PROXIMAL POLICY OPTIMIZATION (PPO)

Presenter: Majid Ghasemi

ECE 750 – October 2024

AGENDA

- Policy Gradient Methods
- TRPO
- PPO
- How to do PPO?
- Applications
- Conclusion

Policy Gradient (REINFORCE)

Algorithm 1 REINFORCE Algorithm

REINFORCE(s_0, π_θ)
Initialize π_θ to anything
while forever (for each episode) **do**
    Generate episode $s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T, a_T, r_T$ with π_θ
    for each step of the episode $n = 0, 1, \ldots, T$ **do**
        $G_n \leftarrow \sum_{t=0}^{T-n} \gamma^t r_{n+t}$
        Update policy: $\theta \leftarrow \theta + \alpha \gamma^n G_n \nabla \log \pi_\theta(a_n|s_n)$
    end for
end while
return π_θ

Update on each batch (trajectory)

$$g = \nabla_\theta J(\pi_\theta) = E_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \log \pi_\theta(a_t | s_t) A^{\pi_\theta}(s_t, a_t) \right]$$

Problems?

Unstable Update

- Step size (*sts*)
- If sts is too large -> bad policy
- Generated batch from a bad policy -> bad samples are collected
- Bad samples -> worse policy

- Sts is too small? -> slow learning process

Problems?

Data Inefficiency

- On-policy method
- The data is thrown out after just one gradient update
- Complex neural networks -> training becomes very slow

Data Inefficiency

- How to make it efficient?

- Avoid sampling from current policy?
- Use previous samples like replay buffer in DQN?

Current Policy $\longleftarrow$ Replay Buffer

New Policy

old data

- Can we estimate an expectation of one distribution without taking samples from it?

YES!

$$E_{x\sim p}[f(x)] \approx \cancel{\frac{1}{N}\sum_{i=1}^{N} f(x^i)}, \quad x^i \sim p$$

$$E_{x\sim p}[f(x)] = \int f(x)p(x)\,dx$$

$$= \int f(x)\frac{p(x)}{q(x)}q(x)\,dx$$

$$= E_{x\sim q}\left[f(x)\frac{p(x)}{q(x)}\right]$$

Importance sampling in Policy Gradient

$$= E_{x\sim q}\left[f(x)\frac{p(x)}{q(x)}\right]$$

New

$$\nabla J(\theta) = E_{(s_t,a_t)\sim\pi_\theta}[\nabla \log \pi_\theta(a_t|s_t)A(s_t,a_t)]$$

IS ratio

old

$$= E_{(s_t,a_t)\sim\pi_{\theta_{\text{old}}}}\left[\boxed{\frac{\pi_\theta(s_t,a_t)}{\pi_{\theta_{\text{old}}}(s_t,a_t)}}\nabla \log \pi_\theta(a_t|s_t)A(s_t,a_t)\right]$$

$$J(\theta) = E_{(s_t,a_t)\sim\pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta(s_t,a_t)}{\pi_{\theta_{\text{old}}}(s_t,a_t)}A(s_t,a_t)\right] \rightarrow \text{Surrogate objective function}$$

Problem?

- Two expectations are same, but we are using sampling method to estimate them -> <u>variance</u>

Close samples? Good to go

Far from each other? High variance

Unstable update

- Confident updates -> New policy ~ Old policy

- Two approaches: Adaptive Learning Rate, Limit the policy update change

How can we measure the distance between two distributions?

KL Divergence

KL Divergence is a tool to measure the distance of two distributions

KL divergence of two policies

$$D_{\mathrm{KL}}(\pi_1||\pi_2)[s] = \sum_{a \in \mathcal{A}} \pi_1(a|s) \log \frac{\pi_1(a|s)}{\pi_2(a|s)}$$

Trust Region Policy Optimization (TRPO)

$$\text{maximize}_\theta \; \hat{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \right] \quad \text{subject to} \quad \hat{E}_t \left[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot|s_t), \pi_\theta(\cdot|s_t)] \right] \leq \delta$$

$$\text{maximize}_\theta \; \hat{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t - \underline{\beta} \, \text{KL}[\pi_{\theta_{\text{old}}}(\cdot|s_t), \pi_\theta(\cdot|s_t)] \right]$$

TRPO uses a hard constraint rather than a penalty because it is hard to choose a single value of β that performs well across different problems—or even within a single problem, where the characteristics change over the course of learning.

Proximal Policy Optimization (PPO)

- **TRPO uses conjugate gradient descent to handle the constraint**
- Hessian Matrix → expensive both in computation and space
- **Idea:**
- The constraint helps in the training process. However, maybe the constraint is not a strict constraint.
- Does it matter if we only break the constraint just a few times?
- What if we treat it as a "**soft**" constraint? Add proximal value to the objective function?

PPO w/ Adaptive KL Penalty

$$L^{KLPEN}(\theta) = \hat{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t - \beta \, \mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot|s_t), \pi_\theta(\cdot|s_t)] \right]$$

Hard to pick β value $\rightarrow$ use adaptive β

Compute $d = \hat{E}_t[\mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot|s_t), \pi_\theta(\cdot|s_t)]]$

– If $d < d_{\text{targ}}/1.5$, $\beta \leftarrow \beta/2$ → Relax

– If $d > d_{\text{targ}} \times 1.5$, $\beta \leftarrow \beta \times 2$ → More Penalty

Algorithm 2 PPO with Adaptive KL Penalty

Input: initial policy parameters θ_0, initial KL penalty β_0, target KL-divergence δ
for $k = 0, 1, 2, \ldots$ **do**
  Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
  Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
  Compute policy update:

$$\theta_{k+1} = \arg\max_{\theta} \mathcal{L}_{\theta_k}(\theta) - \beta_k \overline{D}_{KL}(\theta || \theta_k)$$

  Take K steps of minibatch SGD (via Adam)
  if $\overline{D}_{KL}(\theta_{k+1} || \theta_k) \geq 1.5\delta$ **then**
    $\beta_{k+1} \leftarrow 2\beta_k$
  else if $\overline{D}_{KL}(\theta_{k+1} || \theta_k) \leq \delta / 1.5$ **then**
    $\beta_{k+1} \leftarrow \beta_k / 2$
  end if
end for

PPO with Clipped Objective

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

$$\text{maximize}_\theta \ \hat{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \right]$$

Invariance happens when r changes too quickly $\rightarrow$ limit r within a range?

$$L^{CLIP}(\theta) = \hat{E}_t\left[\min\left(r_t(\theta)\hat{A}_t,\ \mathrm{clip}(r_t(\theta), \overbrace{1-\epsilon}^{\text{too small}}, \underbrace{1+\epsilon}_{\text{too big}})\hat{A}_t\right)\right]$$

Algorithm 3 PPO with Clipped Objective

1: **Input:** initial policy parameters θ_0, clipping threshold ϵ
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: Collect set of partial trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$
4: Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm
5: Compute policy update

$$\theta_{k+1} = \arg\max_{\theta} L_{\theta_k}^{CLIP}(\theta)$$

by taking K steps of minibatch SGD (via Adam), where

$$L_{\theta_k}^{CLIP}(\theta) = E_{\tau \sim \pi_k}\left[\sum_{t=0}^{T} \min\left(r_t(\theta)\hat{A}_t^{\pi_k},\ \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t^{\pi_k}\right)\right]$$

6: **end for**

Practical Guide on PPO

Components of PPO

- Policy Network
- Value Function

Policy Network

State -> Neural Networks -> Probability o taking actions in that state

Value Function

State -> Neural Networks -> Q(s, a1)

Q(s, a2)

Q(s, a3)

Q(s, a4)

Applications?

- RLHF
- Discrete & Continuous states space
- You name some?