

Lecture 18 - Policy Optimization

Guiliang Liu

The Chinese University of Hong Kong, Shenzhen

DDA4230: Reinforcement Learning
Course Page: [\[Click\]](#)

Policy Gradient in Episodic MDP

Policy Gradient Methods:

- Let τ denote a state-action sequence $s_0, a_0, \dots, s_T, a_T$.
- Let $r(\tau) = \sum_{t=0}^T r(s_t, a_t)$ denote trajectory reward .
- Let $P^{\pi_\theta}(\tau)$ denote the corresponding occupancy measure

Then for a policy π parameterized by θ , we desire to find

$$\max_{\theta} \mathbb{E} [P^{\pi_\theta}(\tau) r(\tau)] .$$



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Policy Gradient in Episodic MDP

Policy Gradient Methods: Taking gradient (denoted as g) with respect to θ gives us

$$g = \mathbb{E} \left[r(\tau) \nabla_{\theta} \log \left(\sum_{t=0}^T P(s_{t+1} | s_t, a_t) \cdot \pi_{\theta}(a_t | s_t) \right) \right] = \mathbb{E} \left[\sum_{t=0}^T r_t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \right].$$

This gradient is **unbiased** and we **do not need access to the dynamic model** to compute this.



Policy Gradient in Stationary MDP

There are several different related expressions for the policy gradient, which have the form

$$g = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t \Psi_t \nabla_{\theta} \log \pi_{\theta} (a_t | s_t) \right],$$



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Policy Gradient in Stationary MDP

$\sum_{t=0}^{\infty} \Psi_t$ could be the following:

1. $\sum_{t=0}^{\infty} \gamma^t r_t$: the total reward of the trajectory **Monta-Carlo**;
2. $Q^{\pi}(s_t, a_t)$: the action value function **Temporal Difference**;
3. $\sum_{t'=t}^{\infty} \gamma^{t'} r_{t'}$: the reward following action a_t **Monta-Carlo**;
4. $\sum_{t'=t}^{\infty} \gamma^{t'} [r_{t'} - b(s_t)]$: the reward following action a_t with a baseline **Monta-Carlo**;
5. $\sum_{t'=t}^{\infty} \gamma^{t'} A^{\pi}(s_t, a_t)$: the advantage function **Temporal Difference**;
6. $\sum_{t'=t}^{\infty} \gamma^{t'} (r_{t'} + V^{\pi}(s_{t+1}) - V^{\pi}(s_t))$: the TD residual **Temporal Difference**.

The latter formulas use the definitions $A^{\pi}(s_t, a_t) := Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$, which is the advantage function.



Off Policy Optimization

- In policy optimization methods, the data samples we have collected **may not correspond to the policy we wish to optimize**.
- In this case, some special handling is needed so that the gradient estimates can be **unbiased**.

Let π_1 be the policy we are **currently following** and π_2 be the policy **we want to optimize**. Let them be parameterized by θ_1, θ_2 , respectively. Then we can use importance sampling to re-weight our objective as

$$\max_{\theta_1} \mathbb{E} \left[\frac{P^{\pi_{\theta_2}}(\tau)}{P^{\pi_{\theta_1}}(\tau)} r(\tau) \right].$$



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Trust Region Policy Optimization (TRPO)

Motivation: The challenge of step size.

- In the classic supervised learning setting or in the optimization literature, **having a bad step size may not be terrible**. This is because the next update can partially correct the error in the previous steps.
- In policy optimization, when the step size is too far, we obtain a terrible policy. This indicates that **the next batch of data will be collected under this terrible policy**. **Exploration could be exploratory, but updates should be more conservative**.
- It becomes not clear how to **recover** short of going back and **shrinking** the step size.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Trust Region Policy Optimization (TRPO)

One method of choosing the step size is by **line search**. The procedure is:

1. Calculate the **initial loss** (e.g., with Monte-Carlo estimation) and initialize the step size to be **a large value**;
2. Update the parameter with the gradients under the current step size can calculate the **new loss**;
3. **Decrease the value of step size** until we have found a new loss that is **less** than the initial loss.

However, 1) it may be expensive to compute so many gradients, 2) this method ignores the quality of our gradients.



Trust Region Policy Optimization (TRPO)

An alternative method is the trust region method.

Let us first denote $P(\tau | \theta) = P(s_0) \cdot \prod_{t=1}^T P(s_{t+1} | s_t, a_t) \pi_{\theta}(a_t | s_t)$. Then the trust region method finds us the next parameter $\theta + \delta\theta$ by solving the following problem.

$$\begin{aligned} \max_{\delta\theta} \quad & g^{\top} \delta\theta \\ \text{subject to} \quad & d_{\text{KL}}(P(\tau|\theta) || P(\tau|\theta + \delta\theta)) \leq \varepsilon, \end{aligned}$$

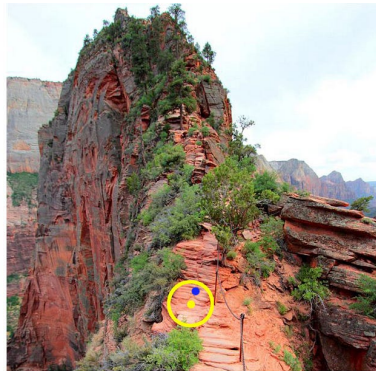
where d_{KL} denotes the KL divergence, g is our gradient estimate, and ε is a parameter we can set. Here, the change in the objective function is estimated by assuming that **the objective function in this neighboring area is linear.**



Trust Region Policy Optimization (TRPO)



Line search
(like gradient ascent)



Trust region

Source.¹

¹<https://jonathan-hui.medium.com/>

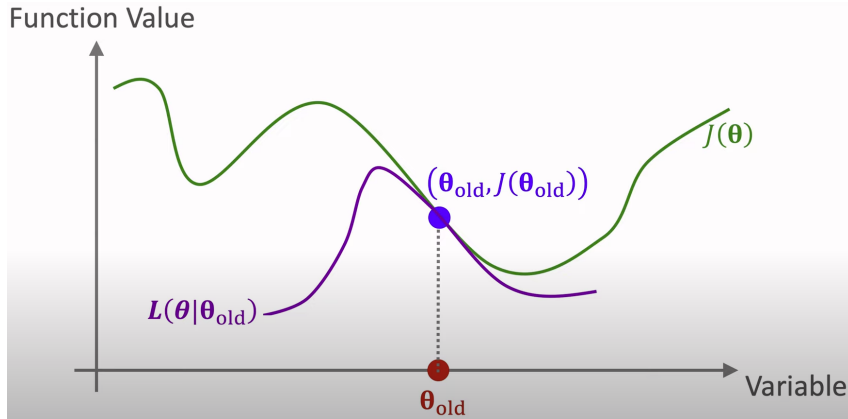
[rl-trust-region-policy-optimization-trpo-explained-a6ee04e00000](https://jonathan-hui.medium.com/rl-trust-region-policy-optimization-trpo-explained-a6ee04e00000)



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Trust Region Policy Optimization (TRPO)



Source.¹

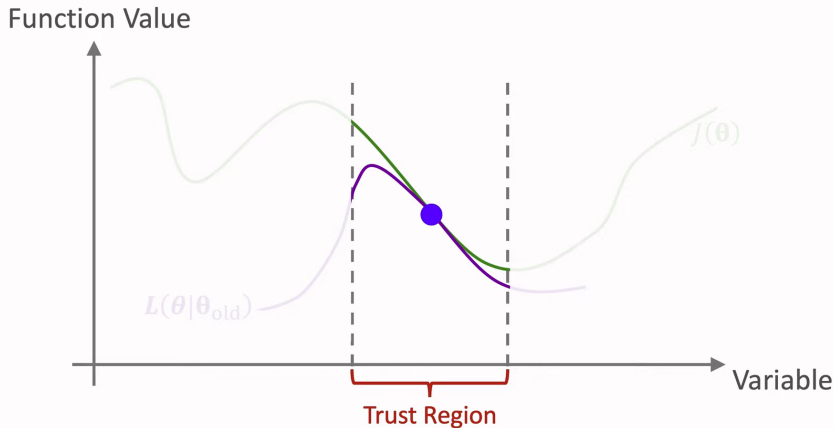
¹https://www.youtube.com/watch?v=fcSYiyvPjm4&ab_channel=ShusenWang



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Trust Region Policy Optimization (TRPO)



Source.¹

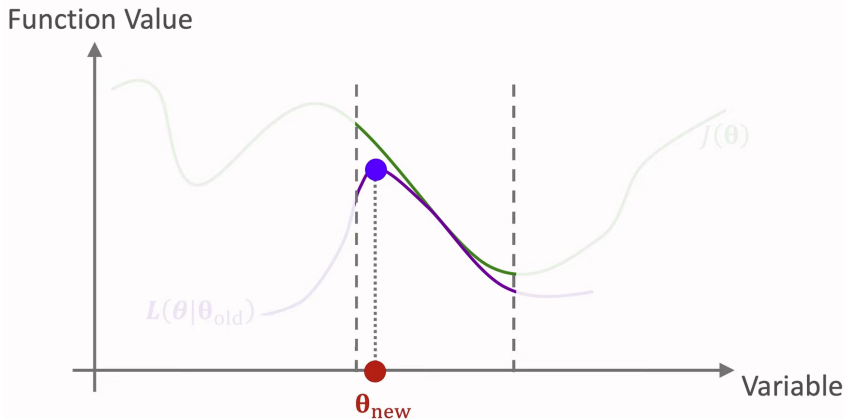
¹https://www.youtube.com/watch?v=fcSYiyvPjm4&ab_channel=ShusenWang



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Trust Region Policy Optimization (TRPO)



Source.¹

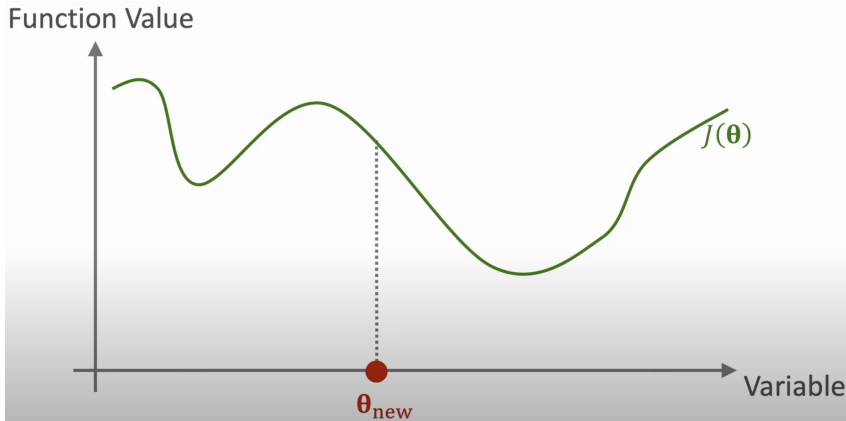
¹https://www.youtube.com/watch?v=fcSYiyvPjm4&ab_channel=ShusenWang



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Trust Region Policy Optimization (TRPO)



Source.¹

¹https://www.youtube.com/watch?v=fcSYiyvPjm4&ab_channel=ShusenWang



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Trust Region Policy Optimization (TRPO)

Using the expression of the KL divergence, we have

$$\begin{aligned} d_{\text{KL}}(P(\tau; \theta) \| P(\tau; \theta + \delta\theta)) &= \sum_{\tau} P(\tau; \theta) \log \frac{P(\tau; \theta)}{P(\tau; \theta + \delta\theta)} \\ &= \sum_{\tau} P(\tau; \theta) \log \frac{P(s_0) \prod_{t=0}^{T-1} \pi_{\theta}(a_t | s_t) P(s_{t+1} | s_t, a_t)}{P(s_0) \prod_{t=0}^{T-1} \pi_{\theta+\delta\theta}(a_t | s_t) P(s_{t+1} | s_t, a_t)} \\ &= \sum_{\tau} P(\tau; \theta) \log \frac{\prod_{t=0}^{T-1} \pi_{\theta}(a_t | s_t)}{\prod_{t=0}^{T-1} \pi_{\theta+\delta\theta}(a_t | s_t)}. \end{aligned}$$



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Trust Region Policy Optimization (TRPO)

With M samples, this term can be approximated by the sample average and we may rewrite the maximization problem to be

$$\begin{aligned} & \max_{\delta\theta} g^\top \delta\theta \\ & \text{subject to } \frac{1}{M} \sum_{(s,a)} \log \frac{\pi_\theta(a|s)}{\pi_{\theta+\delta\theta}(a|s)} \leq \epsilon. \end{aligned}$$

This maximization problem with the constraint can be hard to enforce given complicated policies like neural networks.



Trust Region Policy Optimization (TRPO)

We would need to approximate the KL divergence further for a feasible objective. This is done through **second-order approximation** with fisher matrix F_θ .

$$\begin{aligned} d_{\text{KL}}(\pi_\theta(a | s) || \pi_{\theta+\delta\theta}(a | s)) &\approx \delta\theta^\top \left(\sum_{(s,a) \sim \theta} \nabla_\theta \log \pi_\theta(a | s) \nabla_\theta \log \pi_\theta(a | s)^\top \right) \delta\theta \\ &= \delta\theta^\top F_\theta \delta\theta. \end{aligned}$$

Our problem is simplified to linear objective quadratic constrained optimization:

$$\begin{aligned} \max_{\delta\theta} \quad & g^\top \delta\theta \\ \text{subject to} \quad & \delta\theta^\top F_\theta \delta\theta \leq \epsilon, \end{aligned}$$



Trust Region Policy Optimization (TRPO)

The above linear objective quadratic constrained optimization problem could be solved analytically using the Karush-Kuhn-Tucker (KKT) conditions. Thus the final TRPO objective is given as

$$\text{Surrogate loss : } \max_{\pi} L(\pi) = \mathbb{E}_{\pi_{\text{old}}} \left[\frac{\pi(a | s)}{\pi_{\text{old}}(a | s)} A^{\pi_{\text{old}}}(s, a) \right]$$

$$\text{Constraint: } \mathbb{E}_{\pi_{\text{old}}} [d_{\text{KL}}(\pi \| \pi_{\text{old}})] \leq \varepsilon,$$

where A denotes the advantage function. This corresponds to a general policy gradient form we have mentioned earlier.



Proximal Policy Optimization

The PPO method enforces a “soft” constraint by adding a proximal value to the objective function. The objective is the following

$$L(\pi) = \mathbb{E}_{\pi_{\text{old}}} \left[\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)} A^{\pi_{\text{old}}}(s, a) - \beta d_{\text{KL}}(\pi_{\theta_{\text{old}}}, \pi_{\theta}) \right].$$

The β can be fixed or adaptively chosen (or simply set to 0).



Proximal Policy Optimization

The PPO method enforces a “soft” constraint by adding a proximal value to the objective function. The objective is the following

$$L(\pi) = \mathbb{E}_{\pi_{\text{old}}} \left[\frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_{\text{old}}}(a_t | s_t)} A^{\pi_{\text{old}}}(s, a) - \beta d_{\text{KL}}(\pi_{\theta_{\text{old}}}, \pi_{\theta}) \right].$$

The β can be fixed or adaptively chosen (or simply set to 0). One reason why one may wish to adaptively choose β is because it can be hard to find one β that performs well across different problems.



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Proximal Policy Optimization

The policy's performance can fluctuate greatly when $\rho_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ changes too quickly. Thus PPO limits ρ to a range of $[1 - \varepsilon, 1 + \varepsilon]$ such that no abrupt updates to the policy will be made. The surrogate objective is then written as

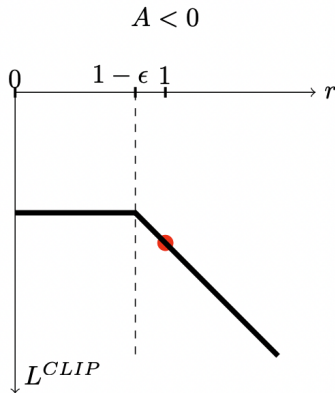
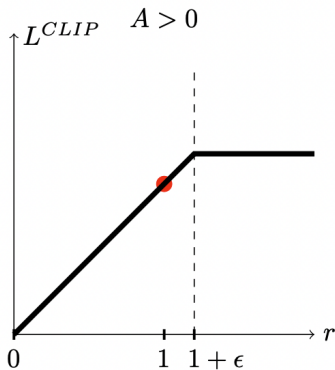
$$L^{CLIP}(\pi) = \mathbb{E}[\min\{\rho_t(\theta)A(s, a), \text{clip}(\rho_t(\theta), 1 - \varepsilon, 1 + \varepsilon)A(s, a)\}].$$

We take the minimum of the constrained and unconstrained objectives such that our final objective is a lower bound of the unclipped objective. With this scheme, we only ignore the change in probability ratio when it would make the objective improve, and we include it when it makes the objective worse.



Proximal Policy Optimization

In the following plots, r denotes the **rate** s.t. $r = \rho_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$



圳)
g Kong, Shenzhen

Proximal Policy Optimization

In the following algorithm, g denotes the clipping function.

Algorithm 1 PPO-Clip

- 1: Input: initial policy parameters θ_0 , initial value function parameters ϕ_0
- 2: **for** $k = 0, 1, 2, \dots$ **do**
- 3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
- 4: Compute rewards-to-go \hat{R}_t .
- 5: Compute advantage estimates, \hat{A}_t (using any method of advantage estimation) based on the current value function V_{ϕ_k} .
- 6: Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg \max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \min \left(\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \quad g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

typically via stochastic gradient ascent with Adam.

- 7: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \left(V_{\phi}(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.

- 8: **end for**



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Proximal Policy Optimization

Proximal Policy Optimization Algorithms

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov
OpenAI

`{joschu, filip, prafulla, alec, oleg}@openai.com`



香港中文大學(深圳)

The Chinese University of Hong Kong, Shenzhen

Proximal Policy Optimization

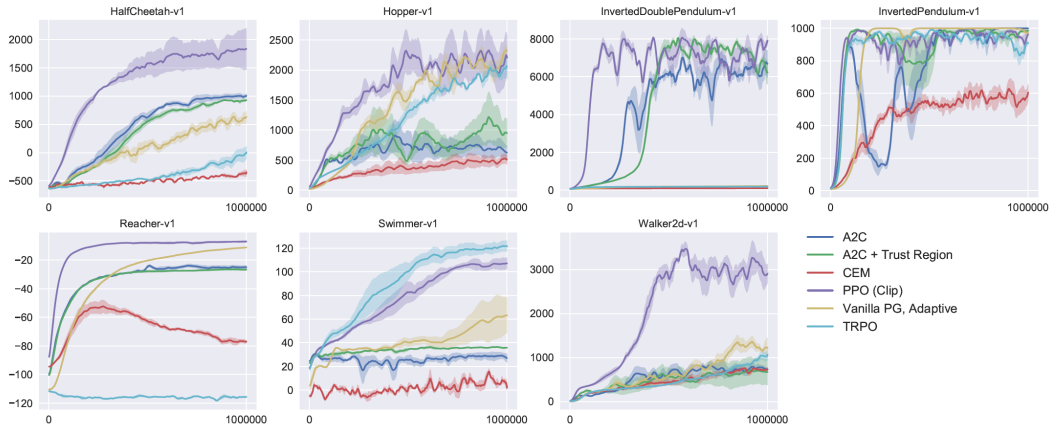
algorithm	avg. normalized score
No clipping or penalty	-0.39
Clipping, $\epsilon = 0.1$	0.76
Clipping, $\epsilon = 0.2$	0.82
Clipping, $\epsilon = 0.3$	0.70
Adaptive KL $d_{\text{targ}} = 0.003$	0.68
Adaptive KL $d_{\text{targ}} = 0.01$	0.74
Adaptive KL $d_{\text{targ}} = 0.03$	0.71
Fixed KL, $\beta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

Table 1: Results from continuous control benchmark. Average normalized scores (over 21 runs of the algorithm, on 7 environments) for each algorithm / hyperparameter setting . β was initialized at 1.



Proximal Policy Optimization

Results of robot control tasks in the MuJoCo environments ([click me](#)).



Shenzhen

Proximal Policy Optimization

Step 1

Collect demonstration data and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3.5 with supervised learning.



Step 2

Collect comparison data and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.



Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

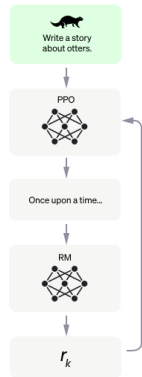
A new prompt is sampled from the dataset.

The PPO model is initialized from the supervised policy.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



彭 (深圳)

y of Hong Kong, Shenzhen

Question and Answering (Q&A)



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen