

Advanced Actor-Critic Methods (PPO, TRPO)

Alina Vereshchaka

CSE4/510 Reinforcement Learning
Fall 2019

avereshc@buffalo.edu

November 7, 2019

*Slides are adopted from Deep Reinforcement Learning by Sergey Levine & Policy Gradients Algorithms by Lilian Weng

Table of Contents

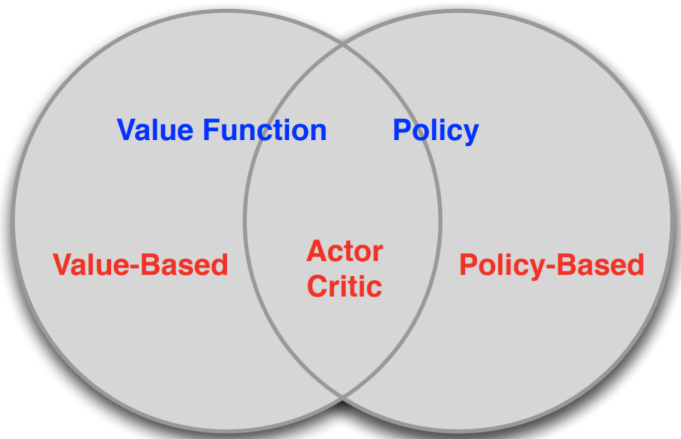
1 Recap: Actor-Critic

2 Importance Sampling

3 Trust region policy optimization (TRPO)

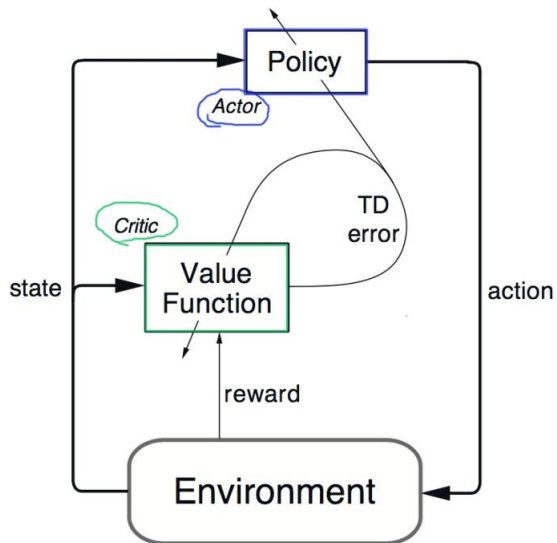
4 Proximal Policy Optimization (PPO)

Value Based and Policy-Based RL



- Value Based
 - Learn Value Function
 - Implicit policy
- Policy Based
 - No Value Function
 - Learn Policy
- Actor-Critic
 - Learn Value Function
 - Learn Policy

Actor-Critic



Summary of Policy Gradient Algorithms

- The **policy gradient** has many equivalent forms

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(s, a) G_t] \\ &= \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(s, a) Q_w(s, a)] \\ &= \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(s, a) A_w(s, a)] \\ &= \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(s, a) \delta]\end{aligned}$$

REINFORCE

Q Actor-Critic

Advantage Actor-Critic (A2C)

TD Actor-Critic

- Each leads a stochastic gradient ascent algorithm
- Critic uses **policy evaluation** (e.g. MC or TD learning) to estimate $Q_{\pi}(s, a)$, $A_{\pi}(s, a)$ or $V_{\pi}(s)$.

Deep Deterministic Policy Gradient (DDPG)

DDPG does soft updates (“conservative policy iteration”) on the parameters of both actor and critic

$$\theta' \leftarrow \tau\theta + (1 - \tau)\theta'$$

In this way, the target network values are constrained to change slowly, different from the design in DQN that the target network stays frozen for some period of time.

Deep Deterministic Policy Gradient (DDPG)

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ .

Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$

Initialize replay buffer R

for episode = 1, M **do**

 Initialize a random process \mathcal{N} for action exploration

 Receive initial observation state s_1

for $t = 1, T$ **do**

 Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise

 Execute action a_t and observe reward r_t and observe new state s_{t+1}

 Store transition (s_t, a_t, r_t, s_{t+1}) in R

 Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R

 Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$

 Update critic by minimizing the loss: $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$

 Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

 Update the target networks:

$$\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu'}$$

end for

end for

Deep Deterministic Policy Gradient (DDPG)

- DDPG is an algorithm which concurrently learns a Q-function and a policy
- DDPG is an off-policy algorithm
- DDPG can only be used for environments with continuous action spaces
- DDPG can be thought of as being deep Q-learning for continuous action spaces

Table of Contents

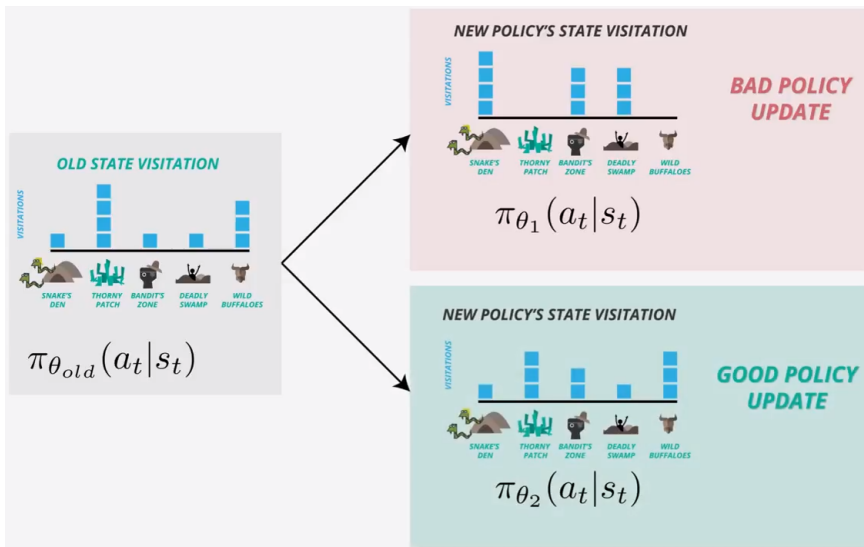
1 Recap: Actor-Critic

2 Importance Sampling

3 Trust region policy optimization (TRPO)

4 Proximal Policy Optimization (PPO)

Problems in Policy Gradient



On-policy Sampling

$$\theta^* = \arg \max_{\theta} J(\theta)$$

$$J(\theta) = E_{\tau \sim \pi_{\theta}(\tau)}[r(\tau)]$$


$$\nabla_{\theta} J(\theta) = E_{\tau \sim \pi_{\theta}(\tau)}[\nabla_{\theta} \log \pi_{\theta}(\tau) r(\tau)]$$

↑
this is trouble...

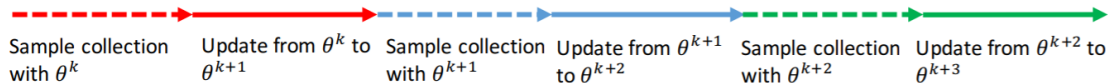
- Neural networks change only a little bit with each gradient step
- On-policy learning can be extremely inefficient!

REINFORCE algorithm:

↖ can't just skip this!

- 
1. sample $\{\tau^i\}$ from $\pi_{\theta}(\mathbf{a}_t|\mathbf{s}_t)$ (run it on the robot)
 2. $\nabla_{\theta} J(\theta) \approx \sum_i \left(\sum_t \nabla_{\theta} \log \pi_{\theta}(\mathbf{a}_t^i|\mathbf{s}_t^i) \right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i) \right)$
 3. $\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta)$

On-policy Sampling



Off-policy Sampling

$$\theta^* = \arg \max_{\theta} J(\theta)$$

$$J(\theta) = E_{\tau \sim \pi_{\theta}(\tau)}[r(\tau)]$$

what if we don't have samples from $\pi_{\theta}(\tau)$?

(we have samples from some $\bar{\pi}(\tau)$ instead)

$$J(\theta) = E_{\tau \sim \bar{\pi}(\tau)} \left[\frac{\pi_{\theta}(\tau)}{\bar{\pi}(\tau)} r(\tau) \right]$$

$$\pi_{\theta}(\tau) = p(\mathbf{s}_1) \prod_{t=1}^T \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)$$

$$\frac{\pi_{\theta}(\tau)}{\bar{\pi}(\tau)} = \frac{\cancel{p(\mathbf{s}_1)} \prod_{t=1}^T \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t) \cancel{p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)}}{\cancel{p(\mathbf{s}_1)} \prod_{t=1}^T \bar{\pi}(\mathbf{a}_t | \mathbf{s}_t) \cancel{p(\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t)}} = \frac{\prod_{t=1}^T \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)}{\prod_{t=1}^T \bar{\pi}(\mathbf{a}_t | \mathbf{s}_t)}$$

importance sampling

$$\begin{aligned} E_{x \sim p(x)}[f(x)] &= \int p(x) f(x) dx \\ &= \int \frac{q(x)}{q(x)} p(x) f(x) dx \\ &= \int q(x) \frac{p(x)}{q(x)} f(x) dx \\ &= E_{x \sim q(x)} \left[\frac{p(x)}{q(x)} f(x) \right] \end{aligned}$$

Off-Policy Sampling

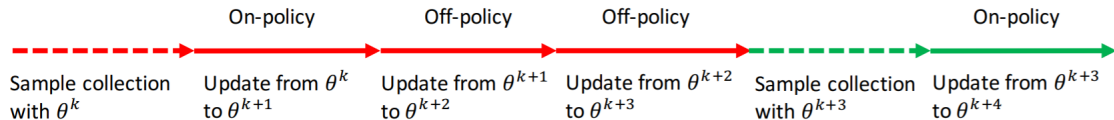


Table of Contents

- 1 Recap: Actor-Critic
- 2 Importance Sampling
- 3 Trust region policy optimization (TRPO)**
- 4 Proximal Policy Optimization (PPO)

Trust region policy optimization (TRPO)

- To improve training stability, we should avoid parameter updates that change the policy too much at one step

Trust region policy optimization (TRPO)

- To improve training stability, we should avoid parameter updates that change the policy too much at one step
- Trust region policy optimization (TRPO) (Schulman, et al., 2015) carries out this idea by enforcing a KL divergence constraint on the size of policy update at each iteration.

Trust region policy optimization (TRPO)

If off policy, the objective function measures the total advantage over the state visitation distribution and actions, while the rollout is following a different behavior policy $\beta(a|s)$:

$$J(\theta) = \sum_{s \in \mathcal{S}} \rho^{\pi_{\theta_{\text{old}}}} \sum_{a \in \mathcal{A}} (\pi_{\theta}(a|s) \hat{A}_{\theta_{\text{old}}}(s, a))$$

Trust region policy optimization (TRPO)

If off policy, the objective function measures the total advantage over the state visitation distribution and actions, while the rollout is following a different behavior policy $\beta(a|s)$:

$$\begin{aligned} J(\theta) &= \sum_{s \in \mathcal{S}} \rho^{\pi_{\theta_{\text{old}}}} \sum_{a \in \mathcal{A}} (\pi_{\theta}(a|s) \hat{A}_{\theta_{\text{old}}}(s, a)) \\ &= \sum_{s \in \mathcal{S}} \rho^{\pi_{\theta_{\text{old}}}} \sum_{a \in \mathcal{A}} \left(\beta(a|s) \frac{\pi_{\theta}(a|s)}{\beta(a|s)} \hat{A}_{\theta_{\text{old}}}(s, a) \right) \end{aligned} \quad ; \text{ Importance sampling}$$

Trust region policy optimization (TRPO)

If off policy, the objective function measures the total advantage over the state visitation distribution and actions, while the rollout is following a different behavior policy $\beta(a|s)$:

$$\begin{aligned} J(\theta) &= \sum_{s \in \mathcal{S}} \rho^{\pi_{\theta_{\text{old}}}} \sum_{a \in \mathcal{A}} (\pi_{\theta}(a|s) \hat{A}_{\theta_{\text{old}}}(s, a)) \\ &= \sum_{s \in \mathcal{S}} \rho^{\pi_{\theta_{\text{old}}}} \sum_{a \in \mathcal{A}} \left(\beta(a|s) \frac{\pi_{\theta}(a|s)}{\beta(a|s)} \hat{A}_{\theta_{\text{old}}}(s, a) \right) && \text{; Importance sampling} \\ &= \mathbb{E}_{s \sim \rho^{\pi_{\theta_{\text{old}}}}, a \sim \beta} \left[\frac{\pi_{\theta}(a|s)}{\beta(a|s)} \hat{A}_{\theta_{\text{old}}}(s, a) \right] \end{aligned}$$

where θ_{old} is the policy parameters before the update and thus known to us; $\beta(a|s)$ is the behavior policy for collecting trajectories.

Kullback-Leibler (KL) Divergence

- The Kullback-Leibler (KL) Divergence score, or KL divergence score, quantifies how much one probability distribution differs from another probability distribution

Kullback-Leibler (KL) Divergence

- The Kullback-Leibler (KL) Divergence score, or KL divergence score, quantifies how much one probability distribution differs from another probability distribution
- The KL divergence between two distributions Q and P:

$$KL(P||Q)$$

,where the $||$ indicates “divergence” or P's divergence from Q

Kullback-Leibler (KL) Divergence

- The Kullback-Leibler (KL) Divergence score, or KL divergence score, quantifies how much one probability distribution differs from another probability distribution
- The KL divergence between two distributions Q and P:

$$KL(P||Q)$$

,where the || indicates “divergence” or P's divergence from Q

- Given two probability distributions $p(x)$ and $q(x)$ over a discrete random variable X , the relative entropy given by $D(p||q)$ is defined as follows:

$$D_{KL}(p||q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}$$

Kullback-Leibler (KL) Divergence

- The Kullback-Leibler (KL) Divergence score, or KL divergence score, quantifies how much one probability distribution differs from another probability distribution
- The KL divergence between two distributions Q and P:

$$KL(P||Q)$$

,where the || indicates “divergence” or P's divergence from Q

- Given two probability distributions $p(x)$ and $q(x)$ over a discrete random variable X , the relative entropy given by $D(p||q)$ is defined as follows:

$$D_{KL}(p||q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}$$

- **Intuition:** when the probability for an event from P is large, but the probability for the same event in Q is small \rightarrow there is a large divergence. If the probability from P is small and the probability from Q is large \rightarrow a large divergence, but not as large as the first case.

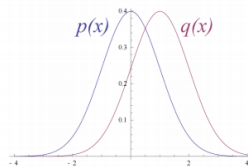
KL-Divergence

Measure the distance of two distributions

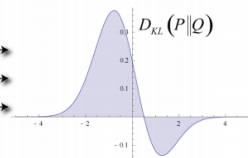
$$D_{KL}(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$$

KL divergence of two policies

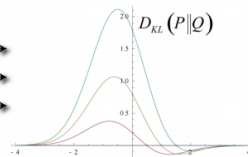
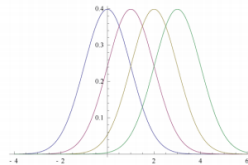
$$D_{KL}(\pi_1||\pi_2)[s] = \sum_{a \in A} \pi_1(a|s) \log \frac{\pi_1(a|s)}{\pi_2(a|s)}$$



Original Gaussian PDF's



KL Area to be Integrated



Trust Region Policy Optimization (TRPO)

Objective function:

$$J(\theta) = \mathbb{E}_{s \sim \rho^{\pi_{\theta_{\text{old}}}}, a \sim \pi_{\theta_{\text{old}}}} \left[\frac{\pi_{\theta}(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} \hat{A}_{\theta_{\text{old}}}(s, a) \right]$$

TRPO aims to maximize $J(\theta)$ subject to, **trust region** constraint which enforces the distance between old and new policies measured by KL-divergence to be small enough, within a parameter δ :

$$\mathbb{E}_{s \sim \rho^{\pi_{\theta_{\text{old}}}}} [D_{\text{KL}}(\pi_{\theta_{\text{old}}}(\cdot|s) \parallel \pi_{\theta}(\cdot|s))] \leq \delta$$

In this way, the old and new policies would not diverge too much when this hard constraint is met. While still, TRPO can guarantee a monotonic improvement over policy iteration.

Trust Region Policy Optimization



Gradient descent is fast and simple in optimizing an objective function. It picks the steepest direction and then move forward by a step size, but may not work in reinforcement learning.



Trust region determines the maximum step size that we want to explore. Then, locate the optimal point within the trust region and resume the search from there.

Table of Contents

- 1 Recap: Actor-Critic
- 2 Importance Sampling
- 3 Trust region policy optimization (TRPO)
- 4 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO)

- Given that TRPO is relatively complicated, proximal policy optimization (PPO) simplifies it by using a **clipped surrogate objective** while retaining similar performance.

Proximal Policy Optimization (PPO)

- Given that TRPO is relatively complicated, proximal policy optimization (PPO) simplifies it by using a **clipped surrogate objective** while retaining similar performance.
- Let's denote the probability ratio between old and new policies as:

$$r(\theta) = \frac{\pi_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)}$$

Proximal Policy Optimization (PPO)

- Given that TRPO is relatively complicated, proximal policy optimization (PPO) simplifies it by using a **clipped surrogate objective** while retaining similar performance.
- Let's denote the probability ratio between old and new policies as:

$$r(\theta) = \frac{\pi_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)}$$

- Then, the objective function of TRPO (on policy) becomes:

$$J^{\text{TRPO}}(\theta) = \mathbb{E}[r(\theta)\hat{A}_{\theta_{old}}(s, a)]$$

Proximal Policy Optimization (PPO)

- Given that TRPO is relatively complicated, proximal policy optimization (PPO) simplifies it by using a **clipped surrogate objective** while retaining similar performance.
- Let's denote the probability ratio between old and new policies as:

$$r(\theta) = \frac{\pi_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)}$$

- Then, the objective function of TRPO (on policy) becomes:

$$J^{\text{TRPO}}(\theta) = \mathbb{E}[r(\theta)\hat{A}_{\theta_{old}}(s, a)]$$

- Without a limitation on the distance between θ_{old} and θ , it would lead to instability with extremely large parameter updates and big policy ratios

Proximal Policy Optimization (PPO)

- Given that TRPO is relatively complicated, proximal policy optimization (PPO) simplifies it by using a **clipped surrogate objective** while retaining similar performance.
- Let's denote the probability ratio between old and new policies as:

$$r(\theta) = \frac{\pi_{\theta}(a|s)}{\pi_{\theta_{old}}(a|s)}$$

- Then, the objective function of TRPO (on policy) becomes:

$$J^{\text{TRPO}}(\theta) = \mathbb{E}[r(\theta)\hat{A}_{\theta_{old}}(s, a)]$$

- Without a limitation on the distance between θ_{old} and θ , it would lead to instability with extremely large parameter updates and big policy ratios
- PPO force $r(\theta)$ to stay within a small interval around 1 $\rightarrow [1 - \epsilon, 1 + \epsilon]$, where ϵ is hyperparameter

Proximal Policy Optimization (PPO)

$$J^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r(\theta) \hat{A}_{\theta_{\text{old}}}(s, a), \text{clip}(r(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_{\theta_{\text{old}}}(s, a) \right) \right]$$

θ is the policy parameter

\hat{E}_t is the empirical expectation over timesteps

r_t is the ratio of the probability under the new and old policies

\hat{A}_t is the estimated advantage at time t

ϵ is a hyperparameter, usually 0.1 or 0.2

- Function $\text{clip}(r(\theta), 1 - \epsilon, 1 + \epsilon)$ clips the ratio within $[1 - \epsilon, 1 + \epsilon]$
- The objective function of PPO takes the minimum one between the original value and the clipped version
- Thus we lose the motivation for increasing the policy update to extremes for better rewards

Proximal Policy Optimization (PPO)

$$J^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r(\theta) \hat{A}_{\theta_{\text{old}}}(s, a), \text{clip}(r(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_{\theta_{\text{old}}}(s, a) \right) \right]$$

θ is the policy parameter

\hat{E}_t is the empirical expectation over timesteps

r_t is the ratio of the probability under the new and old policies

\hat{A}_t is the estimated advantage at time t

ϵ is a hyperparameter, usually 0.1 or 0.2

- Function $\text{clip}(r(\theta), 1 - \epsilon, 1 + \epsilon)$ clips the ratio within $[1 - \epsilon, 1 + \epsilon]$
- The objective function of PPO takes the minimum one between the original value and the clipped version
- Thus we lose the motivation for increasing the policy update to extremes for better rewards

Proximal Policy Optimization (PPO)

$$J^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(\textcolor{blue}{r(\theta)\hat{A}_{\theta_{\text{old}}}(s, a)}, \textcolor{red}{\text{clip}(r(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_{\theta_{\text{old}}}(s, a)} \right) \right]$$

If $\hat{A}_t > 0$

Proximal Policy Optimization (PPO)

$$J^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(\textcolor{blue}{r(\theta) \hat{A}_{\theta_{\text{old}}}(s, a)}, \textcolor{red}{\text{clip}(r(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_{\theta_{\text{old}}}(s, a)} \right) \right]$$

If $\hat{A}_t > 0$

- The action is better than the average of all the actions in that state

Proximal Policy Optimization (PPO)

$$J^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r(\theta) \hat{A}_{\theta_{\text{old}}}(s, a), \text{clip}(r(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_{\theta_{\text{old}}}(s, a) \right) \right]$$

If $\hat{A}_t > 0$

- The action is better than the average of all the actions in that state
- Therefore, the action should be encouraged by increasing r_t , so that this action has a higher chance to be adopted

Proximal Policy Optimization (PPO)

$$J^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r(\theta) \hat{A}_{\theta_{\text{old}}}(s, a), \text{clip}(r(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_{\theta_{\text{old}}}(s, a) \right) \right]$$

If $\hat{A}_t > 0$

- The action is better than the average of all the actions in that state
- Therefore, the action should be encouraged by increasing r_t , so that this action has a higher chance to be adopted
- Increasing r_t implies increasing the new policy $\pi_\theta(a_t|s_t)$

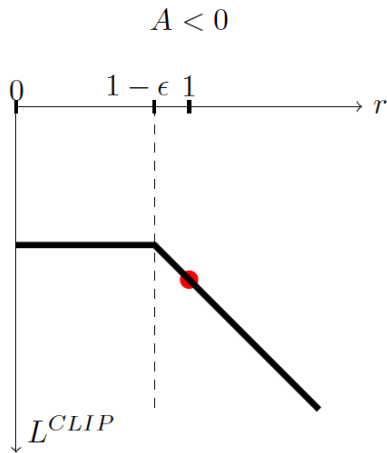
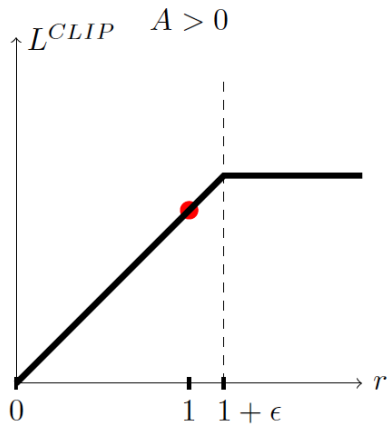
Proximal Policy Optimization (PPO)

$$J^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r(\theta) \hat{A}_{\theta_{\text{old}}}(s, a), \text{clip}(r(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_{\theta_{\text{old}}}(s, a) \right) \right]$$

If $\hat{A}_t > 0$

- The action is better than the average of all the actions in that state
- Therefore, the action should be encouraged by increasing r_t , so that this action has a higher chance to be adopted
- Increasing r_t implies increasing the new policy $\pi_{\theta}(a_t|s_t)$
- Because of the clip, r_t will only grows to as much as $1 + \epsilon$

Loss Function in PPO



Loss Function in PPO

Ratio

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

Advantage Function

$$A_t = Q(s_t, a_t) - V(s_t)$$

$$L^{CPIP}(\theta) = \hat{E}_t \left[\min(\underbrace{r_t(\theta)}_{\text{Normal Policy Gradient Objective}} \underbrace{\hat{A}_t}_{\text{Clipped version of Normal Policy Gradient Objective}}, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t) \right]$$

**Normal Policy
Gradient Objective**

**Clipped version of Normal
Policy Gradient Objective**

PPO with Clipped Objective

Algorithm 5 PPO with Clipped Objective

$$A_{\pi}(s, a) = Q_{\pi}(s, a) - V_{\pi}(s), \text{ where } a_t \sim \pi(a_t|s_t), s_{t+1} \sim P(s_{t+1}|s_t, a_t), \text{ for } t \geq 0$$

Input: initial policy parameters θ_0 , clipping threshold ϵ

for $k = 0, 1, 2, \dots$ **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

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

end for

- OpenAI Blog Article

Overview

Algorithm	Model	Policy	Action Space	Observation Space	Operator
Q-Learning	Model-Free	Off-policy	Discrete	Discrete	Q-value
SARSA	Model-Free	On-policy	Discrete	Discrete	Q-value
DQN	Model-Free	Off-policy	Discrete	Continuous	Q-value
DDPG	Model-Free	Off-policy	Continuous	Continuous	Q-value
TRPO	Model-Free	Off-policy	Continuous	Continuous	Advantage
PPO	Model-Free	Off-policy	Continuous	Continuous	Advantage