+1}, a)^\top v_t - \psi(s_t, a_t)^\top v_t \right) \psi(s_t, a_t)$$

- **Policy improvement:** ϵ-greedy policy improvement

The deadly triad

Off-policy TD(0) might diverge

Intuition: only one action is available, and it results deterministically in a transition to the second state with a reward of 0 [Sutton and Barto, 2018]:



- The linear function approximation assumes the value takes the form

$$[w,\ 2w] \qquad \text{with} \quad \phi(\text{left}) = 1,\ \phi(\text{right}) = 2.$$

- For one transition from left state to right state, we have

$$\delta_t = r_t + \gamma V(\text{right}) - V(\text{left}) = \gamma 2w_t - w_t = (2\gamma - 1)w_t,$$
$$\rho_t = 1.$$

- The off-policy TD(0) updates

$$w_{t+1} = w_t + \alpha_t \rho_t \delta_t \phi(\text{left}) = (1 + \alpha_t(2\gamma - 1))\, w_t.$$

Diverges whenever $\gamma > 1/2$ for any $\alpha_t > 0$ if we do this over and over!

Baird's example

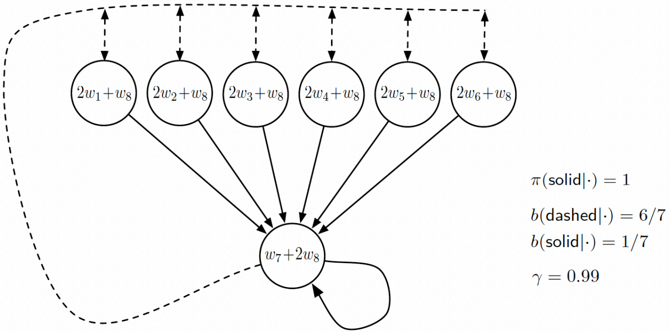


Figure 11.1: Baird's counterexample. The approximate state-value function for this Markov process is of the form shown by the linear expressions inside each state. The solid action usually results in the seventh state, and the dashed action usually results in one of the other six states, each with equal probability. The reward is always zero.

Figure source: [Sutton and Barto, 2018]

Baird's example explained

- 7 states, feature dimension $= 8!!!$
- The set of features is linearly independent, e.g.

$$\phi(1) = [2, 0, 0, 0, 0, 0, 0, 1]^\top$$

- The true value function is

$$V^\pi(s) = 0, \quad \text{which can be exactly approximated by} \quad w = 0.$$

- The behavior policy π_b offers a path to skip the absorbing state 8 of π, creating a path mimicking our intuition earlier (focusing on w_8).
- We will be okay with on-policy evaluation.

Numerical divergence on Baird's example

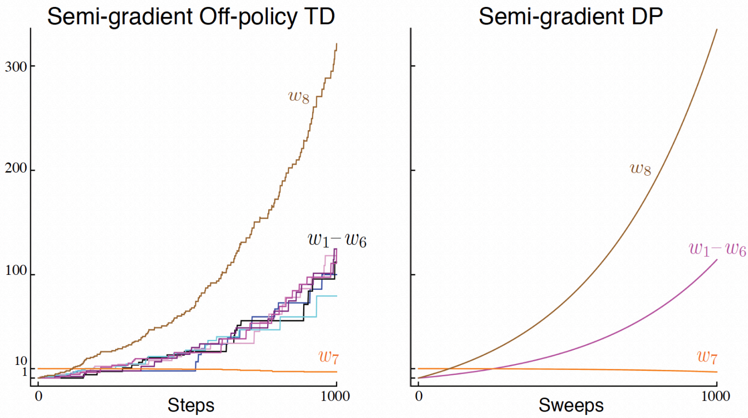
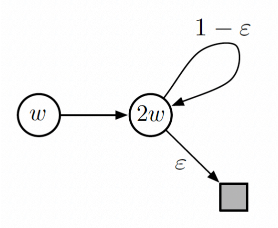


Figure 11.2: Demonstration of instability on Baird's counterexample. Shown are the evolution of the components of the parameter vector $\mathbf{w}$ of the two semi-gradient algorithms. The step size was $\alpha = 0.01$, and the initial weights were $\mathbf{w} = (1, 1, 1, 1, 1, 1, 10, 1)^\top$.

Figure source: [Sutton and Barto, 2018]

Does LSTD resolve the issue?



- Tsitsiklis and Van Roy's Counterexample: the reward is zero on all transitions, so the true value function is

$$V^\pi(s) = 0, \quad \text{and} \quad w = 0.$$

- Suppose we use least-squares at each step with **DP** to update

$$\begin{aligned} w_{t+1} &= \arg\min_{w\in\mathbb{R}} \sum_{s\in\mathcal{S}} \left(\widehat{V}(s,w) - \mathbb{E}_\pi[r_t + \gamma \widehat{V}(s_{t+1}, w_k) | S_t = s]\right)^2 \\ &= \arg\min_{w\in\mathbb{R}} (w - \gamma 2 w_k)^2 + (2w - (1-\varepsilon)\gamma 2 w_k)^2 \\ &= \frac{6-4\varepsilon}{5}\gamma w_k, \qquad \text{which diverges as long as} \quad \gamma > \frac{5}{6-4\varepsilon}. \end{aligned}$$

The deadly triad

The risk of divergence arises whenever we combine:

Richard Sutton

- **Function approximation:**
  significantly generalizing from large numbers of examples

- **Bootstrapping:**
  learning value estimates from other value estimates, as in dynamic programming and temporal-difference learning

- **Off-policy learning:**
  learning about a policy from data not due to that policy, as in Q-learning, where we learn about the greedy policy from data with a necessarily more exploratory policy

Any two without the third is okay.

Possible remedies

- More careful algorithm designs [Sutton et al., 2009]:
  - Gradient TD (GTD)
  - TD with gradient correction (TDC)
  - Emphatic TD [Sutton et al., 2016], etc...
- Using a target network [Mnih et al., 2015, Zhang et al., 2021]:

$$f(s_t, a_t; v) = \frac{1}{2}\left(r_t + \gamma \max_a Q_{\text{target}}(s_{t+1}, a; v) - Q(s_t, a_t; v)\right)^2$$

  - Target network Q_{target}: periodically synced by the value network.
  - Value network Q: updated via gradient methods.

A key ingredients in *(double) deep Q-learning (DQN)*.

Function approximation in policy gradient and actor-critic

Recall: policy gradient methods

Recall the policy gradient expression

$$\nabla_\theta V^{\pi_\theta}(\rho) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\rho^{\pi_\theta}, a\sim\pi_\theta(\cdot|s)}\Big[Q^{\pi_\theta}(s,a)\nabla\log\pi_\theta(a|s)\Big],$$

where

- $d_\rho^{\pi_\theta}$ is the state visitation distribution,
- $\nabla\log\pi_\theta(a|s)$ is the score function.

Function approximation in PG

How do we inject function approximation into policy gradient methods?

Answer: using a **critic** with function approximation

$$Q^{\pi_\theta}(s,a) \approx Q_w(s,a)$$

parameterized by some w.

Actor-critic framework

- **Critic:** update the parameter w of the Q-function $Q_w(s,a)$ by approximately minimizing

$$J_{\text{critic}}(w) = \mathbb{E}_{s\sim d_\rho^{\pi_\theta}, a\sim\pi_\theta(\cdot|s)}\left[\left(Q_w(s,a) - Q^{\pi_\theta}(s,a)\right)^2\right]$$

- **Actor:** update the parameter θ of the policy π_θ, by moving along the policy gradient

$$\nabla_\theta V^{\pi_\theta}(\rho) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d_\rho^{\pi_\theta}, a\sim\pi_\theta(\cdot|s)}\left[\overset{Q_w(s,a)}{\cancel{Q^{\pi_\theta}(s,a)}}\nabla\log\pi_\theta(a|s)\right],$$

How does value function approximation impacts the evaluation of the policy gradient?

Compatible function approximation

Theorem 1 (Compatible function approximation)

If $Q_w(s,a)$ is compatible to the policy, i.e.

$$\nabla_w Q_w(s,a) = \nabla_\theta \log \pi_\theta(a|s),$$

then the policy gradient is still unbiased if w is a stationary point of $J_{\text{critic}}(w)$:

$$\mathbb{E}_{s,a\sim d_\rho^{\pi_\theta}}\left[Q^{\pi_\theta}(s,a)\nabla \log \pi_\theta(a|s)\right] = \mathbb{E}_{s,a\sim d_\rho^{\pi_\theta}}\left[Q_w(s,a)\nabla \log \pi_\theta(a|s)\right].$$

- This allows us to use $Q_w(s,a)$ in the policy gradient without introducing bias.
- One possible candidate:

$$Q_w(s,a) = w^\top \phi(s,a), \qquad \pi_\theta(a|s) \propto \exp(\theta^\top \phi(s,a))$$

Proof

Suppose we find w that is a stationary point of $J_{\text{critic}}(w)$, it holds that

$$\mathbb{E}_{s,a\sim d_\rho^{\pi_\theta}}\left[\big(Q_w(s,a) - Q^{\pi_\theta}(s,a)\big)\nabla_w Q_w(s,a)\right] = 0.$$

$$\Updownarrow$$

$$\mathbb{E}_{s,a\sim d_\rho^{\pi_\theta}}\left[Q_w(s,a)\nabla_w Q_w(s,a)\right] = \mathbb{E}_{s,a\sim d_\rho^{\pi_\theta}}\left[Q^{\pi_\theta}(s,a)\nabla_w Q_w(s,a)\right]$$

$$\Updownarrow$$

$$\mathbb{E}_{s,a\sim d_\rho^{\pi_\theta}}\left[Q_w(s,a)\nabla_\theta \log \pi_\theta(a|s)\right] = \mathbb{E}_{s,a\sim d_\rho^{\pi_\theta}}\left[Q^{\pi_\theta}(s,a)\nabla_\theta \log \pi_\theta(a|s)\right].$$

Reducing variance using a baseline

Instead of using $Q^{\pi_\theta}(s,a)$ in the policy gradient, we can use the advantage function

$$A^{\pi_\theta}(s,a) = Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s),$$

which helps reduce the variance.

- We can set the critic to estimate the advantage function instead
- **Key observation:** the TD error

  $$\delta^{\pi_\theta} = r + \gamma V^{\pi_\theta}(s') - V^{\pi_\theta}(s)$$

  is an unbiased estimate of the advantage function

  $$\begin{aligned}\mathbb{E}[\delta^{\pi_\theta}|s,a] &= \mathbb{E}[r + \gamma V^{\pi_\theta}(s')|s,a] - V^{\pi_\theta}(s)\\ &= Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)\\ &= A^{\pi_\theta}(s,a)\end{aligned}$$

Actor-critic with TD error

Use the TD error for policy gradient

$$\nabla_\theta V^{\pi_\theta}(\theta) = \mathbb{E}\left[\nabla_\theta \log \pi_\theta(s|a)\delta^{\pi_\theta}\right]$$

This only requires one set of critic parameter:

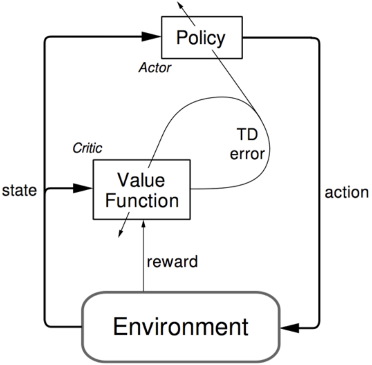
- Compute the TD error

$$\delta^{\pi_\theta} = r + \gamma V^{\pi_\theta}(s') - V^{\pi_\theta}(s)$$

- Update the policy parameter

$$\theta \leftarrow \theta + \beta \delta^{\pi_\theta} \nabla_\theta \log \pi_\theta(a|s)$$

where β is the learning rate.



Natural actor-critic

Consider the linear value approximation

$$A_w(s, a) = w^\top \underbrace{\nabla_\theta \log \pi_\theta(a|s)}_{\text{features}},$$

where the compatible function approximation holds

$$\nabla_w A_w(s, a) = \nabla_\theta \log \pi_\theta(a|s),$$

the natural gradient simplifies.

- Let w be the minimizer of

$$\min_w \mathbb{E}\left[\left(A_w(s, a) - A^{\pi_\theta}(s, a)\right)^2\right] = \mathbb{E}\left[\left(w^\top \nabla_\theta \log \pi_\theta(a|s) - A^{\pi_\theta}(s, a)\right)^2\right].$$

Natural actor-critic

- The policy gradient reduces to

$$\begin{aligned}\nabla_\theta V^{\pi_\theta}(\theta) &= \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a|s) A^{\pi_\theta}(s,a)\right] \\ &= \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a|s) A_w(s,a)\right] \\ &= \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^\top w\right] \\ &= F_\theta w,\end{aligned}$$

  where F_θ is the Fisher information matrix.

- The NPG update is thus

$$\theta \leftarrow \theta + \beta (F_\theta)^\dagger \nabla_\theta V^{\pi_\theta}(\theta) = \theta + \beta w.$$

  Update the actor directly in the direction of w!

References I

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015).
Human-level control through deep reinforcement learning.
Nature, 518(7540):529–533.

Sutton, R. S. and Barto, A. G. (2018).
Reinforcement learning: An introduction.
MIT press.

Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C., and Wiewiora, E. (2009).
Fast gradient-descent methods for temporal-difference learning with linear function approximation.
In *Proceedings of the 26th annual international conference on machine learning*, pages 993–1000.

Sutton, R. S., Mahmood, A. R., and White, M. (2016).
An emphatic approach to the problem of off-policy temporal-difference learning.
The Journal of Machine Learning Research, 17(1):2603–2631.

Zhang, S., Yao, H., and Whiteson, S. (2021).
Breaking the deadly triad with a target network.
In *International Conference on Machine Learning*, pages 12621–12631. PMLR.