PPO

Proximal Policy Optimization Algorithms

Proximal Policy Optimization Algorithms

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov
OpenAI
{joschu, filip, prafulla, alec, oleg}@openai.com

Proximal Policy Optimization Algorithms
https://arxiv.org › cs
by J Schulman - 2017 - Cited by 1011 - Related articles
Jul 20, 2017 - The new methods, which we call proximal policy optimization (**PPO**), have some of the ... game playing, and we show that **PPO** outperforms other online policy gradient methods, ...
Which authors of this **paper** are endorsers?

Abstract

We propose a new family of policy gradient methods for reinforcement learning, which alternate between sampling data through interaction with the environment, and optimizing a "surrogate" objective function using stochastic gradient ascent. Whereas standard policy gradient methods perform one gradient update per data sample, we propose a novel objective function that enables multiple epochs of minibatch updates. The new methods, which we call proximal policy optimization (PPO), have some of the benefits of trust region policy optimization (TRPO), but they are much simpler to implement, more general, and have better sample complexity (empirically). Our experiments test PPO on a collection of benchmark tasks, including simulated robotic locomotion and Atari game playing, and we show that PPO outperforms other online policy gradient methods, and overall strikes a favorable balance between sample complexity, simplicity, and wall-time.

SOTA

Open ai

Openai는 일단 이것으로 시작한다고 함

cites 1011

Introduction

1 Introduction

In recent years, several different approaches have been proposed for reinforcement learning with neural network function approximators. The leading contenders are deep Q-learning [Mni+15], "vanilla" policy gradient methods [Mni+16], and trust region / natural policy gradient methods [Sch+15b]. However, there is room for improvement in developing a method that is scalable (to large models and parallel implementations), data efficient, and robust (i.e., successful on a variety of problems without hyperparameter tuning). Q-learning (with function approximation) fails on many simple problems[1] and is poorly understood, vanilla policy gradient methods have poor data effiency and robustness; and trust region policy optimization (TRPO) is relatively complicated, and is not compatible with architectures that include noise (such as dropout) or parameter sharing (between the policy and value function, or with auxiliary tasks).

This paper seeks to improve the current state of affairs by introducing an algorithm that attains the data efficiency and reliable performance of TRPO, while using only first-order optimization. We propose a novel objective with clipped probability ratios, which forms a pessimistic estimate (i.e., lower bound) of the performance of the policy. To optimize policies, we alternate between sampling data from the policy and performing several epochs of optimization on the sampled data.

Our experiments compare the performance of various different versions of the surrogate objective, and find that the version with the clipped probability ratios performs best. We also compare PPO to several previous algorithms from the literature. On continuous control tasks, it performs better than the algorithms we compare against. On Atari, it performs significantly better (in terms of sample complexity) than A2C and similarly to ACER though it is much simpler.

TPRO로 부터 시작함

TPRO는 복잡하고,
Second order로
풀었지만

PPO는 **Only first order optimization**으로 풀어서
더 간단하게 구현
가능하고
실증적인 논문이다.

empirically

Background

2.1 Policy Gradient Methods

Policy gradient methods work by computing an estimator of the policy gradient and plugging it into a stochastic gradient ascent algorithm. The most commonly used gradient estimator has the form

$$\hat{g} = \hat{\mathbb{E}}_t\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\hat{A}_t\right] \tag{1}$$

where π_θ is a stochastic policy and $\hat{A}_t$ is an estimator of the advantage function at timestep t. Here, the expectation $\hat{\mathbb{E}}_t[\ldots]$ indicates the empirical average over a finite batch of samples, in an algorithm that alternates between sampling and optimization. Implementations that use automatic differentiation software work by constructing an objective function whose gradient is the policy gradient estimator; the estimator $\hat{g}$ is obtained by differentiating the objective

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t\left[\log \pi_\theta(a_t \mid s_t)\hat{A}_t\right]. \tag{2}$$

While it is appealing to perform multiple steps of optimization on this loss L^{PG} using the same trajectory, doing so is not well-justified, and empirically it often leads to destructively large policy updates (see Section 6.1; results are not shown but were similar or worse than the "no clipping or penalty" setting).

Policy Gradient 식

Vanilla Policy Method라고도 함

Gradient Log 파이 곱하기 Avantage

Advantage대신 TD error를 쓸수도 있음

Automatic differentiation software(Tensorflow) 같은거를 쓰면 자동으로 미분해 주고,
Loss함수를 (2)로 표현 가능함
L pg -> policy gradient의 Loss임

근데 이것을 그대로 **weight update**를 하면 **weight**가 불안정

empirically

Trust Region Methods

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t\left[\log \pi_\theta(a_t \mid s_t)\hat{A}_t\right]. \tag{2}$$

- Equivalently differentiate

$$L^{IS}_{\theta_{\text{old}}}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right].$$

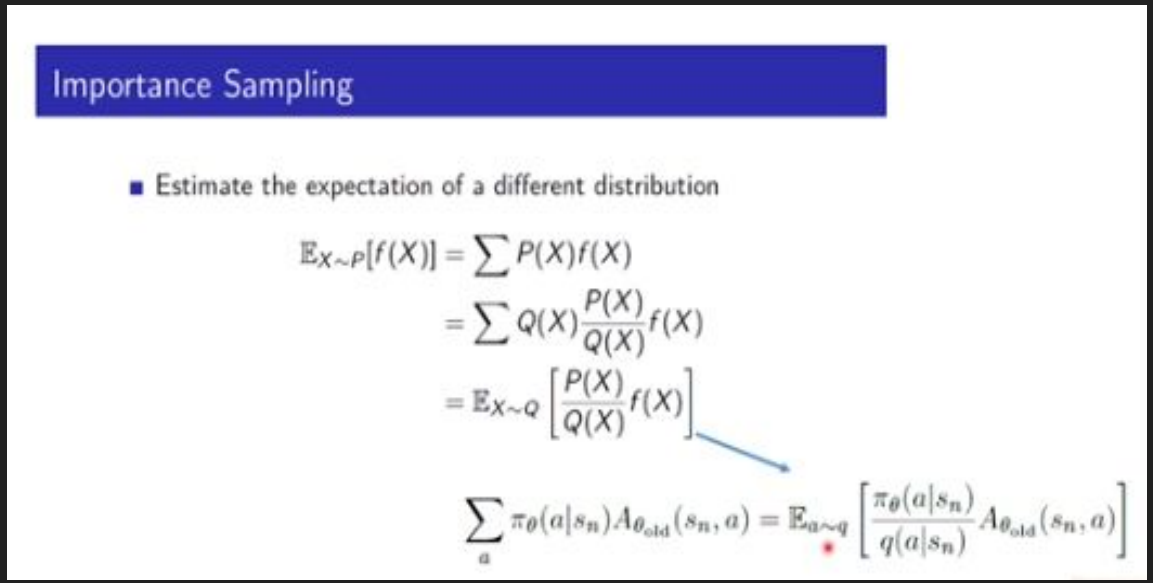
at $\theta = \theta_{\text{old}}$, state-actions are sampled using θ_{old}. (IS = importance sampling)

Just the chain rule: $\nabla_\theta \log f(\theta)\big|_{\theta_{\text{old}}} = \frac{\nabla_\theta f(\theta)\big|_{\theta_{\text{old}}}}{f(\theta_{\text{old}})} = \nabla_\theta\left(\frac{f(\theta)}{f(\theta_{\text{old}})}\right)\Big|_{\theta_{\text{old}}}$

L pg를 Important sampling를 통해서 L is로 변경함

다음장에 important sampling 설명

Important Sampling 복습



어떤 X가 p를 따를때 그 X값을 함수 f에 넣은 값의 기대값을 구하고 싶다.

주사위를 예로 ⅙ * 1 + ⅙ * 2 .....

뭔가 다른 확률을 따르는 X를 넣고 싶다면

Q(X) 분모, 분자에 등장

시그마를 다시 기대값으로 바꾸면

X는 Q를 따르는 Q분의 P 확률에다가 함수 f(X)의 기대값으로 변경가능

한번더 Important sampling

좀더 쉽게...

$$U(\theta) = \mathbb{E}_{\tau\sim\theta_{\text{old}}}\left[\frac{P(\tau|\theta)}{P(\tau|\theta_{\text{old}})}R(\tau)\right]$$

$$\nabla_\theta U(\theta) = \mathbb{E}_{\tau\sim\theta_{\text{old}}}\left[\frac{\nabla_\theta P(\tau|\theta)}{P(\tau|\theta_{\text{old}})}R(\tau)\right]$$

$$\nabla_\theta\, U(\theta)|_{\theta=\theta_{\text{old}}} = \mathbb{E}_{\tau\sim\theta_{\text{old}}}\left[\frac{\nabla_\theta\, P(\tau|\theta)|_{\theta_{\text{old}}}}{P(\tau|\theta_{\text{old}})}R(\tau)\right]$$

$$= \mathbb{E}_{\tau\sim\theta_{\text{old}}}\left[\nabla_\theta \log P(\tau|\theta)|_{\theta_{\text{old}}} R(\tau)\right]$$

U 세타가 타우가 올드세타를 따를때
타우가 올드세타를 따를때 확률
타우가 세타를 따를때 확률에 어떤 함수
R에다 타우를 넣었을때의 기대값

세타로 미분하고, 세타는 분자에 밖에 없고

그걸 정리하면 맨 마지막 식이 됨
(1/x을 미분하면 log x)

어쨋든 맨 밑에 있는 식이 결국 policy gradient식이고, 맨 위에 식으로 대체할수 있다.

맨 위의 식을 미분 시키면 된다.

TRPO

2.2 Trust Region Methods

In TRPO [Sch+15b], an objective function (the "surrogate" objective) is maximized subject to a constraint on the size of the policy update. Specifically,

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] \tag{3}$$

$$\text{subject to} \quad \hat{\mathbb{E}}_t[\mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \le \delta. \tag{4}$$

Constraint 형태

Here, θ_{old} is the vector of policy parameters before the update. This problem can efficiently be approximately solved using the conjugate gradient algorithm, after making a linear approximation to the objective and a quadratic approximation to the constraint.

The theory justifying TRPO actually suggests using a penalty instead of a constraint, i.e., solving the unconstrained optimization problem

Penalty 형태

$$\underset{\theta}{\text{maximize}}\, \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t - \beta\,\mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]\right] \tag{5}$$

TRPO는 그냥 미분 시켜서 업데이트 시키면 안되고, 어쨌든 변화 시키는 량을 KL Divergence를 이용해서 거리가 델타 이상 벌어지지 않도록 컨스트레인트 형태로 풀수도 있지만 마이너스 하는 패널티 형태로도 풀수 있고 두 식은 같다.

Clipped Surrogate Objective

$$\underset{\theta}{\text{maximize}}\, \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t - \beta\,\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]\right] \tag{5}$$

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)},\ \text{so}\ r(\theta_{\text{old}}) = 1.$$

3 Clipped Surrogate Objective

Let $r_t(\theta)$ denote the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$, so $r(\theta_{\text{old}}) = 1$. TRPO maximizes a "surrogate" objective

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right]. \tag{6}$$

식이 복잡하니깐 r을 파이세타 올드 분에 파이세타로 변경함

r은 올드 세타의 행동과, 현재 세타 행동의 비율이고, 세타에 세타 올드를 넣으면 1이다.

L cpi는 r 곱하기 a 이다

Clipped Surrogate Objective

The superscript CPI refers to conservative policy iteration [KL02], where this objective was proposed. Without a constraint, maximization of L^{CPI} would lead to an excessively large policy update; hence, we now consider how to modify the objective, to penalize changes to the policy that move $r_t(\theta)$ away from 1.

The main objective we propose is the following:

Original loss Clipped loss

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right] \quad (7)$$

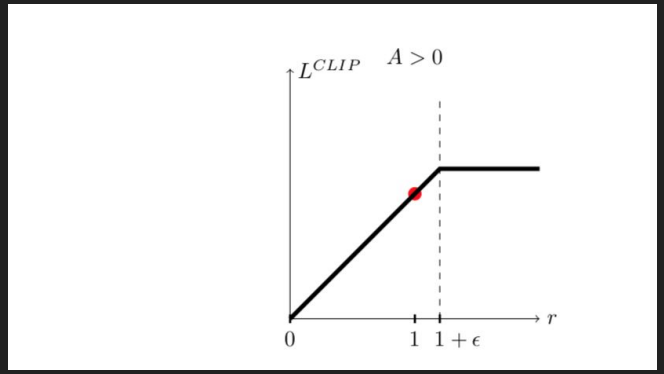
TRPO에서 부터 계속 되온 이야기, 어쨋든 많이 업데이트를 하면 안된다. 그래서 contraint, KL divergence의 차이로 베타보다 작게, 이번에는 r을 사용해서 clip을 한다.

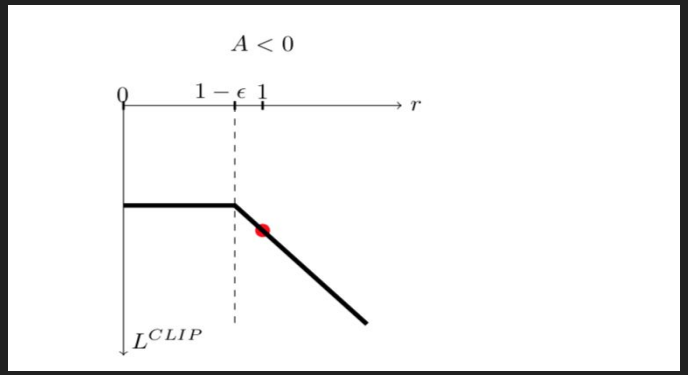
clip은 어떤 구간 내에 있도록 잘라주는 함수이다.

즉 r곱하기 a를 구하거나, r을 1-입실론, 1+ 입실론 사이로 clip을 해서 a를 곱한값중에 작은 값을 구하는 함수는 L CLIP이라고 한다.

오리지날 loss가 있고 clipped loss가 있고 둘중에 작은것을 쓰자

Clipped Surrogate Objective





r을 policy의 비율인데, A가 0 보다 크다면
빨간점은 시작점이고, 보다 좋은 방향인 양의 방향으로 r을 변경해주는데 너무 많이 변경하면 안되니깐 1+입실론 까지만 변경함

빨간점은 우리의 시작점 A가 0보다 작으면 r을 이전보다 적게 해야함.
r이 작아지는데 1-입실론까지만 작게 함

Algorithm

Algorithm 1 PPO, Actor-Critic Style

for iteration=$1, 2, \ldots$ **do**
  for actor=$1, 2, \ldots, N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
end for

iteration을 반복하고,
N개의 actor가 있음

actor가 policy 파이 올드 세타로
환경에서 T만큼 실행함

모든 actor들의 advantage를
계산함

그리고 그냥 loss함수 미분후
업데이트

근데 한번만 하지 않고 K번
반복하여 weight를 업데이트 함

Adaptive KL penalty Coefficient

4 Adaptive KL Penalty Coefficient

Another approach, which can be used as an alternative to the clipped surrogate objective, or in addition to it, is to use a penalty on KL divergence, and to adapt the penalty coefficient so that we achieve some target value of the KL divergence d_{targ} each policy update. In our experiments, we found that the KL penalty performed worse than the clipped surrogate objective, however, we've included it here because it's an important baseline.

In the simplest instantiation of this algorithm, we perform the following steps in each policy update:

- Using several epochs of minibatch SGD, optimize the KL-penalized objective

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t - \beta\,\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]\right] \quad (8)$$

TRPO에서 나온 식

- Compute $d = \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]]$
  - If $d < d_{\text{targ}}/1.5$, $\beta \leftarrow \beta/2$
  - If $d > d_{\text{targ}} \times 1.5$, $\beta \leftarrow \beta \times 2$

실험에서는 첫번째 clipped방법이 더 좋음

KLPEN(KL divergence penalty)

베타를 뭔가 계속 바꿔줌

KL divergence가 큰다는 것은 policy가 베타를 키우고, KL작으면 베타를 줄인다.

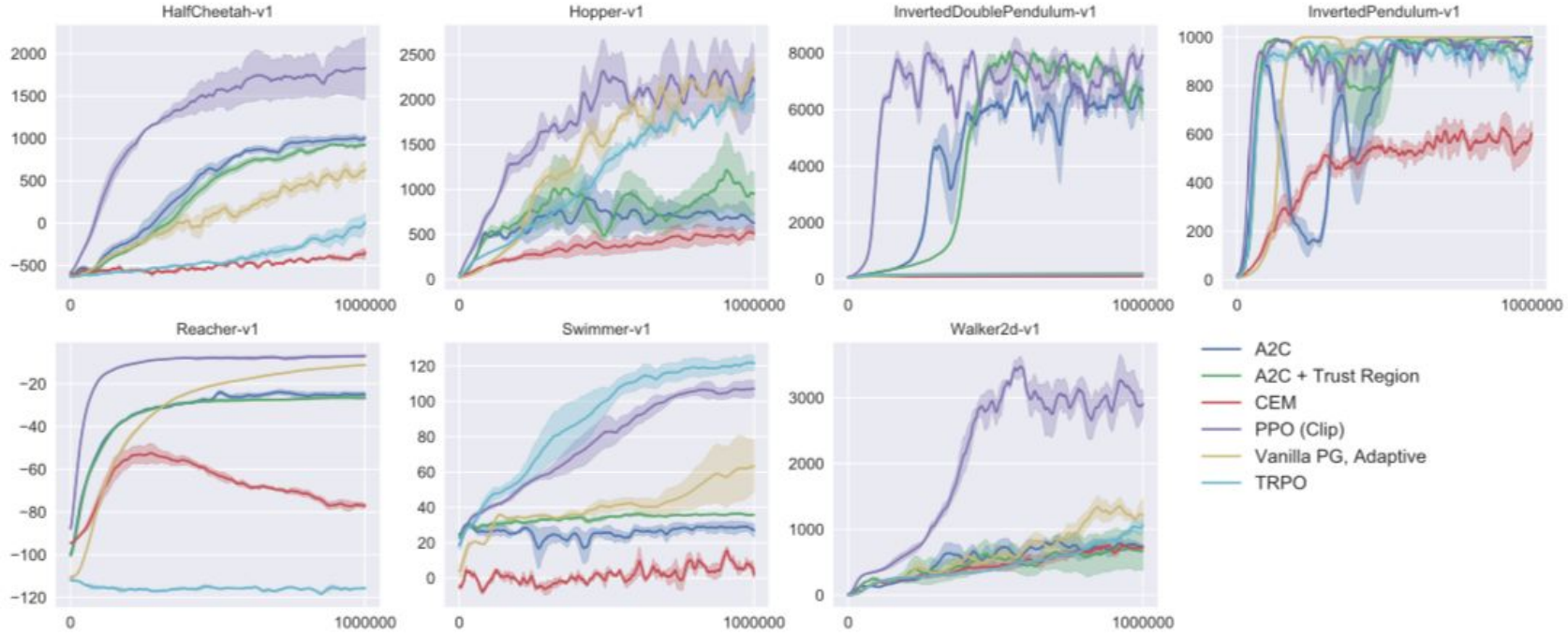
KL이 크다는 것은 policy가 많이 바꼈다는것이고 패널티를 크게 줘서 업데이트가 적게 되야함

KL이 작다는것은 베타를 줄여줘서 업데이트에 최선을 다해라는 뜻

Experiments

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

Experiments



Experiments

	A2C	ACER	PPO	Tie
(1) avg. episode reward over all of training	1	18	**30**	0
(2) avg. episode reward over last 100 episodes	1	**28**	19	1

(1)은 학습이 진행되는 모든 시간에 대한 평균
(2)마지막 100개의 episodes에 대한 평균

최종 훈련후 ppo랑 비교해보면 acer가 더 높음 (잉?). 그러나 ppo는 훈련이 빨리 됨

끝