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 (Undiscounted):

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

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

- Can you guess the influence of γ in this result?

We will assume the discounted setting from now on.

Review: Creating an Unbiased Estimate for PG

We have shown that

$$\nabla_\theta V^{\pi_\theta,\gamma}(\boldsymbol{s}_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \nabla_\theta \ln(\pi_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t)) \cdot \sum_{i=t}^{\infty} \gamma^{i-k} \cdot \boldsymbol{r}_i\right]$$

- 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**.

Practical First-Order Policy Optimization

Advanced Value Estimates

We have seen that we can use $\sum_{i=h}^{\infty} \gamma^{i-h} \cdot r_i$ 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 (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

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

Advantage Estimates

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

$Q^{\pi_\theta,\gamma}(s,a) - V^{\pi_\theta,\gamma}(s)$ is called **advantage**, which is typically denoted as $A^{\pi_\theta,\gamma}(s,a)$. In fact, the same derivation works if we replace $V^{\pi_\theta,\gamma}(s)$ by any quantity that depends only on s (e.g., 0, in our original derivation).

Advantage Estimates (Cont'd)

With the new representation of the policy gradient, we can now derive a new estimate of the policy gradient

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

Without resampling, we cannot unbiasedly estimate the advantage, fortunately we can still use a value network.

Advantage Estimates (Cont'd)

Recall that we said $Q^{\pi_\theta,\gamma}(s_t, a_t)$ can be estiamted by $\sum_{i=h}^{\infty} \alpha_i e_i$ in general, where

$$\sum_{i=h}^{\infty} \alpha_i = 1$$

and

$$e_t = \sum_{i=h}^{t} \gamma^{i-h} \cdot r_i + \gamma^{k-h+1} \cdot v_\omega(s_{k+1}).$$

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

Advantage Estimates (Cont'd)

That is, the General Advantage Estimate (GAE) estimate $A^{\pi_\theta,\gamma}(s_t, a_t)$ by

$$\hat{A}^{\pi_\theta,\gamma}_{\mathrm{GAE}(\lambda)}(s_t, a_t) = (1-\lambda)\sum_{t=h}^{\infty}\lambda^{k-h}\left(\sum_{i=h}^{t}\gamma^{i-h}\cdot r_i + \gamma^{k-h+1}\cdot v_\omega(s_{k+1}) - v_\omega(s_h)\right)$$

$$\text{(calculation omitted)} = \sum_{t=h}^{\infty}(\gamma\lambda)^{k-h}\left(r_t + \gamma v_\omega(s_{k+1}) - v_\omega(s_k)\right)$$

Define $0^0 = 1$, we have

$$\hat{A}^{\pi_\theta,\gamma}_{\mathrm{GAE}(0)}(s_t, a_t) = r_t + \gamma v_\omega(s_{h+1}) - v_\omega(s_h).$$

We also have

$$\hat{A}^{\pi_\theta,\gamma}_{\mathrm{GAE}(1)}(s_t, a_t) = \sum_{i=h}^{\infty}\gamma^{i-h} r_i - v_\omega(s_t).$$

Some Additional Notes

- 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

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

- Now that we know how to estimate policy gradients, any method/trick that can be applied to general first-order optimization can in principle to be used for policy optimization.

Efficient and Stable Policy Optimization

On-Policy RL vs. Off-Policy RL

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

vs.

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

- 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

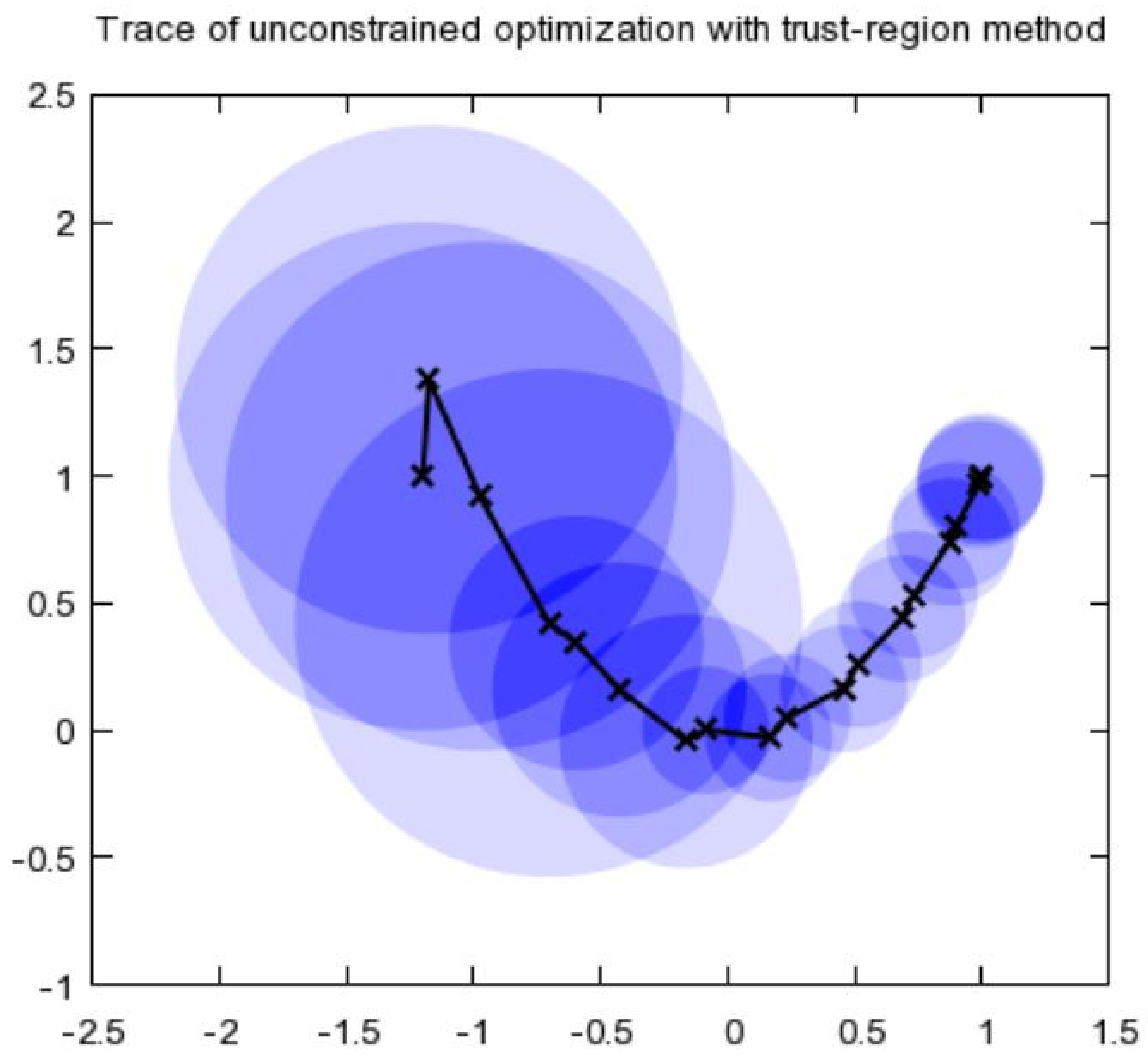
$$\underset{x}{\text{minimize}} \qquad f(x)$$

- Iterate:
  - ○ Solve a constrained sub-problem

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

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

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



https://optimization.mccormick.northwestern.edu/index.php/Trust-region_methods

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

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

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

- It is easy to verify that

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

- 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:

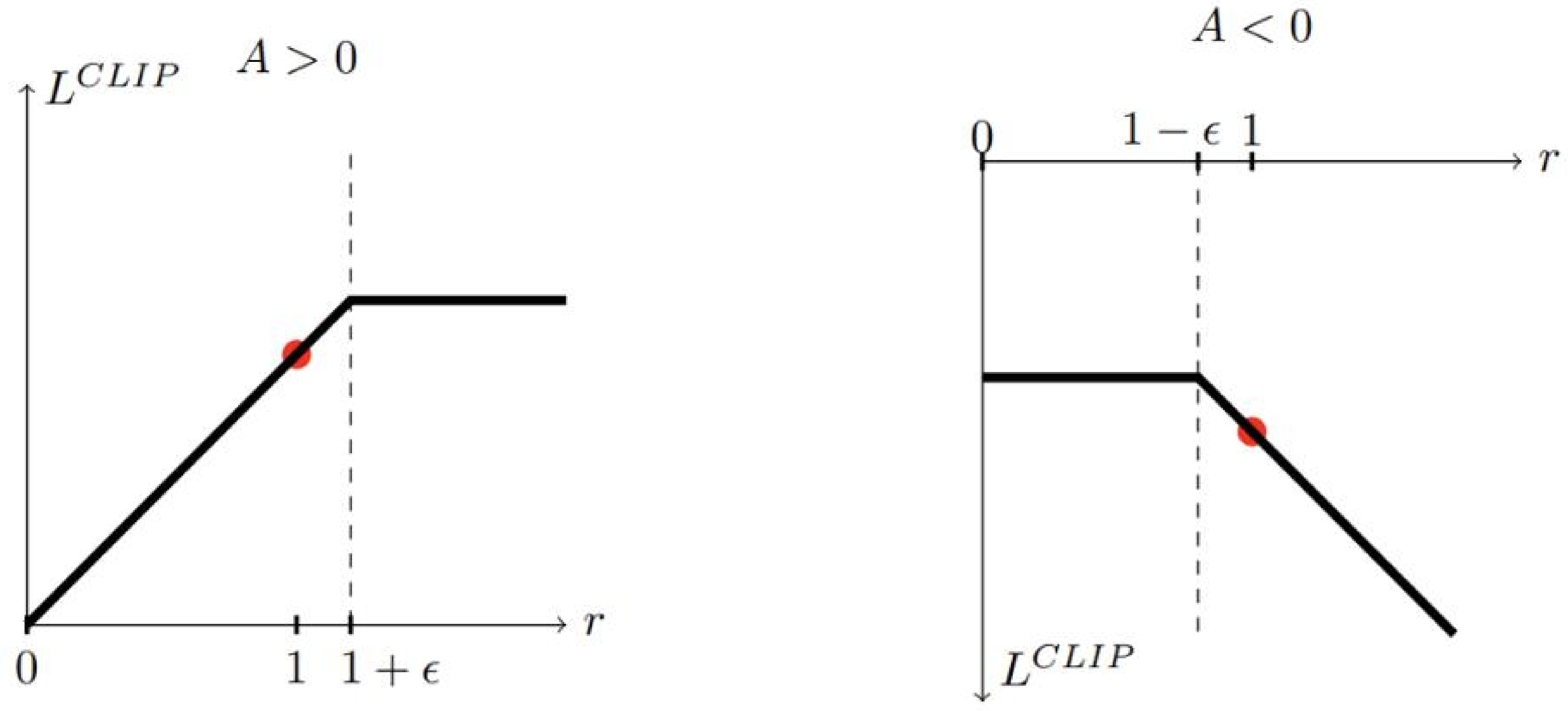
$$\begin{aligned} \underset{\theta}{\text{maximize}} \qquad & \mathbb{E}_{\pi_{\theta_k}} \sum_{t=0}^{\infty} \frac{\pi_\theta(s_t, a_t)}{\pi_{\theta_k}(s_t, a_t)} A^{\pi_{\theta_k}, \gamma}(s_t, a_t) \\ \text{subject to} \qquad & \mathbb{E}_{\pi_{\theta_k}} [\mathrm{KL}(\pi_{\theta_k}(s, \cdot) \| \pi_\theta(s,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

The constraint restricts that π_θ not deviates much from π_{θ_k}.

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

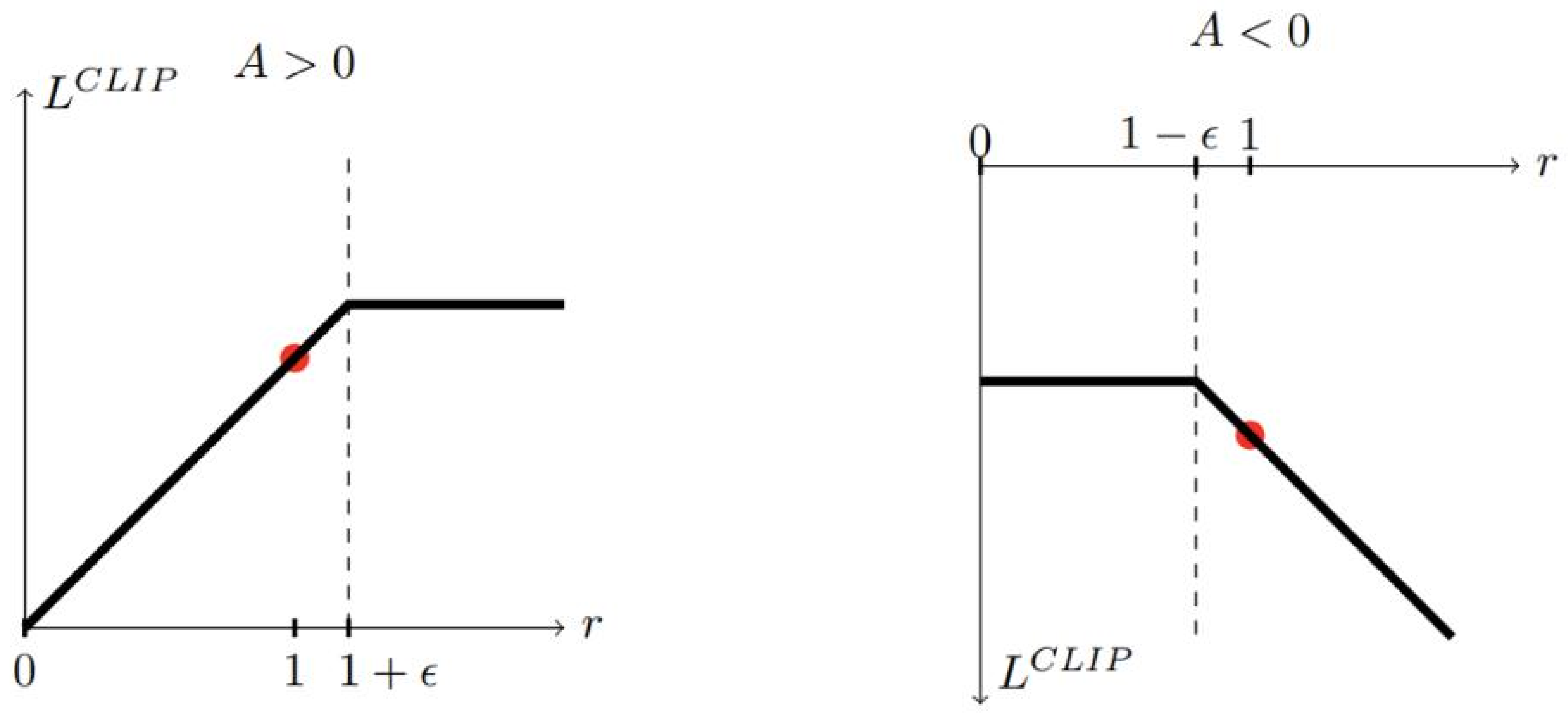
Constrained Opt. → Unconstrained Opt.

- 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.

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

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

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

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.
- Improved exploration by entropy regularizer:

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

- 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