Deep Reinforcement Learning

Lecture 9: Advanced Policy Gradient Methods

Instructor: Chongjie Zhang

Tsinghua University

Policy Gradients

Monte Carlo Policy Gradients (REINFORCE), gradient direction: $\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \hat{A}_t \right]$

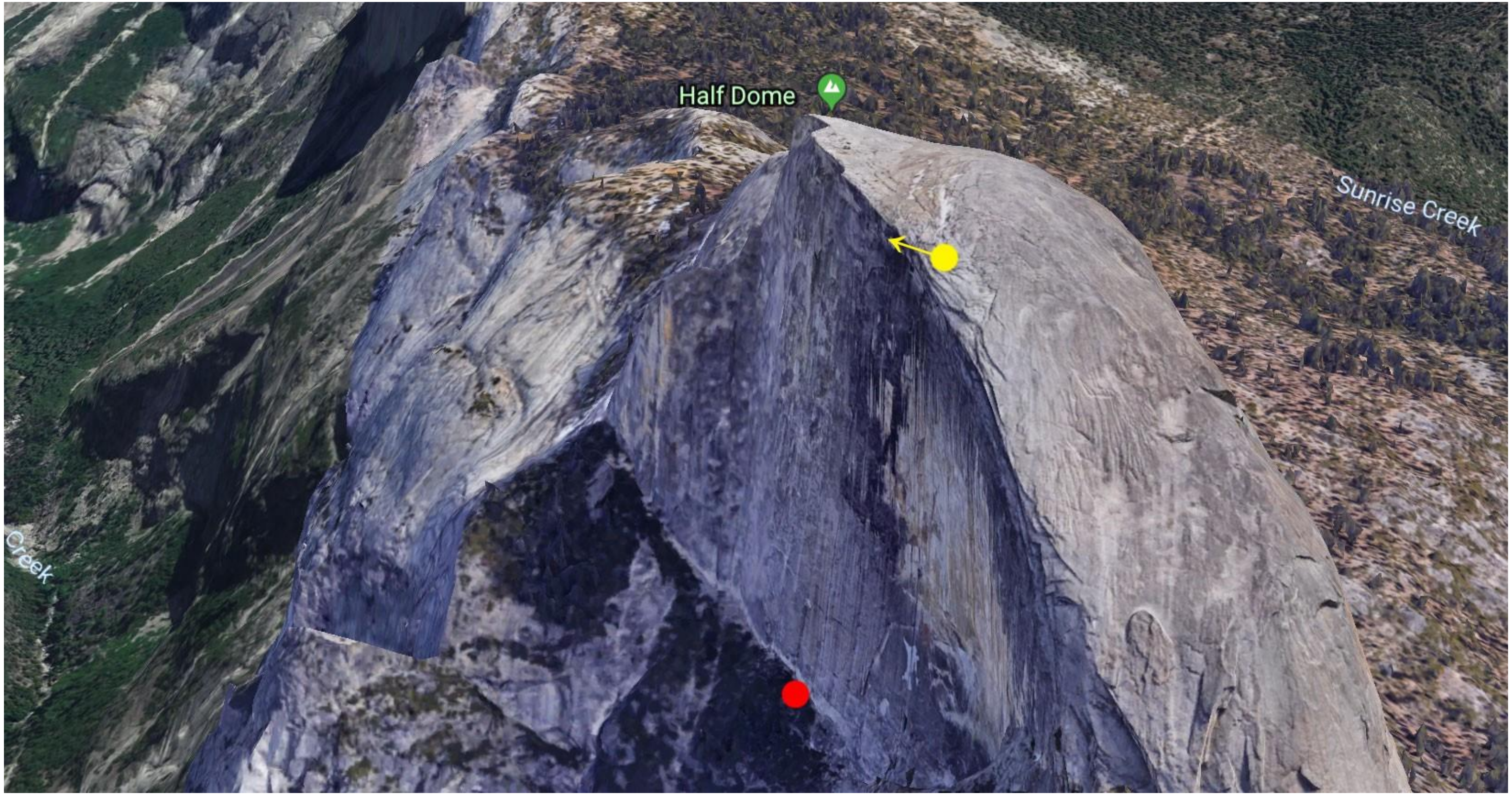
Actor-Critic Policy Gradient: $\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) A_{\mathbf{w}}(s_t) \right]$

1. Collect trajectories for policy π_θ
2. Estimate advantages A
3. Compute policy gradient $\hat{g}$
4. Update policy parameters $\theta_{new} = \theta + \epsilon \cdot \hat{g}$
5. GOTO 1

Problem?

- Data Inefficiency
  - On-policy method: for each new policy, we need to generate a completely new trajectory
  - The data is thrown out after just one gradient update
  - As complex neural networks need many updates, this makes the training process very slow
- Unstable update： step size is very important
  - If step size is too large:
    - Large step → bad policy
    - Next batch is generated from current bad policy → collect bad samples
    - Bad samples → worse policy
      (compare to supervised learning: the correct label and data in the following batches may correct it)
  - If step size is too small: the learning process is slow

A too large step leads to a disaster



Trust Region Policy Optimization (TRPO)

Line search
(like gradient ascent)

Trust region

Outline

- Deriving the optimization objective function of TRPO
- Guaranteed monotonic improvement
- Optimizing the objective function with KL constrained
    - Natural Gradient Ascent (NGA)
- TRPO improvements over NGA
- Shortcomings of TRPO
- Proximal Policy Optimization (PPO)

Objective of Policy Gradient Methods

- Policy objective:

$$J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\sum_{t=0}^{\infty}\gamma^t r_t$$

- Policy objective can be written in terms of old one:

$$J(\pi_{\theta'}) - J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi'_\theta}\sum_{t=0}^{\infty}\gamma^t A^{\pi_\theta}(s_t, a_t)$$

- Equivalently for succinctness:

$$J(\pi') - J(\pi) = \mathbb{E}_{\tau\sim\pi'}\sum_{t=0}^{\infty}\gamma^t A^{\pi}(s_t, a_t)$$

Related to the Advantage Function

How to estimate this?

$$
\begin{aligned}
&\mathop{\mathrm{E}}_{\tau\sim\pi'}\left[\sum_{t=0}^{\infty}\gamma^t A^{\pi}(s_t, a_t)\right] \\
&= \mathop{\mathrm{E}}_{\tau\sim\pi'}\left[\sum_{t=0}^{\infty}\gamma^t \left(R(s_t, a_t, s_{t+1}) + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)\right)\right] \\
&= J(\pi') + \mathop{\mathrm{E}}_{\tau\sim\pi'}\left[\sum_{t=0}^{\infty}\gamma^{t+1} V^{\pi}(s_{t+1}) - \sum_{t=0}^{\infty}\gamma^t V^{\pi}(s_t)\right] \\
&= J(\pi') + \mathop{\mathrm{E}}_{\tau\sim\pi'}\left[\sum_{t=1}^{\infty}\gamma^{t} V^{\pi}(s_{t}) - \sum_{t=0}^{\infty}\gamma^t V^{\pi}(s_t)\right] \\
&= J(\pi') - \mathop{\mathrm{E}}_{\tau\sim\pi'}\left[V^{\pi}(s_0)\right] \\
&= J(\pi') - J(\pi)
\end{aligned}
$$

Aside: Importance Sampling

- Estimate one distribution by sampling from another distribution

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

$$\begin{aligned}
E_{x\sim p}[f(x)] &= \int f(x)p(x)dx \\
&= \int f(x)\frac{p(x)}{q(x)}q(x)dx \\
&= E_{x\sim q}\left[f(x)\frac{p(x)}{q(x)}\right] \\
&\approx \frac{1}{N}\sum_{i=1,x^i\in q}^{N} f(x^i)\frac{p(x^i)}{q(x^i)}
\end{aligned}$$

Aside: Variance After Importance Sampling

No free lunch!

Two expectations are same, but we are using sampling method to estimate them

➔ variance is also important

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

$$VAR[X] = E[X^2] - (E[X])^2$$

$$Var_{x\sim p}[f(x)]$$

$$= E_{x\sim p}[f(x)^2] - \left(E_{x\sim p}[f(x)]\right)^2$$

$$Var_{x\sim q}[f(x)\frac{p(x)}{q(x)}]$$

$$= E_{x\sim q}\left[\left(f(x)\frac{p(x)}{q(x)}\right)^2\right] - \left(E_{x\sim q}\left[f(x)\frac{p(x)}{q(x)}\right]\right)^2$$

$$= E_{x\sim p}\left[f(x)^2\frac{p(x)}{q(x)}\right] - \left(E_{x\sim p}[f(x)]\right)^2$$

Price (Tradeoff): we may need to sample more data, if $\frac{p(x)}{q(x)}$ is far away from 1

Estimating Objective with Importance Sampling

$$J(\pi') - J(\pi) = \mathbb{E}_{\tau \sim \pi'} \sum_{t=0}^{\infty} \gamma^t A^{\pi}(s_t, a_t)$$
$$= \mathbb{E}_{s \sim d^{\pi'}, a \sim \pi'} A^{\pi}(s, a)$$
$$= \mathbb{E}_{s \sim d^{\pi'}, a \sim \pi} \left[\frac{\pi'(a \mid s)}{\pi(a \mid s)} A^{\pi}(s, a) \right]$$

Discounted state visit frequency:

$$d^{\pi}(s) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t P(s_t = s \mid \pi)$$

But how to sample state from a policy that we are trying to optimize?

Just using the old policy:

$$J(\pi') - J(\pi) \approx \mathbb{E}_{s \sim d^{\pi}, a \sim \pi} \frac{\pi'(a \mid s)}{\pi(a \mid s)} A^{\pi}(s, a)$$
$$= \mathscr{L}_{\pi}(\pi')$$

Lower bound of Optimization Objective

- Lower Bound:

$$J(\pi') - J(\pi) \geq \mathcal{L}_\pi(\pi') - C\sqrt{\underset{s\sim d^\pi}{\mathrm{E}}[D_{KL}(\pi'||\pi)[s]]}$$

- Optimizing the objective function:

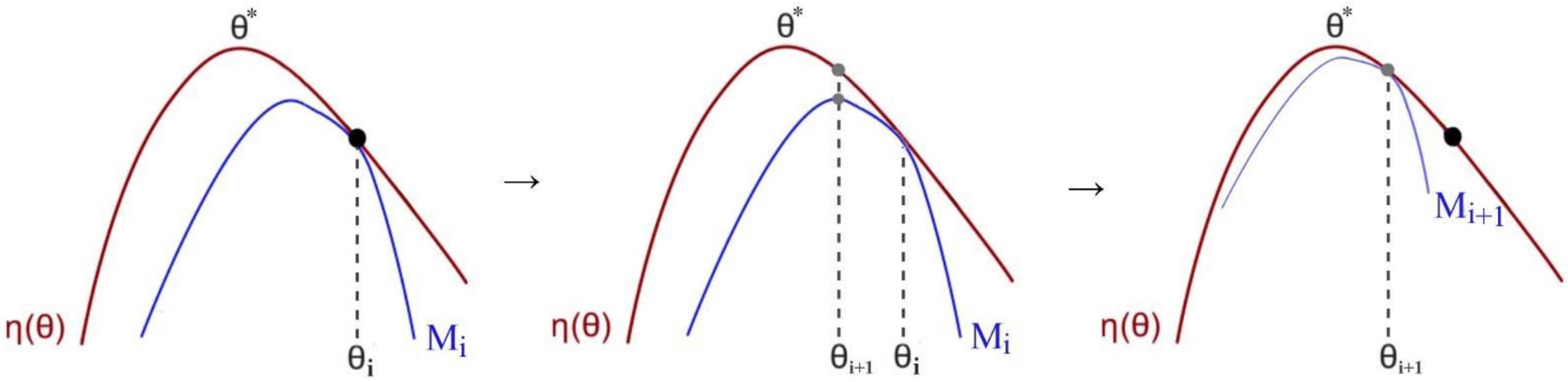
$$\underset{\pi'}{maximize} \;\; J(\pi') - J(\pi)$$

- By maximizing the lower bound

$$\max_{\pi'} \mathcal{L}_\pi \; (\pi') - C\sqrt{\underset{s\sim d^{\pi_k}}{\mathrm{E}}[D_{KL}(\pi'||\pi)[s]]}$$

J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. "Trust Region Policy Optimization".

Minorize-Maximization algorithm



By optimizing a lower bound function approximating η locally, it ***guarantees*** *policy improvement every time and leads us to the (local) optimal policy eventually.*

Monotonic Improvement Theorem

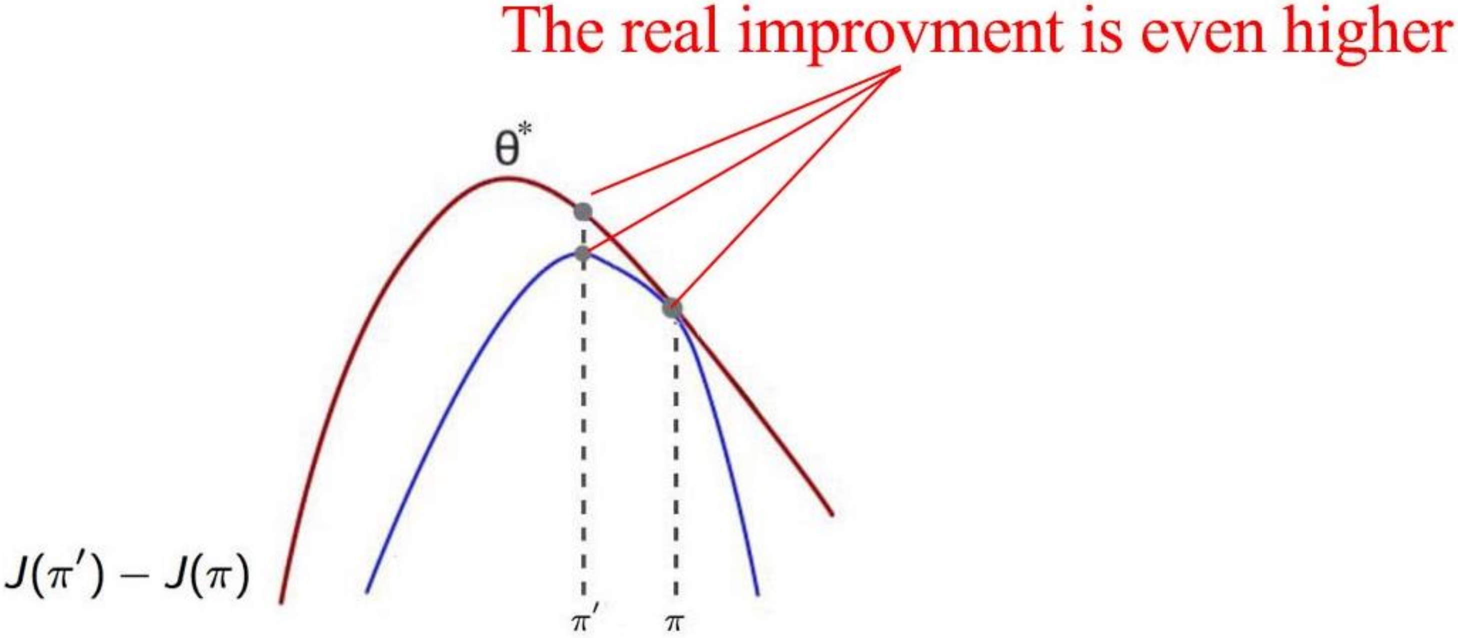
Proof of improvement guarantee: Suppose π_{k+1} and π_k are related by

$$\pi_{k+1} = \arg\max_{\pi'} \mathcal{L}_{\pi_k}(\pi') - C\sqrt{\mathop{\mathrm{E}}_{s\sim d^{\pi_k}}\left[D_{KL}(\pi'||\pi_k)[s]\right]}.$$

- π_k is a feasible point, and the objective at π_k is equal to 0.
  - $\mathcal{L}_{\pi_k}(\pi_k) \propto \mathop{\mathrm{E}}_{s,a\sim d^{\pi_k},\pi_k}\left[A^{\pi_k}(s,a)\right] = 0$
  - $D_{KL}(\pi_k||\pi_k)[s] = 0$
- $\implies$ optimal value ≥ 0
- $\implies$ by the performance bound, $J(\pi_{k+1}) - J(\pi_k) \geq 0$

Greater Improvement in the real objective

$$J(\pi') - J(\pi) \geq \mathcal{L}_\pi(\pi') - C\sqrt{\mathop{\mathrm{E}}_{s\sim d^\pi}[D_{KL}(\pi'||\pi)[s]]}$$



Recap: Objective Function

$$\max_{\pi'} \mathcal{L}_\pi(\pi') - C\sqrt{\mathop{\mathrm{E}}_{s\sim d^{\pi_k}}[D_{KL}(\pi' \| \pi)[s]]}$$

- With the Lagrangian Duality, this object is mathematically the same as following using a trust region constraint:

$$\max_{\pi'} \mathcal{L}_\pi(\pi')$$
$$\text{s.t.} \mathop{\mathrm{E}}_{s\sim d^{\pi}}[D_{KL}(\pi' \| \pi)[s]] \leq \delta$$

Trust region

KL Penalty vs. KL constraint

- C gets very high when γ is close to one and the corresponding gradient step size becomes too small.

$$C \propto \frac{\epsilon\gamma}{(1-\gamma)^2}$$

- Empirical results show that it needs to more adaptive
- But Tuning C is hard (need some trick just like PPO)
- TRPO uses trust region constraint and make δ a tunable hyperparameter.

Outline

- Deriving the optimization objective function of TRPO
- Guaranteed monotonic improvement
- **Optimizing the objective function with KL constrained**
  - **Natural Gradient Ascent (NGA)**
- TRPO improvements over NGA
- Shortcomings of TRPO
- Proximal Policy Optimization (PPO)

Trust Region Policy Optimization

$$\max_{\pi'} \mathcal{L}_\pi(\pi')$$
$$\text{s.t.} \underset{s\sim d^\pi}{\mathrm{E}}\left[D_{KL}(\pi' || \pi)[s]\right] \leq \delta$$

- $\text{maximize}_\theta\, L_{\pi_{\theta_{\text{old}}}}(\pi_\theta) - \beta \cdot \overline{\mathrm{KL}}_{\pi_{\theta_{\text{old}}}}(\pi_\theta)$
- Make linear approximation to $L_{\pi_{\theta_{\text{old}}}}$ and quadratic approximation to KL term:

$$\underset{\theta}{\text{maximize}} \quad g \cdot (\theta - \theta_{\text{old}}) - \tfrac{\beta}{2}(\theta - \theta_{\text{old}})^T F(\theta - \theta_{\text{old}})$$
$$\text{where} \quad g = \frac{\partial}{\partial\theta} L_{\pi_{\theta_{\text{old}}}}(\pi_\theta)\big|_{\theta=\theta_{\text{old}}}, \quad F = \frac{\partial^2}{\partial^2\theta}\overline{\mathrm{KL}}_{\pi_{\theta_{\text{old}}}}(\pi_\theta)\big|_{\theta=\theta_{\text{old}}}$$

Solving the KL Constrained Problem

Unconstrained penalized objective:

$$\boxed{\theta_{new} = \theta_{old} + d}$$

$$d^* = \arg\max_d J(\theta + d) - \lambda(\mathrm{D}_{\mathrm{KL}}\left[\pi_\theta \| \pi_{\theta+d}\right] - \epsilon)$$

First order Taylor expansion for the loss and second order for the KL:

$$\approx \arg\max_d J(\theta_{old}) + \nabla_\theta J(\theta)\,|_{\theta=\theta_{old}} \cdot d - \frac{1}{2}\lambda(d^\top \nabla_\theta^2 \mathrm{D}_{\mathrm{KL}}\left[\pi_{\theta_{old}} \| \pi_\theta\right]|_{\theta=\theta_{old}} d) + \lambda\epsilon$$

Taylor Expansion of KL

$$D_{KL}(p_{\theta_{old}}|p_\theta) \approx D_{KL}(p_{\theta_{old}}|p_{\theta_{old}}) + d^\top \nabla_\theta D_{KL}(p_{\theta_{old}}|p_\theta)|_{\theta=\theta_{old}} + \frac{1}{2} d^\top \nabla^2_\theta D_{KL}(p_{\theta_{old}}|p_\theta)|_{\theta=\theta_{old}} d$$

$$\theta_{new} = \theta_{old} + d$$

$$\begin{aligned}
\nabla_\theta D_{KL}(p_{\theta_{old}}|p_\theta)|_{\theta=\theta_{old}} &= -\nabla_\theta \mathbb{E}_{x\sim p_{\theta_{old}}} \log P_\theta(x)|_{\theta=\theta_{old}} \\
&= -\mathbb{E}_{x\sim p_{\theta_{old}}} \nabla_\theta \log P_\theta(x)|_{\theta=\theta_{old}} \\
&= -\mathbb{E}_{x\sim p_{\theta_{old}}} \frac{1}{P_{\theta_{old}}(x)} \nabla_\theta P_\theta(x)|_{\theta=\theta_{old}} \\
&= \int_x P_{\theta_{old}}(x) \frac{1}{P_{\theta_{old}}(x)} \nabla_\theta P_\theta(x)|_{\theta=\theta_{old}} \, dx \\
&= \int_x \nabla_\theta P_\theta(x)|_{\theta=\theta_{old}} \, dx \\
&= \nabla_\theta \int_x P_\theta(x)|_{\theta=\theta_{old}} \, dx \\
&= 0
\end{aligned}$$

$$KL(p_{\theta_{old}}|p_\theta) = \mathbb{E}_{x\sim p_{\theta_{old}}} \log\left(\frac{P_{\theta_{old}}(x)}{P_\theta(x)}\right)$$

Taylor Expansion of KL

$$\mathrm{D_{KL}}(p_{\theta_{old}}|p_\theta) \approx \mathrm{D_{KL}}(p_{\theta_{old}}|p_{\theta_{old}}) + d^\top \nabla_\theta \mathrm{KL}(p_{\theta_{old}}|p_\theta)|_{\theta=\theta_{old}} + \frac{1}{2} d^\top \boxed{\nabla_\theta^2 \mathrm{D_{KL}}(p_{\theta_{old}}|p_\theta)|_{\theta=\theta_{old}}}\, d$$

$$\begin{aligned}
\boxed{\nabla_\theta^2 \mathrm{D_{KL}}(p_{\theta_{old}}|p_\theta)|_{\theta=\theta_{old}}} &= -\mathbb{E}_{x\sim p_{\theta_{old}}} \nabla_\theta^2 \log P_\theta(x)|_{\theta=\theta_{old}} \\
&= -\mathbb{E}_{x\sim p_{\theta_{old}}} \nabla_\theta \left(\frac{\nabla_\theta P_\theta(x)}{P_\theta(x)} \right)|_{\theta=\theta_{old}} \\
&= -\mathbb{E}_{x\sim p_{\theta_{old}}} \left(\frac{\nabla_\theta^2 P_\theta(x) P_\theta(x) - \nabla_\theta P_\theta(x) \nabla_\theta P_\theta(x)^\top}{P_\theta(x)^2} \right)|_{\theta=\theta_{old}} \\
&= -\mathbb{E}_{x\sim p_{\theta_{old}}} \frac{\nabla_\theta^2 P_\theta(x)|_{\theta=\theta_{old}}}{P_{\theta_{old}}(x)} + \mathbb{E}_{x\sim p_{\theta_{old}}} \nabla_\theta \log P_\theta(x) \nabla_\theta \log P_\theta(x)^\top|_{\theta=\theta_{old}} \\
&= \mathbb{E}_{x\sim p_{\theta_{old}}} \nabla_\theta \log P_\theta(x) \nabla_\theta \log P_\theta(x)^\top|_{\theta=\theta_{old}}
\end{aligned}$$

$$\boxed{\mathrm{D_{KL}}(p_{\theta_{old}}|p_\theta) = \mathbb{E}_{x\sim p_{\theta_{old}}} \log \left(\frac{P_{\theta_{old}}(x)}{P_\theta(x)} \right)}$$

Fish Information Matrix

- Exactly Equivalent to the Hessian of KL divergence

$$\mathbf{F}(\theta) = \mathbb{E}_\theta \left[\nabla_\theta \log p_\theta(x) \nabla_\theta \log p_\theta(x)^\top \right]$$

$$\mathbf{F}(\theta_{old}) = \nabla^2_\theta \mathrm{D}_{\mathrm{KL}}(p_{\theta_{old}} | p_\theta) |_{\theta=\theta_{old}}$$

- Map changes between policy and parameter spaces

$$\begin{aligned}\mathrm{D}_{\mathrm{KL}}(p_{\theta_{old}} | p_\theta) &\approx \mathrm{D}_{\mathrm{KL}}(p_{\theta_{old}} | p_{\theta_{old}}) + d^\top \nabla_\theta \mathrm{D}_{\mathrm{KL}}(p_{\theta_{old}} | p_\theta) |_{\theta=\theta_{old}} + \frac{1}{2} d^\top \nabla^2_\theta \mathrm{D}_{\mathrm{KL}}(p_{\theta_{old}} | p_\theta) |_{\theta=\theta_{old}} d \\ &= \frac{1}{2} d^\top \mathbf{F}(\theta_{old}) d \\ &= \frac{1}{2} (\theta - \theta_{old})^\top \mathbf{F}(\theta_{old}) (\theta - \theta_{old})\end{aligned}$$

Solving the KL Constrained Problem

Unconstrained penalized objective:

$$d* = \arg\max_{d} J(\theta + d) - \lambda(\mathrm{D}_{\mathrm{KL}}\left[\pi_\theta \| \pi_{\theta+d}\right] - \epsilon)$$

First order Taylor expansion for the loss and second order for the KL:

$$\approx \arg\max_{d} J(\theta_{old}) + \nabla_\theta J(\theta)\mid_{\theta=\theta_{old}} \cdot d - \frac{1}{2}\lambda(d^\top \nabla_\theta^2 \mathrm{D}_{\mathrm{KL}}\left[\pi_{\theta_{old}} \| \pi_\theta\right]\mid_{\theta=\theta_{old}} d) + \lambda\epsilon$$

Substitute for the information matrix:

$$= \arg\max_{d} \nabla_\theta J(\theta)\mid_{\theta=\theta_{old}} \cdot d - \frac{1}{2}\lambda(d^\top \mathbf{F}(\theta_{old})d)$$

$$= \arg\min_{d} - \nabla_\theta J(\theta)\mid_{\theta=\theta_{old}} \cdot d + \frac{1}{2}\lambda(d^\top \mathbf{F}(\theta_{old})d)$$

Natural Gradient Descent

Setting the gradient to zero:

$$0 = \frac{\partial}{\partial d}\left(-\nabla_\theta J(\theta)\,|_{\theta=\theta_{old}} \cdot d + \frac{1}{2}\lambda(d^\top \mathbf{F}(\theta_{old})d)\right)$$

$$= -\nabla_\theta J(\theta)\,|_{\theta=\theta_{old}} + \frac{1}{2}\lambda(\mathbf{F}(\theta_{old}))d$$

$$d = \frac{2}{\lambda}\mathbf{F}^{-1}(\theta_{old})\nabla_\theta J(\theta)\,|_{\theta=\theta_{old}}$$

The natural gradient:

$$\boxed{\tilde{\nabla} J(\theta) = \mathbf{F}^{-1}(\theta_{old})\nabla_\theta J(\theta)}$$

$$\theta_{new} = \theta_{old} + \alpha \cdot \mathbf{F}^{-1}(\theta_{old})\hat{g}$$

$$\mathrm{D}_{\mathrm{KL}}(\pi_{\theta_{old}} | \pi_\theta) \approx \frac{1}{2}(\theta - \theta_{old})^\top \mathbf{F}(\theta_{old})(\theta - \theta_{old})$$

$$\frac{1}{2}(\alpha g_N)^\top \mathbf{F}(\alpha g_N) = \epsilon$$

$$\alpha = \sqrt{\frac{2\epsilon}{(g_N^\top \mathbf{F} g_N)}}$$

Trust Region Policy Optimization

Due to the quadratic approximation, the KL constraint may be violated! What if we just do a line search to find the best stepsize, making sure:

- I am improving my objective J(\theta)
- The KL constraint is not violated!

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] \\ \text{subject to} \quad & \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta. \end{aligned}$$

Algorithm 2 Line Search for TRPO

Compute proposed policy step $\Delta_k = \sqrt{\frac{2\delta}{\hat{g}_k^T \hat{H}_k^{-1} \hat{g}_k}} \hat{H}_k^{-1} \hat{g}_k$

for $j = 0, 1, 2, ..., L$ **do**

  Compute proposed update $\theta = \theta_k + \alpha^j \Delta_k$

  if $\mathcal{L}_{\theta_k}(\theta) \geq 0$ and $\bar{D}_{KL}(\theta || \theta_k) \leq \delta$ **then**

    accept the update and set $\theta_{k+1} = \theta_k + \alpha^j \Delta_k$

    break

  end if

end for

Trust Region Policy Optimization

TRPO= NPG +Linesearch +monotonic improvement theorem!

Algorithm 3 Trust Region Policy Optimization

Input: initial policy parameters θ_0

for $k = 0, 1, 2, \ldots$ **do**

Collect set of trajectories $\mathcal{D}_k$ on policy $\pi_k = \pi(\theta_k)$

Estimate advantages $\hat{A}_t^{\pi_k}$ using any advantage estimation algorithm

Form sample estimates for

- policy gradient $\hat{g}_k$ (using advantage estimates)
- and KL-divergence Hessian-vector product function $f(v) = \hat{H}_k v$

Use CG with n_{cg} iterations to obtain $x_k \approx \hat{H}_k^{-1} \hat{g}_k$

Estimate proposed step $\Delta_k \approx \sqrt{\frac{2\delta}{x_k^T \hat{H}_k x_k}} x_k$

Perform backtracking line search with exponential decay to obtain final update

$$\theta_{k+1} = \theta_k + \alpha^j \Delta_k$$

end for

Summary of TROP

- Pros
  - Proper learning step
  - Monotonic improvement guarantee

- Cons
  - Poor scalability
    - Computing Fisher Information Matrix every time for the current policy model is expensive
  - Not quite sample efficient
    - Requiring a large batch of rollouts to approximate accurately

Actor-Critic using Kronecker-Factored Trust Region (ACKTR)

ACKTR speeds up the optimization by reducing the complexity of calculating the inverse of the ***F*** using the Kronecker-factored approximation

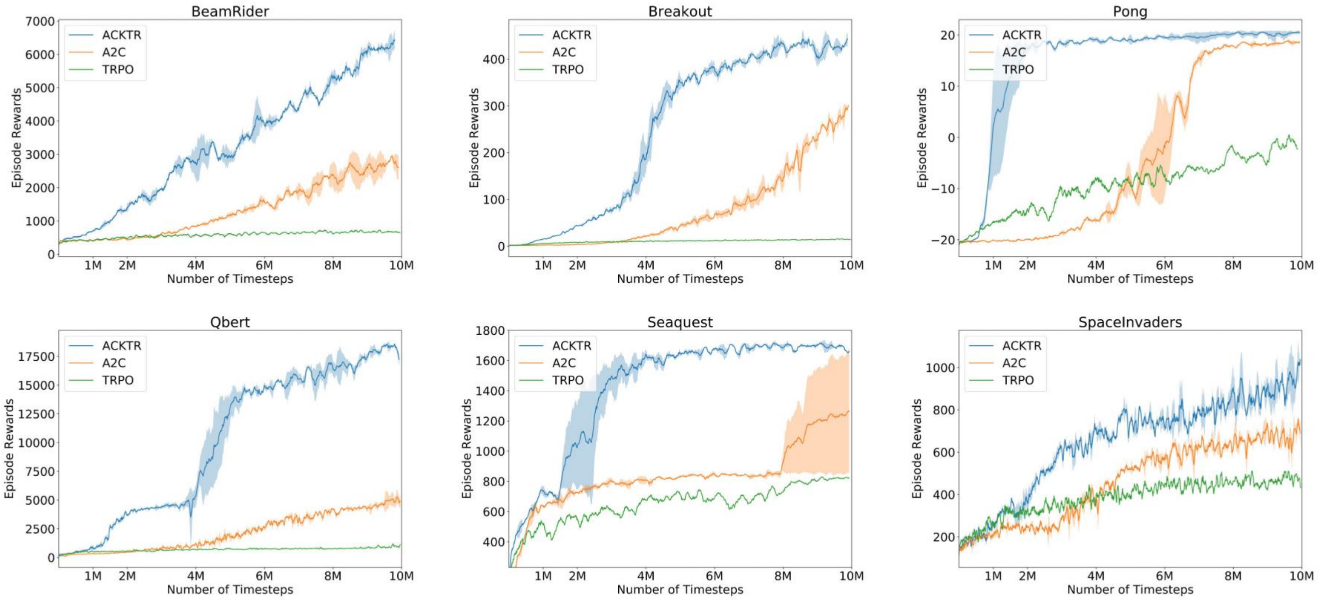


Figure 1: Performance comparisons on six standard Atari games trained for 10 million timesteps (1 timestep equals 4 frames). The shaded region denotes the standard deviation over 2 random seeds.

Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO), which perform comparably or better than state-of-the-art approaches while being much simpler to implement and tune.

-- OpenAI

- 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 objective function?

PPO with Adaptive KL Penalty

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right]$$

$$\text{subject to} \quad \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta.$$

or

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] - \beta\hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]]$$

Hard to pick β value → use adaptive β

PPO with Adaptive KL Penalty

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

Hard to pick β value → use adaptive β

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

- If $d < d_{\text{targ}}/1.5$, $\beta \leftarrow \beta/2$
- If $d > d_{\text{targ}} \times 1.5$, $\beta \leftarrow \beta \times 2$

Still need to set up a KL divergence target value ...

PPO with Adaptive KL Penalty

Algorithm 4 PPO with Adaptive KL Penalty

Input: initial policy parameters θ_0, initial KL penalty β_0, target KL-divergence δ
for $k = 0, 1, 2, ...$ **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

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

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

PPO with Clipped Objective

Importance Sampling in Policy Gradient

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

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

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

Optimization objective function with this gradient?

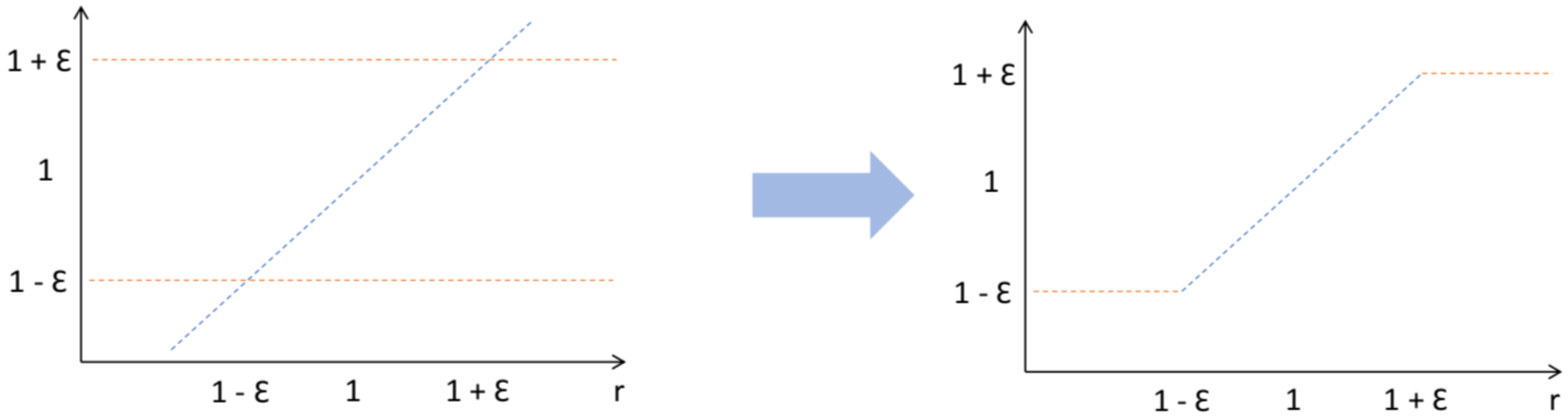
$$J(\theta) = E_{(s_t, a_t)\sim\pi_{\theta_{old}}}[\frac{\pi_\theta(s_t, a_t)}{\pi_{\theta_{old}}(s_t, a_t)} A(s_t, a_t)]$$

Surrogate objective function

PPO with Clipped Objective

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t \right] \qquad r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$$

Fluctuation happens when r changes too quickly → limit r within a range?



$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t) \right]$$

PPO with Clipped Objective

Algorithm 5 PPO with Clipped Objective

Input: initial policy parameters θ_0, clipping threshold ϵ

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

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

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

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

end for

PPO in practice

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right]$$

Surrogate objective function

a squared-error loss for "critic"

$$(V_\theta(s_t) - V_t^{\mathrm{targ}})^2$$

entropy bonus to ensure sufficient exploration

encourage "diversity"

c1, c2: empirical values, in the paper, c1=1, c2=0.01

Performance

No clipping or penalty:	$L_t(\theta) = r_t(\theta)\hat{A}_t$
Clipping:	$L_t(\theta) = \min(r_t(\theta)\hat{A}_t, \mathrm{clip}(r_t(\theta)), 1-\epsilon, 1+\epsilon)\hat{A}_t$
KL penalty (fixed or adaptive)	$L_t(\theta) = r_t(\theta)\hat{A}_t - \beta\,\mathrm{KL}[\pi_{\theta_{\mathrm{old}}}, \pi_\theta]$

- Results from continuous control benchmark. Average normalized scores (over 21 runs of the algorithm, on 7 environments)

algorithm	avg. normalized score
No clipping or penalty	-0.39
Clipping, $\epsilon = 0.1$	0.76
Clipping, $\epsilon = 0.2$	**0.82**
Clipping, $\epsilon = 0.3$	0.70
Adaptive KL $d_{\mathrm{targ}} = 0.003$	0.68
Adaptive KL $d_{\mathrm{targ}} = 0.01$	0.74
Adaptive KL $d_{\mathrm{targ}} = 0.03$	0.71
Fixed KL, $\beta = 0.3$	0.62
Fixed KL, $\beta = 1.$	0.71
Fixed KL, $\beta = 3.$	0.72
Fixed KL, $\beta = 10.$	0.69

Performance

Results in MuJoCo environments, training for one million timesteps