TEK 5040/9040 Reinforcement Learning (Supplement)

Narada Warakagoda

Policy gradient improvements

- Disadvantage of simple policy gradients
  - Actions are tied to all rewards including past rewards
    - Solution: Reward-to-go policy gradient
  - Estimation of policy gradients has high variance
    - Solution: Baseline
  - Gradient based parameter update can take the policy too far away
    - Solution 1: Trust Region Policy Optimization
    - Solution2: Proximal Policy Optimization

Reward-to-go policy gradient

- Instead of using all rewards, use only future rewards

$$\nabla_\theta J(\pi_\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau) \right].$$

$$\nabla_\theta J(\pi_\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t'=t}^{T} R(s_{t'}, a_{t'}, s_{t'+1}) \right].$$

Reward-to-go

Baseline

- Any quantity independent of policy network parameters θ
- Usually the value function $b(s_t) = V^{\pi}(s_t)$
- Can show that the modified gradient estimate
  - Is same as the gradient estimate without baseline
  - Has a lower variance

$$\nabla_\theta J(\pi_\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\left(\sum_{t'=t}^{T} R(s_{t'}, a_{t'}, s_{t'+1}) - \underbrace{b(s_t)}_{\text{Baseline}}\right)\right]$$

Generalized form of policy gradients (PG)

- General form of policy gradients

$$\nabla_\theta J(\pi_\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \Phi_t \right],$$

  with following possibilities

- Simple PG $\Phi_t = R(\tau)$
- Reward-to-go $\Phi_t = \sum_{t'=t}^{T} R(s_{t'}, a_{t'}, s_{t'+1}).$
- Baseline

$$\Phi_t = \sum_{t'=t}^{T} R(s_{t'}, a_{t'}, s_{t'+1}) - b(s_t).$$

- Action-value function $\Phi_t = Q^{\pi_\theta}(s_t, a_t)$
- Advantage function $\Phi_t = A^{\pi_\theta}(s_t, a_t)$ where $A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$

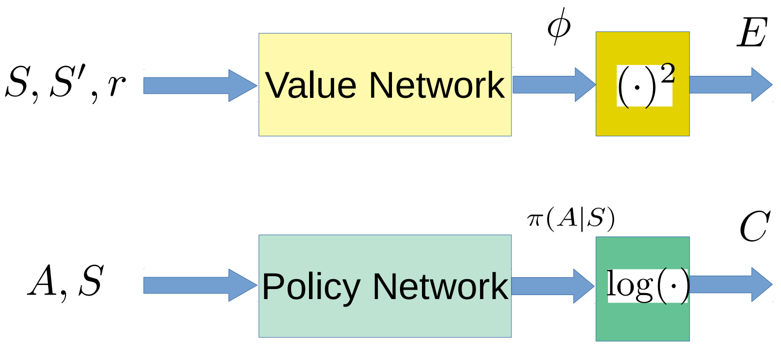
Actor-Critic

- Many variants
- Generally combines policy gradients and value iteration (eg: Q-learning)
- Example: Advantage Actor Critic
  - Policy gradient $$\nabla_\theta J(\pi_\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\Phi_t\right],$$

    Where $\phi_t = A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$
  - Approximate $\phi_t = r + \gamma V^\pi_{\mathbf{w}}(s_{t+1}) - V^\pi_{\mathbf{w}}(s_t)$
  - Two networks
    - Policy network representing π with parameters θ
      - Update using the policy gradients
    - Value network representing V^π with parameters w
      - Update using the gradient of error $\mathrm{E}=\phi_t^2$

Actor-Critic algorithm

- Input: policy network with parameters $\pi_\theta(a_t|s_t)$
- Input: value function with parameters $V_W(s_t)$
- Input: Learning rates η and α
- Initialize policy and value parameters, θ, W
- Loop for each episode:
  - Choose an initial state S
  - Loop while S is not terminal
    - $A \sim \pi(\cdot|S, \theta)$
    - Take action A and observe the next state S' and reward r
    - Calculate $\phi = r + \gamma V_W(S') - V_W(S)$
    - Back-propagate gradient of $E = \phi^2$ through the policy network to get $\nabla_W E$
      - update $W \leftarrow W + \alpha \nabla_W E$
    - Back-propagate $C = \log \pi_\theta(A|S)$ through the policy network to get $\nabla_\theta C$
      - multiply $\nabla_\theta C$ with ϕ to get $\nabla_\theta J = \phi \nabla_\theta \log \pi_\theta(A|S)$
      - update $\theta \leftarrow \theta + \eta \nabla_\theta J$
    - Update $S \leftarrow S'$



Ratio form of policy gradient

- General form of policy gradient

$$\nabla_\theta J(\pi_\theta) = \mathop{\mathrm{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\Phi_t\right],$$

- In practice, we use empirical average over a batch

$$\nabla_\theta J(\pi_\theta) = \mathop{E}_{a_t, s_t\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)\phi_t]$$

- Differentiate log function

$$\nabla_\theta J(\pi_\theta) = \mathop{E}_{a_t, s_t\sim\pi_\theta}[\frac{\nabla_\theta \pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)}\phi_t]$$

- If we evaluate the gradient at a particular θ_k

$$\nabla_\theta J(\pi_\theta)|_{\theta_k} = \mathop{E}_{a_t, s_t\sim\pi_{\theta_k}}[\nabla_\theta \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}\right)\Big|_{\theta_k}\phi_t]$$

- We can back-propagate $\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}\right)\phi_t$ instead of $\log \pi_\theta(a_t|s_t)\phi_t$

Trust Region Policy Optimization (TRPO)

- Solves the problem that parameter updates takes the policy too far.

- Theoretical update equation

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}(\theta_k, \theta)$$
$$\text{s.t. } \bar{D}_{KL}(\theta || \theta_k) \leq \delta$$

where $\mathcal{L}(\theta_k, \theta) = \mathop{\mathrm{E}}_{s,a \sim \pi_{\theta_k}} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a) \right]$ and $\bar{D}_{KL}(\theta || \theta_k) = \mathop{\mathrm{E}}_{s \sim \pi_{\theta_k}} \left[D_{KL}\left(\pi_\theta(\cdot|s) || \pi_{\theta_k}(\cdot|s) \right) \right]$

with θ_k and θ are current and next policy parameters

- Lot of math required to convert to a practical algorithm

Proximal Policy Optimization (PPO)

- Addresses the same problem as TRPO
- However, much simpler math in the practical algorithm
- Probably the most widely used algorithm

- The update rule is

$$\theta_{k+1} = \arg\max_{\theta} \underset{s,a\sim\pi_{\theta_k}}{\mathrm{E}} \left[L(s,a,\theta_k,\theta) \right],$$

with

$$L(s,a,\theta_k,\theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a), \;\; g(\epsilon, A^{\pi_{\theta_k}}(s,a)) \right)$$

where

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A < 0. \end{cases}$$

PPO cases

- Positive advantage

$$L(s, a, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, (1+\epsilon)\right) A^{\pi_{\theta_k}}(s, a).$$

Cannot go above $(1+\epsilon)A^{\pi_{\theta_k}}(s, a).$

- Negative advantage

$$L(s, a, \theta_k, \theta) = \max\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, (1-\epsilon)\right) A^{\pi_{\theta_k}}(s, a)$$

Cannot go below $(1-\epsilon)A^{\pi_{\theta_k}}(s, a).$