DS 598
Introduction to RL

Xuezhou Zhang

Chapter 5: Policy-based RL (continued)

The REINFORCE algorithm

1. Initialize θ_0
2. For iteration t = 0,…,T
   1) Run π_{θ_t} and collect trajectories $\tau_1, \ldots, \tau_n$
   2) Estimate the PG by

$$g_t = \frac{1}{n} \sum_{i=1}^{n} \left[\sum_{h=0}^{\infty} \nabla_\theta \log \pi(a_{i;h} | s_{i;h}) R(\tau_i) \right]$$

   3) Do SGD update $\theta_{t+1} = \theta_t + \alpha_t g_t$

The REINFORCE algorithm

- Pros:
  - ✓Convergence
  - ✓Conceptually simple

- Cons:
  - ❖Only works with stochastic policies
  - ❖On-policy -> Sample inefficient
  - ❖High Variance $\mathbb{E}\left[||g_t - \nabla_\theta J(\pi_\theta)||_2^2 \right]$

High Variance

- $\mathbb{E}\left[||g_t - \nabla_\theta J(\pi_\theta)||_2^2\right]$ can be up to $(1-\gamma)^3$.

$$g_t = \frac{1}{n}\sum_{i=1}^{n}\left[\sum_{h=0}^{\infty}\nabla_\theta \log \pi(a_{i;h}|s_{i;h})R(\tau_i)\right]$$

Reducing Variance

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{h=0}^{\infty} \nabla_\theta \log \pi(a_h|s_h) R(\tau)\right]$$

- Key Observation: $\mathbb{E}_{a\sim\pi(s)}\left[\nabla_\theta \log \pi(a|s) f(s)\right] = 0$

- Therefore,

$$\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{h=0}^{\infty} \nabla_\theta \log \pi(a_h|s_h) \sum_{i=0}^{\infty} \gamma^h r(s_i, a_i)\right] \\
&= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{h=0}^{\infty} \nabla_\theta \log \pi(a_h|s_h) \sum_{i=h}^{\infty} \gamma^h r(s_i, a_i)\right] \\
&= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{h=0}^{\infty} \nabla_\theta \log \pi(a_h|s_h) Q^{\pi_\theta}(s_h, a_h)\right] \\
&= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{h=0}^{\infty} \nabla_\theta \log \pi(a_h|s_h) \left(Q^{\pi_\theta}(s_h, a_h) - b(s_h)\right)\right]
\end{aligned}$$

Reducing Variance

- A “good” baseline: $b(s) = V^{\pi_\theta}(s)$.

$$
\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{h=0}^{\infty} \nabla_\theta \log \pi(a_h|s_h)\left(Q^{\pi_\theta}(s_h,a_h) - b(s_h)\right)\right] \\
&= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{h=0}^{\infty} \nabla_\theta \log \pi(a_h|s_h)\left(Q^{\pi_\theta}(s_h,a_h) - V^{\pi_\theta}(s_h)\right)\right] \\
&= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{h=0}^{\infty} \nabla_\theta \log \pi(a_h|s_h) A^{\pi_\theta}(s_h,a_h)\right]
\end{aligned}
$$

Advantage function: $A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$

Reducing Variance

- Sure, but how do we estimate $A^{\pi_\theta}(s, a)$?
- Estimate $V^{\pi_\theta}(s)$ using TD-learning.

$$\hat{V}^{\pi} = \arg\min_{f} \sum_{i=1}^{N} \sum_{h=1}^{H} \left[f(s_{i;h}) - (r_{i;h} + \gamma r_{i;h+1} + \gamma^2 r_{i;h+2} + \ldots +) \right]^2$$

- Then, $A^{\pi_\theta}(s_h, a_h) \approx r_{i;h} + \gamma r_{i;h+1} + \gamma^2 r_{i;h+2} + \ldots - \hat{V}^{\pi}(s_{i;h})$
- This is the so-called Advantage Actor-Critic (A2C) algorithm.

The A2C algorithm

1. Initialize θ_0
2. For iteration t = 0,…,T
   1) Run π_{θ_t} and collect trajectories $\tau_1, \ldots, \tau_n$
   2) Update Critic: $\hat{V} = \arg\min_f \sum_{i=1}^{N} \sum_{h=1}^{H} \left[f(s_{i;h}) - (r_{i;h} + \gamma r_{i;h+1} + \gamma^2 r_{i;h+2} + \ldots +) \right]^2$
   3) Estimate the PG by
   $$g_t = \frac{1}{n} \sum_{i=1}^{n} \left[\sum_{h=0}^{\infty} \nabla_\theta \log \pi(a_{i;h}|s_{i;h})(R(s_{i;h}, a_{i;h}) - \hat{V}(s_{i;h})) \right]$$
   4) Do SGD update $\theta_{t+1} = \theta_t + \alpha_t g_t$

The A2C algorithm

- Pros:
  - ✓Convergence
  - ✓Conceptually simple
  - ✓Low PG Variance

- Cons:
  - ❖Only works with stochastic policies $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{h=0}^{\infty} \nabla_\theta \log \pi(a_h|s_h) R(\tau)\right]$
  - ❖On-policy -> Sample inefficient

Can we make use of off-policy data?

importance sampling

$$
\begin{aligned}
E_{x\sim p(x)}[f(x)] &= \int p(x)f(x)dx \\
&= \int \frac{q(x)}{q(x)}p(x)f(x)dx \\
&= \int q(x)\frac{p(x)}{q(x)}f(x)dx \\
&= E_{x\sim q(x)}\left[\frac{p(x)}{q(x)}f(x)\right]
\end{aligned}
$$

IS-based Off-Policy Policy Gradient

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{h=0}^{\infty}\nabla_\theta \log \pi_\theta(a_h|s_h)R(\tau)\right]$$

$$= \mathbb{E}_{\tau\sim\pi_{\theta'}}\left[\frac{p_\theta(\tau)}{p_{\theta'}(\tau)}\sum_{h=0}^{\infty}\nabla_\theta \log \pi_\theta(a_h|s_h)R(\tau)\right]$$

$$= \mathbb{E}_{\tau\sim\pi_{\theta'}}\left[\prod_{h=0}^{\infty}\frac{\pi_\theta(a_h|s_h)}{\pi_{\theta'}(a_h|s_h)}\sum_{h=0}^{\infty}\nabla_\theta \log \pi_\theta(a_h|s_h)R(\tau)\right]$$

Exponentially large -> blows up the variance

The IS-PG algorithm

- Pros:
  - ✓ Convergence
  - ✓ Conceptually simple
  - ✓ Low PG Variance
  - ✓ Can be off-policy

- Cons:
  - ❖ Only works with stochastic policies

Deterministic Policy Gradient Theorem

- Deterministic Policy: $\mathrm{a} = \pi_\theta(s)$

- Deterministic Policy Gradient Theorem

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s \sim d^{\pi_\theta}} \left[\nabla_\theta \pi(s) \nabla_a Q^{\pi_\theta}(s, \pi_\theta(a)) \right]$$

Deterministic Policy Gradient Theorem

- How to learn Q^{π_θ}?

- For stochastic policies: $Q^\pi(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}\left[r(s_t, a_t) + \gamma\, \mathbb{E}_{a_{t+1} \sim \pi}\left[Q^\pi(s_{t+1}, a_{t+1})\right]\right]$

- For deterministic policies: $Q^\mu(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}\left[r(s_t, a_t) + \gamma Q^\mu(s_{t+1}, \mu(s_{t+1}))\right]$

- Can do off-policy learning!

Deep Determinisitic Policy Gradient (DDPG)

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

technically wrong

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for