

# **Advanced On-Policy RL**

**Hao Su**

**(slides prepared with the help from Shuang Liu)**

**Spring, 2021**

# Agenda

- Practical First-Order Policy Optimization
- Efficient and Stable Policy Optimization

click to jump to the section.

# Review: Policy Gradient Theorem (Discounted)

- Policy Gradient Theorem (Discounted):

$$\nabla_{\theta} V^{\pi_{\theta}, \gamma}(s_0) = \sum_s \sum_{t=0}^{\infty} \gamma^t \mu_t(s; s_0) \sum_a \nabla_{\theta} \pi_{\theta}(s, a) \cdot Q^{\pi_{\theta}, \gamma}(s, a).$$

$\mu_t(s; s_0)$  is the average visitation frequency of the state  $s$  in step  $k$ .

- Can you guess the influence of  $\gamma$  in this result?

**We will assume the discounted setting from now on.**

# Review: Creating an Unbiased Estimate for PG

We have shown that

$$\nabla_{\theta} V^{\pi_{\theta}, \gamma}(s_0) = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t \nabla_{\theta} \ln(\pi_{\theta}(s_t, a_t)) \cdot \sum_{i=t}^{\infty} \gamma^{i-t} \cdot r_i \right]$$

- Using more trajectories, we can get more accurate gradient estimate (smaller variance)
- Since the unbiased estimate is a summation, we can sample from the individual terms to do batched gradient descent

**We have established an MC sampling based method to estimate the gradient of value w.r.t. policy parameters!**

**This estimate is *unbiased*.**

- In literature, this MC-sampling based policy gradient method is called **REINFORCE**.

# REINFORCE Algorithm

The steps involved in the implementation of REINFORCE would be as follows:

1. Randomly initialize a policy network that takes the state as input and returns the probability of actions
2. Use the policy to play  $n$  episodes of the game. Record  $(s, a, s', r)$  for each step.
3. Calculate the discounted reward for each step backwards
4. Calculate expected reward  $G_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_i$
5. Adjust weights of policy according to the gradient by policy gradient theorem
6. Repeat from 2

<https://www.analyticsvidhya.com/blog/2020/11/reinforce-algorithm-taking-baby-steps-in-reinforcement-learning/>

# Practical First-Order Policy Optimization

# Advanced Value Estimates

We have seen that we can use  $\sum_{i=t}^{\infty} \gamma^{i-t} \cdot r_i$  as an unbiased estimate for  $Q^{\pi_\theta, \gamma}(s_t, a_t)$ .

We can also have a value network  $v_\omega(s)$  to try to **memorize** (the estimates of)  $V^{\pi_\theta, \gamma}(s)$  during the training. This way, whenever we need an estimate of  $Q^{\pi_\theta, h}(s_t, a_t)$ , we can use

# Advanced Value Estimates

We have seen that we can use  $\sum_{i=t}^{\infty} \gamma^{i-t} \cdot r_i$  as an unbiased estimate for  $Q^{\pi_\theta, \gamma}(s_t, a_t)$ .

We can also have a value network  $v_\omega(s)$  to try to **memorize** (the estimates of)  $V^{\pi_\theta, \gamma}(s)$  during the training. This way, whenever we need an estimate of  $Q^{\pi_\theta, h}(s_t, a_t)$ , we can use

- $e_\infty = \sum_{i=t}^{\infty} \gamma^{i-t} \cdot r_i$ , which is unbiased but has high variance.

# Advanced Value Estimates

We have seen that we can use  $\sum_{i=t}^{\infty} \gamma^{i-t} \cdot r_i$  as an unbiased estimate for  $Q^{\pi_\theta, \gamma}(s_t, a_t)$ .

We can also have a value network  $v_\omega(s)$  to try to **memorize** (the estimates of)  $V^{\pi_\theta, \gamma}(s)$  during the training. This way, whenever we need an estimate of  $Q^{\pi_\theta, h}(s_t, a_t)$ , we can use

- $e_\infty = \sum_{i=t}^{\infty} \gamma^{i-t} \cdot r_i$ , which is unbiased but has high variance.
- $e_{t,0} = r_t + \gamma \cdot v_\omega(s_{t+1})$ , which is biased but possibly has lower variance.

# Advanced Value Estimates

We have seen that we can use  $\sum_{i=t}^{\infty} \gamma^{i-t} \cdot r_i$  as an unbiased estimate for  $Q^{\pi_\theta, \gamma}(s_t, a_t)$ .

We can also have a value network  $v_\omega(s)$  to try to **memorize** (the estimates of)  $V^{\pi_\theta, \gamma}(s)$  during the training. This way, whenever we need an estimate of  $Q^{\pi_\theta, h}(s_t, a_t)$ , we can use

- $e_\infty = \sum_{i=t}^{\infty} \gamma^{i-t} \cdot r_i$ , which is unbiased but has high variance.
- $e_{t,0} = r_t + \gamma \cdot v_\omega(s_{t+1})$ , which is biased but possibly has lower variance.
- $e_{t,h} = \sum_{i=t}^{t+h} \gamma^{i-t} \cdot r_i + \gamma^{h+1} \cdot v_\omega(s_{t+h+1})$ , which has a trade-off between the first two, depending on the choice of  $h$ .

# Advanced Value Estimates

We have seen that we can use  $\sum_{i=t}^{\infty} \gamma^{i-t} \cdot r_i$  as an unbiased estimate for  $Q^{\pi_\theta, \gamma}(s_t, a_t)$ .

We can also have a value network  $v_\omega(s)$  to try to **memorize** (the estimates of)  $V^{\pi_\theta, \gamma}(s)$  during the training. This way, whenever we need an estimate of  $Q^{\pi_\theta, h}(s_t, a_t)$ , we can use

- $e_\infty = \sum_{i=t}^{\infty} \gamma^{i-t} \cdot r_i$ , which is unbiased but has high variance.
- $e_{t,0} = r_t + \gamma \cdot v_\omega(s_{t+1})$ , which is biased but possibly has lower variance.
- $e_{t,h} = \sum_{i=t}^{t+h} \gamma^{i-t} \cdot r_i + \gamma^{h+1} \cdot v_\omega(s_{t+h+1})$ , which has a trade-off between the first two, depending on the choice of  $h$ .
- $\sum_{h=0}^{\infty} \alpha_h e_{t,h}$ , further combines different  $e_h$ 's with tunable weights  $\alpha_h$ 's that summing to 1.

# Advantage

- We introduce another statistics trick to reduce the variance of the value estimation.

# Advantage

- We introduce another statistics trick to reduce the variance of the value estimation.
- Suppose that  $X$  and  $Y$  are two random variables.
- Recall that, if  $Z = X - Y$ , then

$$\text{Var}[Z] = \text{Var}[X] + \text{Var}[Y] - 2\text{Cov}(X, Y)$$

- If  $X$  and  $Y$  are strongly correlated, then  $\text{Var}[Z]$  is smaller than  $\text{Var}[X]$  and  $\text{Var}[Y]$ .
  - For example,  $X, Y \sim \mathbb{N}(0, 1)$ , then  $\text{Var}[Z] = 2 - 2\rho$ , where  $\rho$  is the Pearson correlation coefficient and  $-1 \leq \rho \leq 1$ . For highly correlated  $X$  and  $Y$ ,  $\rho \approx 1$ , and  $\text{Var}[Z] \approx 0$ .

# Advantage

- We introduce another statistics trick to reduce the variance of the value estimation.
- Suppose that  $X$  and  $Y$  are two random variables.
- Recall that, if  $Z = X - Y$ , then

$$\text{Var}[Z] = \text{Var}[X] + \text{Var}[Y] - 2\text{Cov}(X, Y)$$

- If  $X$  and  $Y$  are strongly correlated, then  $\text{Var}[Z]$  is smaller than  $\text{Var}[X]$  and  $\text{Var}[Y]$ .
  - For example,  $X, Y \sim \mathbb{N}(0, 1)$ , then  $\text{Var}[Z] = 2 - 2\rho$ , where  $\rho$  is the Pearson correlation coefficient and  $-1 \leq \rho \leq 1$ . For highly correlated  $X$  and  $Y$ ,  $\rho \approx 1$ , and  $\text{Var}[Z] \approx 0$ .
- For this reason, we introduce the function  $A^{\pi_\theta, \gamma}(s, a) = Q^{\pi_\theta, \gamma}(s, a) - V^{\pi_\theta, \gamma}(s)$ , which is called **advantage**.
- Our next goal is to relate  $\nabla V^{\pi_\theta, \gamma}(s, a)$  with  $A^{\pi_\theta, \gamma}(s, a)$ , which has smaller variance than estimating through  $Q^{\pi_\theta, \gamma}(s, a)$ .

# Advantage Estimates

$$\begin{aligned}\nabla_{\theta} V^{\pi_{\theta}, \gamma}(s_0) &= \sum_s \sum_{t=0}^{\infty} \gamma^t \mu_t(s; s_0) \left( \sum_a \nabla_{\theta} \pi_{\theta}(s, a) \cdot Q^{\pi_{\theta}, \gamma}(s, a) - 0 \right) \\ (\text{why?}) &= \sum_s \sum_{t=0}^{\infty} \gamma^t \mu_t(s; s_0) \left( \sum_a \nabla_{\theta} \pi_{\theta}(s, a) \cdot Q^{\pi_{\theta}, \gamma}(s, a) - \sum_a \nabla_{\theta} \pi_{\theta}(s, a) \cdot V^{\pi_{\theta}, \gamma}(s) \right) \\ &= \sum_s \sum_{t=0}^{\infty} \gamma^t \mu_t(s; s_0) \sum_a \nabla_{\theta} \pi_{\theta}(s, a) \cdot (Q^{\pi_{\theta}, \gamma}(s, a) - V^{\pi_{\theta}, \gamma}(s)) . \\ &= \sum_s \sum_{t=0}^{\infty} \gamma^t \mu_t(s; s_0) \sum_a \nabla_{\theta} \ln(\pi_{\theta}(s, a)) \cdot \pi_{\theta}(s, a) A^{\pi_{\theta}, \gamma}(s, a) \\ &= \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t \nabla_{\theta} \ln(\pi_{\theta}(s_t, a_t)) \cdot A^{\pi_{\theta}, \gamma}(s_t, a_t) \right]\end{aligned}$$

- Q: Does this form look reasonable? Compare it with the update for REINFORCE.

# Advantage Estimates (Cont'd)

- Recall that we said  $Q^{\pi_\theta, \gamma}(s_t, a_t)$  can be estimated by  $\sum_{h=0}^{\infty} \alpha_h e_{t,h}$  in general, where

$$\begin{cases} e_{t,h} = \sum_{i=t}^{t+h} \gamma^{i-t} \cdot r_i + \gamma^{h+1} \cdot v_\omega(s_{t+h+1}) \\ \sum_{i=0}^{\infty} \alpha_i = 1 \end{cases}$$

- The very popular **General Advantage Estimate** (GAE) estimates the advantage in the same fashion and it chooses  $\alpha_i$  to be proportional to  $\lambda^i$ , where  $\lambda \in [0, 1]$ .

# Advantage Estimates (Cont'd)

- Define  $\delta_{t,h} = e_{t,h} - v_\omega(s_t) = \sum_{i=t}^{t+h} \gamma^{i-t} \cdot r_i + \gamma^{h+1} \cdot v_\omega(s_{t+h+1}) - v_\omega(s_t)$
- The General Advantage Estimate (GAE) estimates  $A^{\pi_\theta, \gamma}(s_t, a_t)$  by

$$\hat{A}_{\text{GAE}(\lambda)}^{\pi_\theta, \gamma}(s_t, a_t) = (1 - \lambda) \sum_{h=0}^{\infty} \lambda^h \delta_{t,h}$$

(calculation omitted, HW)  $= \sum_{h=0}^{\infty} (\gamma \lambda)^h \delta_{t+h,0}$

- Define  $0^0 = 1$ ,  $\hat{A}_{\text{GAE}(0)}^{\pi_\theta, \gamma}(s_t, a_t) = r_t + \gamma v_\omega(s_{t+1}) - v_\omega(s_t)$ .
- We also have  $\hat{A}_{\text{GAE}(1)}^{\pi_\theta, \gamma}(s_t, a_t) = \sum_{h=0}^{\infty} \gamma^h r_{t+h} - v_\omega(s_t)$ .
- We leave the proofs to homework.
- Q: How to interpret the role of  $\lambda$ ?

# Incremental Monte Carlo

## Value Function Estimation

- Recall that our plan was to have a value network  $v_\omega(s)$  to **memorize** the estimates of  $V^{\pi_\theta, \gamma}(s)$  during the training.
- In practice, certain variants of the incremental Monte-Carlo method will be used to update  $v_\omega(s)$ 
  - e.g., in Sec 5 of High-Dimensional Continuous Control Using Generalized Advantage Estimation,

$$\begin{aligned} & \text{minimize}_\omega && \sum_{n=1}^N \|v_\omega(s_n) - \hat{V}_n\|^2 \\ & \text{subject to} && \frac{1}{N} \sum_{n=1}^N \frac{\|v_\omega(s_n) - v_{\omega_{old}}(s_n)\|^2}{2\sigma^2} \leq \epsilon \end{aligned}$$

where  $\hat{V}_t = \sum_{h=0}^{\infty} \gamma^h r_{t+h}$  is the discounted sum of rewards, and  $n$  indexes over all timestamps in a batch of trajectories.

- Note that we do not need a replay buffer to estimate  $v_\omega(s)$ .

# Some Additional Information

Given any advantage estimate  $\hat{A}^{\pi_{\theta}, \gamma}(s_t, a_t)$ , we can estimate the policy gradient by

$$\hat{\nabla}_{\theta} V^{\pi_{\theta}, \gamma}(s_0) = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t \nabla_{\theta} \ln(\pi_{\theta}(s_t, a_t)) \cdot \hat{A}^{\pi_{\theta}, \gamma}(s_t, a_t) \right].$$

However, in most implementations, people simply use

$$\hat{\nabla}_{\theta} V^{\pi_{\theta}, \gamma}(s_0) = \mathbb{E} \left[ \sum_{t=0}^{\infty} \nabla_{\theta} \ln(\pi_{\theta}(s_t, a_t)) \cdot \hat{A}^{\pi_{\theta}, \gamma}(s_t, a_t) \right].$$

# **Efficient and Stable Policy Optimization**

# On-Policy RL vs. Off-Policy RL

$$\hat{\nabla}_\theta V^{\pi_\theta, \gamma}(s_0) = \mathbb{E}_{\pi_\theta} \left[ \sum_{t=0}^{\infty} \nabla_\theta \ln(\pi_\theta(s_t, a_t)) \cdot \hat{A}^{\pi_\theta, \gamma}(s_t, a_t) \right]. \quad (\text{PGT})$$

vs.

$$\nabla_\theta L(\theta) = \mathbb{E}_{(s, a, s') \sim \text{ReplayBuffer}} [\nabla_\theta \|Q_\theta(s, a) - [R(s, a, s') + \gamma \max_{a'} Q_\theta(s', a')] \|^2] \quad (\text{TD})$$

- On-policy RL:
  - To use PGT, we need *a trajectory under the current policy* to compute the gradient. RL of this kind is called **on-policy**.
  - We must sample actions by the current policy and interact with the environment until the end of an episode. If we revise the policy, we must resample actions and *interact with the environment*.
- Off-policy RL: To use Bellman optimality equation, we sample with distribution *NOT* as the current policy.

# Make Better Use of Samples for On-Policy RL

- Rollouts are precious. It is natural to ask, *how can we make the best use of the recent rollouts?*
- Some straight-forward solutions:
  - using big step size;
  - multiple gradient descents on the same set of rollouts;
- However, PGT only gives a *local approximation* of gradient. Above approaches cause instability in policy updates.

# Trust-region Method

$$\underset{x}{\text{minimize}} \quad f(x)$$

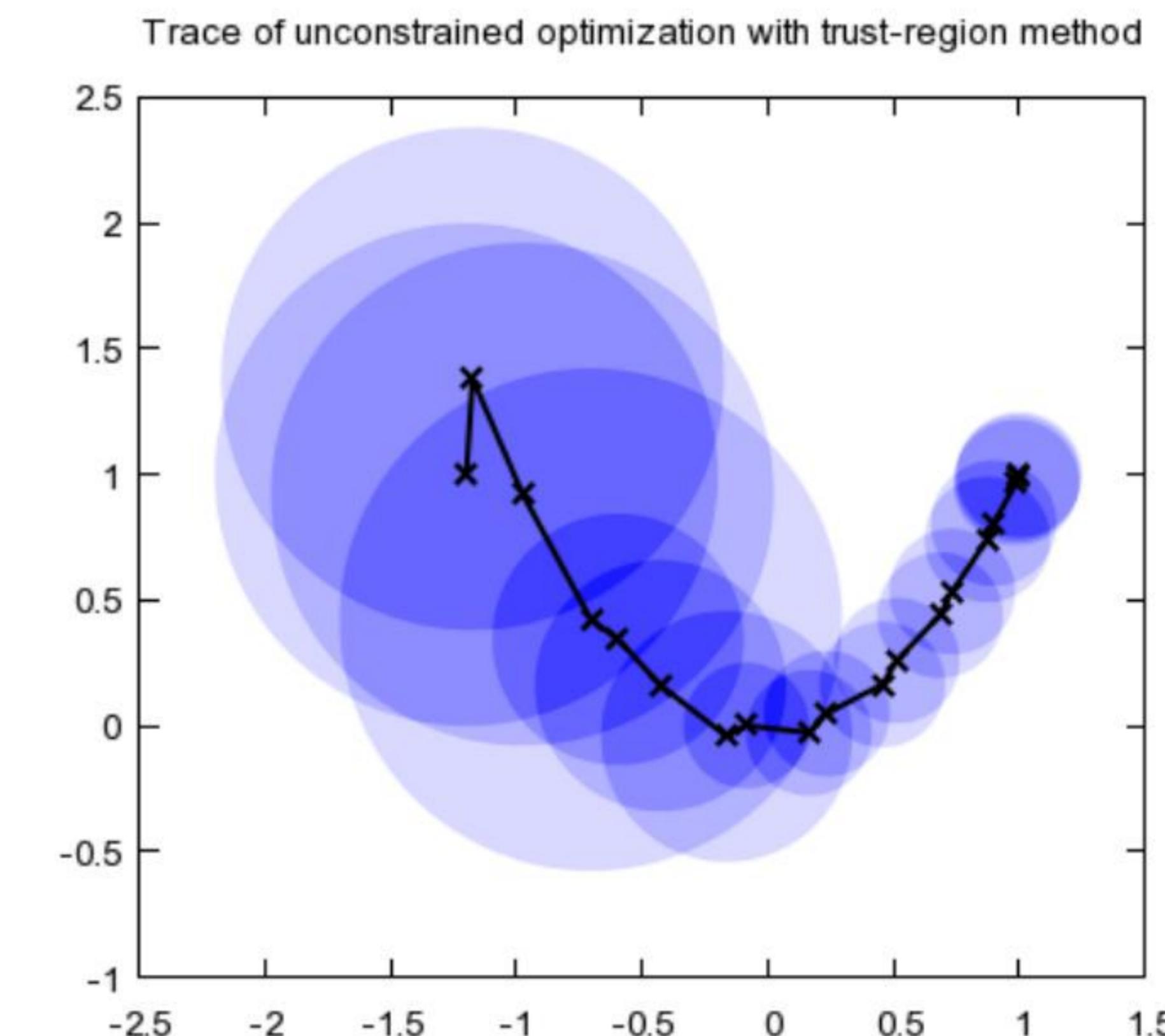
- Iterate:

- Solve a constrained sub-problem

$$\begin{aligned} & \underset{x}{\text{minimize}} && \tilde{f}_{x_k}(x) \\ & \text{subject to} && D(x, x_k) \leq \epsilon \end{aligned}$$

- $x_{k+1} = x; k \leftarrow k + 1$

$\tilde{f}_{x_k}(x)$  is a local approximation of  $f$  near  $x_k$  (e.g., linear or quadratic), and  $D(\cdot, x_k) \leq \epsilon$  restricts the next step  $x$  to be in a local region of  $x_k$ .



[https://optimization.mccormick.northwestern.edu/index.php/Trust-region\\_methods](https://optimization.mccormick.northwestern.edu/index.php/Trust-region_methods)

We compute the gradient (and Hessian) of  $f$  **for once**, but we can use it to update  $x$  **for multiple steps** safely!

# Basic Framework of Trust-region Policy Optimization (e.g, TRPO/PPO)

TRPO/PPO repeat the following procedure:

- Sample multiple (say, 128) trajectories of certain length (say, 128) to get a minibatch of state-actions pairs (say,  $128 * 128$   $(s, a)$  pairs)
- Estimate the advantage of each  $(s, a)$  pair using GAE (say GAE(0.95)) and the corresponding trajectory
- Solve the (mini-batch) local subproblem multiple times (say, mini-batch of size  $128 * 32$ , 16 gradient descents)

# Trust-region Method for Policy Optimization

- Our policy gradient theorem gives us  $\nabla_{\theta} V^{\pi_{\theta}, \gamma}(s_0)$ ; however, to apply trust-region method, we need an optimization form.
- Recall the form of PGT:

$$\hat{\nabla}_{\theta} V^{\pi_{\theta}, \gamma}(s_0) = \mathbb{E}_{\pi_{\theta}} \left[ \sum_{t=0}^{\infty} \nabla_{\theta} \ln(\pi_{\theta}(s_t, a_t)) \cdot \hat{A}^{\pi_{\theta}, \gamma}(s_t, a_t) \right].$$

- Our idea to derive a trust-region based method algorithm. Directly computing the first-order expansion of  $V^{\pi_{\theta}, \gamma}$  is hard, instead,

*we find a series of surrogate objective functions  $\ell_{\theta_k}(\theta)$   
whose gradient is the same as the gradient from PGT **at each step**.*

# Trust-region Method for Policy Optimization

Objective:

- Consider two surrogate loss functions

$$\ell_{\theta_k}^1(\theta) = \mathbb{E}_{\pi_{\theta_k}} \sum_{t=0}^{\infty} \frac{\pi_{\theta}(s_t, a_t)}{\pi_{\theta_k}(s_t, a_t)} A^{\pi_{\theta_k}, \gamma}(s_t, a_t) \quad (\text{ratio loss})$$

$$\ell_{\theta_k}^2(\theta) = \mathbb{E}_{\pi_{\theta_k}} \sum_{t=0}^{\infty} \ln \pi_{\theta}(s_t, a_t) A^{\pi_{\theta_k}, \gamma}(s_t, a_t) \quad (\text{log ratio loss})$$

- It is easy to verify that

$$\nabla_{\theta} \ell_{\theta_k}^1(\theta)|_{\theta=\theta_k} = \nabla_{\theta} \ell_{\theta_k}^2(\theta)|_{\theta=\theta_k} \equiv \nabla_{\theta} V^{\pi_{\theta}, \gamma}(s_0)|_{\theta=\theta_k}$$

- TRPO (Trust-region Policy Optimization) method picks the ratio loss as the objective.

# Trust-region Method for Policy Optimization

The local constrained subproblem in TRPO:

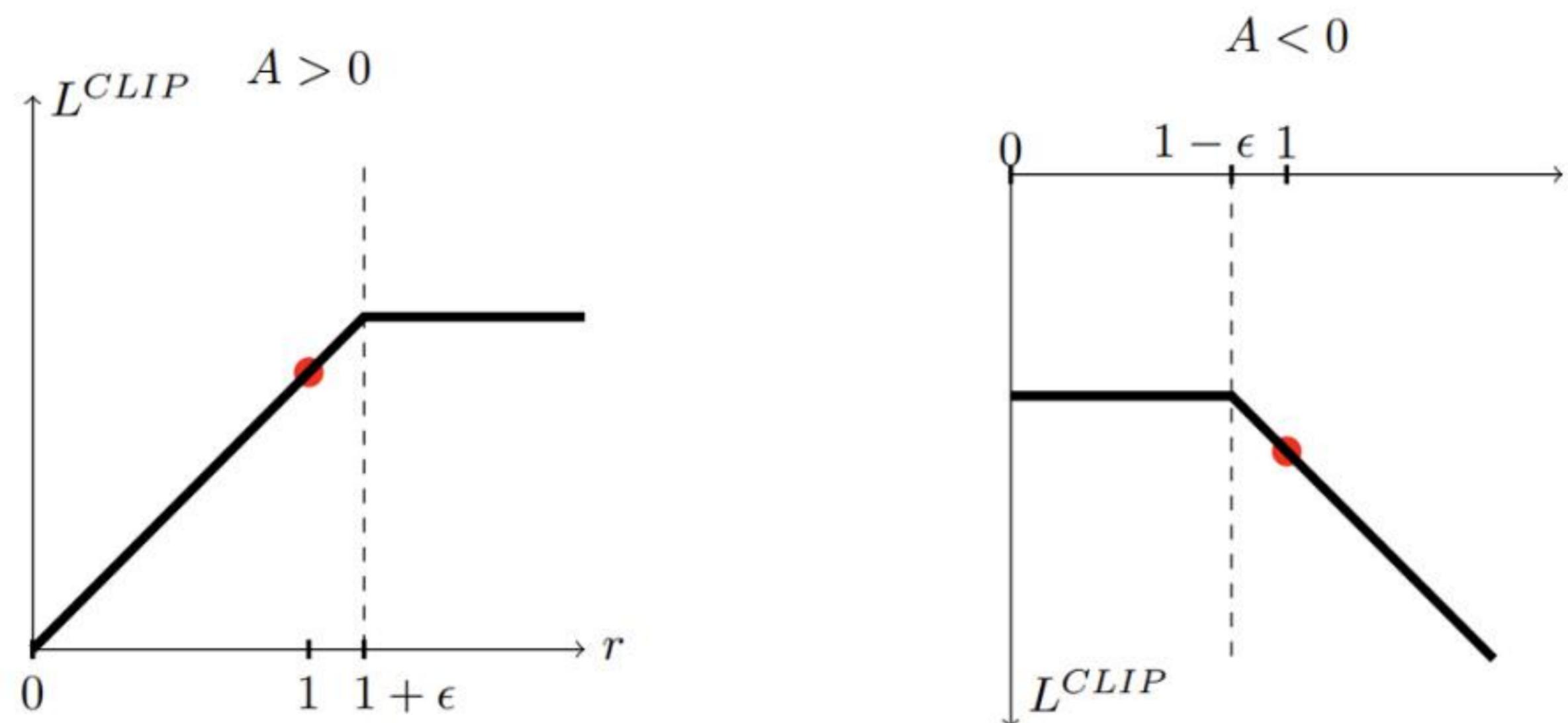
$$\begin{aligned} & \underset{\theta}{\text{maximize}} && \mathbb{E}_{\pi_{\theta_k}} \sum_{t=0}^{\infty} \frac{\pi_{\theta}(s_t, a_t)}{\pi_{\theta_k}(s_t, a_t)} A^{\pi_{\theta_k}, \gamma}(s_t, a_t) \\ & \text{subject to} && \mathbb{E}_{\pi_{\theta_k}} [\text{KL}(\pi_{\theta_k}(s, \cdot) \| \pi_{\theta}(s, \cdot))] \leq \delta \end{aligned}$$

The constraint restricts that  $\pi_{\theta}$  not deviates much from  $\pi_{\theta_k}$ .

- *Q: What is the intuitive explanation of this objective function?*

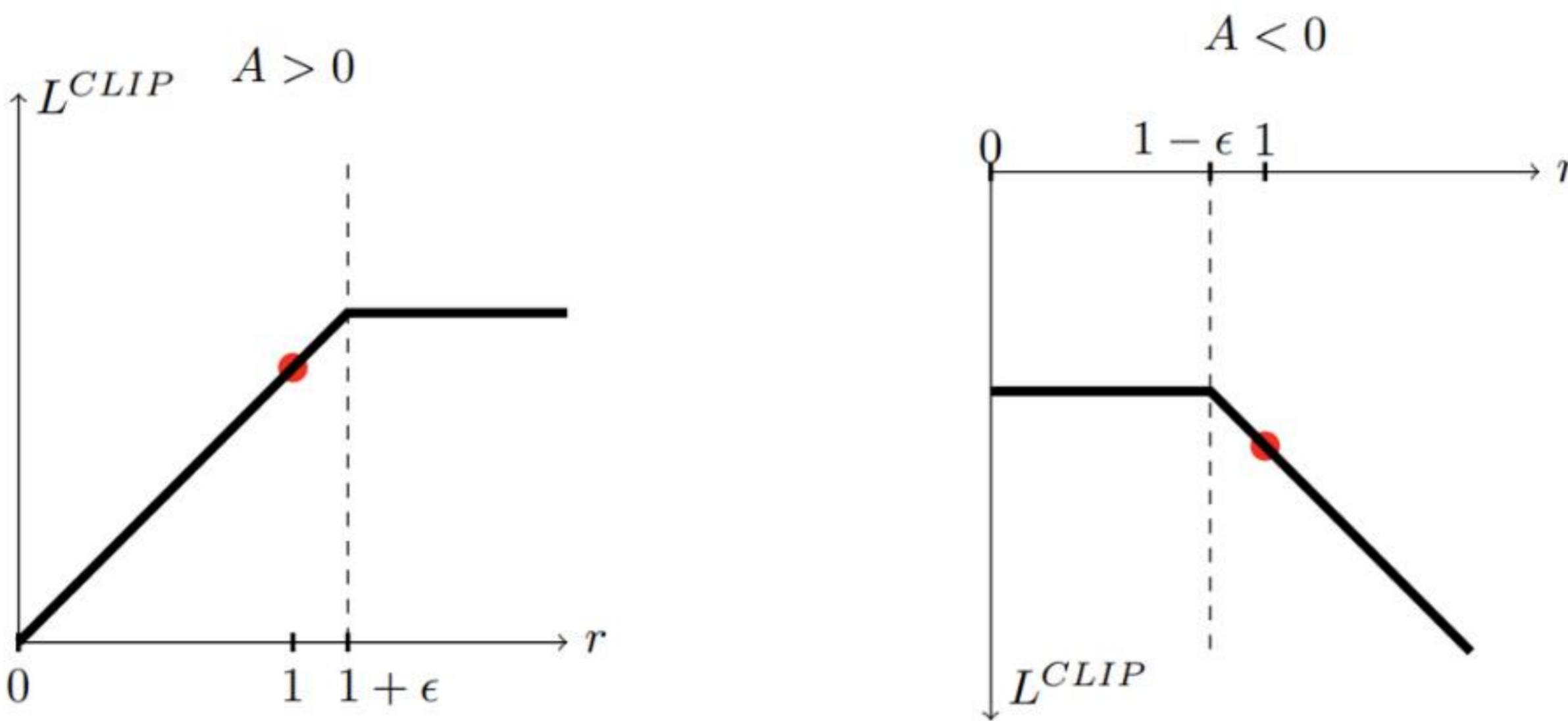
# Constrained Opt. → Unconstrained Opt.

- In TRPO, the subproblem to solve is a **constrained** optimization problem. Dealing with constraints involves tricks.
- PPO uses a unconstrained optimization problem to approximate the constrained problem.
- Let  $r_{t,k}(\theta; s, a) = \frac{\pi_\theta(s_t, a_t)}{\pi_{\theta_k}(s_t, a_t)}$ , we massage the original objective  $\mathbb{E}[rA]$  to a new objective with the following behavior:



Note: For simplicity, dependencies on  $(s, a)$  are omitted.

# Proximal Policy Optimization (PPO)



- The following objective function has the desired graph:

$$f(r, A) = \begin{cases} \text{clip}(r, -\infty, 1 + \epsilon)A, & A > 0 \\ \text{clip}(r, 1 - \epsilon, \infty)A, & A \leq 0 \end{cases}$$

# Proximal Policy Optimization (PPO)

- The Proximal Policy Optimization method solves the unconstrained subproblem to improve the policy.

$$\underset{\theta}{\text{maximize}} \quad \mathbb{E}_{\pi_{\theta_k}} \sum_{t=0}^{\infty} f(r(\theta; s_t, a_t), A(s_t, a_t))$$

- In the original paper, the objective is written in an equivalent form:

$$\underset{\theta}{\text{maximize}} \quad \mathbb{E}_{\pi_{\theta_k}} \sum_{t=0}^{\infty} [\min (r_{t,k}(\theta) A^{\pi_{\theta_k}, \gamma}(s_t, a_t), \text{clip}(r_{t,k}(\theta), 1 - \epsilon, 1 + \epsilon) A^{\pi_{\theta_k}, \gamma}(s_t, a_t))]$$

# Exploration

- The family of policy gradient algorithms also need to implement **exploration**.
- Since the policy function is a distribution over actions, TRPO and PPO randomly sample actions according to the policy function. This random behavior is already an effective exploration strategy.
  - e.g., for discrete action space, we use softmax to output the action probability;
  - for continuous action space, we predict the mean and variance of a Gaussian, which allows us to compute the action probability and sample actions.

# Exploration

- Improved exploration by entropy regularizer:

$$\underset{\theta}{\text{maximize}} \quad \mathbb{E}_{\pi_{\theta_k}} \sum_{t=0}^{\infty} f(r(\theta; s_t, a_t), A(s_t, a_t)) + \eta \cdot \text{entropy}(\pi_{\theta'}(s_t, \cdot))$$

- Entropy:
  - Discrete distribution  $P$ :  $\text{entropy}(P) = - \sum_x p(x) \log p(x)$
  - Continuous distribution  $P$  with density  $f$ :  $\text{entropy}(P) = - \int_x f(x) \log f(x) + \infty$

**End**