POLICY GRADIENT METHODS

So far: local view using Bellman equations

$$Q^*(s,a) = \dots \quad Q^*(s',a')$$

⇝ VALUE-BASED METHODS

Now: Different approach: global view

$$\max_\theta \overbrace{\mathbb{E}_{\tau \sim \pi_\theta}\Big[\underbrace{R_0 + \gamma R_1 + \gamma^2 R_2 + \dots}_{R(\tau)}\Big]}^{J(\theta)}$$

actual objective

using (stochastic) gradient ascent

Key question: How do we compute $\nabla_\theta J(\theta)$?

Policy Gradient Theorem:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t \geq 0} R(\tau) \nabla_\theta \log \pi_\theta(a_t | s_t)\right]$$

s_t : state at time t

a_t : action at time t

Let us define this function:

$$\mathcal{L}(\theta) = -\mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t \geq 0} R(\tau) \log \pi_\theta(a_t | s_t)\right]$$

THIS IS **NOT** A LOSS FUNCTION

But:

$\nabla_\theta \mathcal{L}(\theta) = \nabla_\theta J(\theta)$ it has the right gradient!

REINFORCE ALGORITHM

(1) Generate a batch of episodes $\tau_1, \dots, \tau_B$

(2) For each episode compute $R(\tau_1), \dots, R(\tau_B)$

(3) Estimate the hacky loss :

$$\mathcal{L}(\theta) \simeq \frac{1}{B} \sum_{i=1}^{B} R(\tau_i) \sum_{t \geq 0} \log \pi_\theta(a_t^i | s_t^i)$$

(4) Compute $\nabla_\theta \mathcal{L}(\theta)$

(5) Update policy

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

Issue: requires full trajectories

↳ very high variance

Expected Grad-Log-Prob Lemma (EGLP):

$$\mathbb{E}_{x \sim P_\theta}\left[\nabla_\theta \log P_\theta \right] = 0$$

Consequence of $\int P_\theta(x)\, dx = 1$

Improvement #1: reward-to-go

Replace $R(\tau)$ by "reward-to-go":

$$G_t = \sum_{t' \geq t} \gamma^{t'-t} r_{t'}$$

instead of $R(\tau)$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t \geq 0} \gamma^t G_t \nabla_\theta \log \pi_\theta(a_t|s_t) \right]$$

define:

$$\mathcal{L}(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t \geq 0} \gamma^t G_t \log \pi_\theta(a_t|s_t) \right]$$

we have $\nabla_\theta \mathcal{L}(\theta) = \nabla_\theta J(\theta)$

Improvement #2 : baselines

Consequence of EGLP

$$\nabla_\theta \mathop{\mathbb{E}}_{a_t \sim \pi_\theta} \left[\log \pi_\theta(a_t|s_t)\, b(s_t) \right] = 0$$

for any function b

Natural choice for baseline:

$b(s_t) = V(s_t)$ on-policy value function

Yields:

$$\nabla_\theta J(\theta) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \left[\sum_{t \geq 0} \gamma^t V(s_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \right]$$

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \left[\sum_{t \geq 0} \gamma^t V(s_t) \log \pi_\theta(a_t|s_t) \right]$$

→ Reduces the variance, BUT:

→ we have a new problem: computing V

ESTIMATING THE VALUE FUNCTION

This is the goal of value-based methods!

Long story short:

(*) use a neural network to represent V

(*) we add to the loss a term called

value loss:

$$\mathcal{L}(\theta') = \mathbb{E}_{\tau \sim G_\theta} \left[\sum_{t \geq 0} \left(V^{\theta'}(s_t) - G_t \right)^2 \right]$$

Mean Squared Error (MSE)

At this point we have an algorithm

called REINFORCE

also: Monte Carlo Policy Gradient

also: Vanilla Policy Gradient