Reinforcement Learning Opensource to Baselines

Member

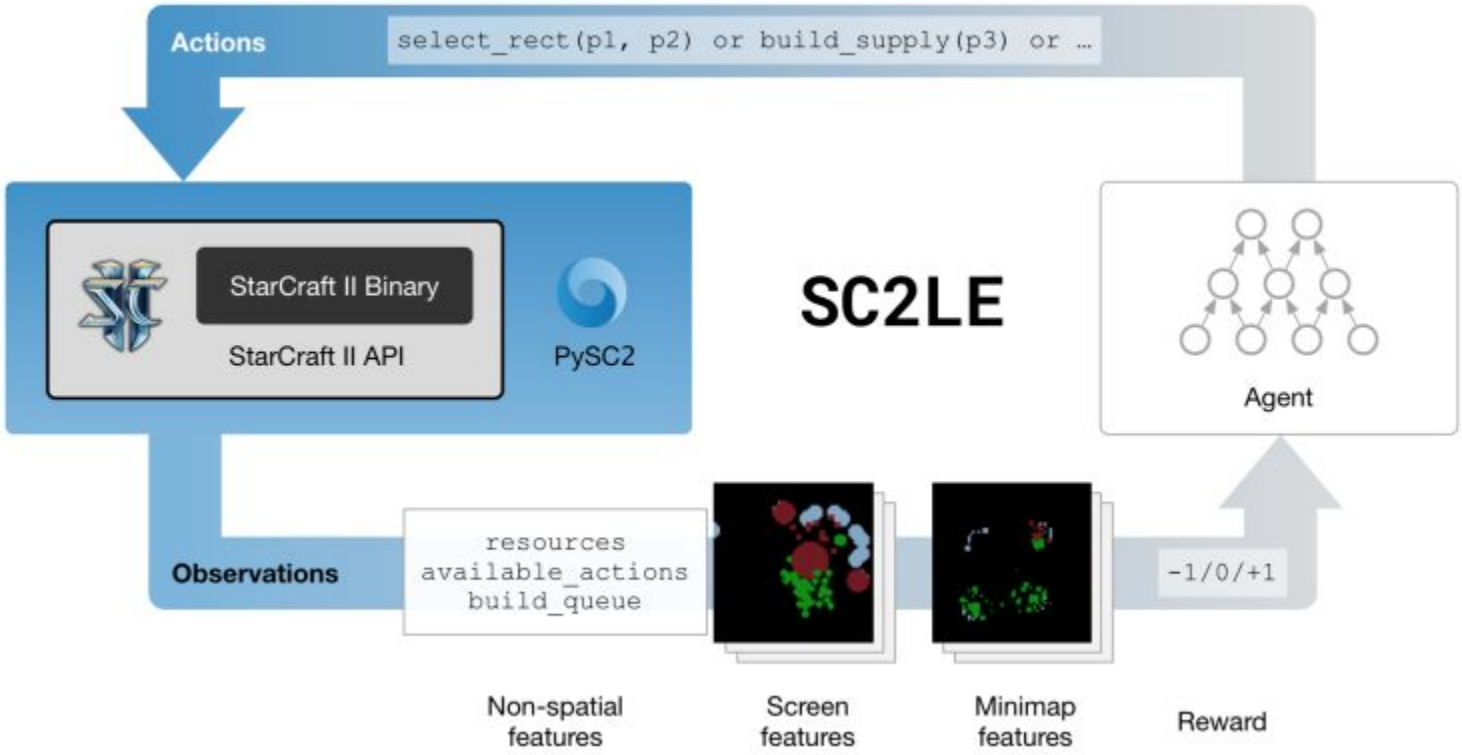
- 한국생산기술연구원
  - 로보틱스 및 재활로봇 연구
- KAIST 연구원
  - IoT 시스템 개발 & 강화학습 연구
- 플랜아이 연구원
  - 추천 시스템 & 강화학습 연구

- 한양대학교 석사과정
  - 강화학습 시뮬레이션 구축
  - 속도 예측 연구

- 호서대학교 로봇 자동화공학과 학부생
  - 임베디드, ROS, 제어, 강화학습
  - 강화학습을 로봇에 적용하는 연구

Index

Reinforcement Learning



Policy base Reinforcement Learning

Policy base Reinforcement Learning

$$\pi(s,a)$$

Policy base Reinforcement Learning

$$\pi(s,a)$$

Probability to select action(a) at state(s)

Policy base Reinforcement Learning

$$J(\theta) = \sum_{s} d^{\pi_\theta}(s) \sum_{a} \pi_\theta(s,a) R_a^s$$

Policy base Reinforcement Learning

$$J(\theta) = \sum_{s} d^{\pi_\theta}(s) \sum_{a} \pi_\theta(s,a) R_a^s$$

State Distribution

Policy base Reinforcement Learning

Policy

$$J(\theta) = \sum_{s} d^{\pi_\theta}(s) \sum_{a} \pi_\theta(s,a) R_a^s$$

State Distribution

Policy base Reinforcement Learning

Policy

$$J(\theta) = \sum_{s} d^{\pi_\theta}(s) \sum_{a} \pi_\theta(s,a) R_a^s$$

State Distribution

Reward

Policy base Reinforcement Learning

$$Maximize J(\theta)$$

Policy base Reinforcement Learning

$$J(\theta) = \sum_{s} d^{\pi_\theta}(s) \sum_{a} \pi_\theta(s,a) R_a^s$$

Policy base Reinforcement Learning

$$J(\theta) = \sum_{s} d^{\pi_\theta}(s) \sum_{a} \pi_\theta(s,a) R_a^s$$

Policy base Reinforcement Learning

$$
\begin{aligned}
J(\theta) &= E_{\pi_\theta}[r] \\
&= \sum_{s \subseteq S} d(s) \sum_{a \subseteq A} \pi(s,a) R_a^s
\end{aligned}
$$

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \sum_{s \subseteq S} d(s) \sum_{a \subseteq A} \pi(s,a) R_a^s \\
&= \sum_{s \subseteq S} d(s) \sum_{a \subseteq A} \nabla_\theta \pi(s,a) R_a^s \\
&= \sum_{s \subseteq S} d(s) \sum_{a \subseteq A} \nabla_\theta \pi(s,a) (log(\pi(s,a) R_a^s))
\end{aligned}
$$

Policy base Reinforcement Learning

$$J(\theta) = E_{\pi_\theta}[r] = \sum_{s \subseteq S} d(s) \sum_{a \subseteq A} \pi(s,a) R_a^s$$

$$\frac{d(\log x)}{dx} = \frac{1}{x}$$

$$\frac{d(\log f(x))}{dx} = \frac{f'(x)}{f(x)}$$

$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta \sum_{s \subseteq S} d(s) \sum_{a \subseteq A} \pi(s,a) R_a^s \\ &= \sum_{s \subseteq S} d(s) \sum_{a \subseteq A} \nabla_\theta \pi(s,a) R_a^s \\ &= \sum_{s \subseteq S} d(s) \sum_{a \subseteq A} \nabla_\theta \pi(s,a) (\log(\pi(s,a) R_a^s)) \end{aligned}$$

Policy base Reinforcement Learning

$$J(\theta) = E_{\pi_\theta}[r] = \sum_{s \subseteq S} d(s) \sum_{a \subseteq A} \pi(s,a) R_a^s$$

$$\frac{d(\log x)}{dx} = \frac{1}{x}$$

$$\frac{d(\log f(x))}{dx} = \frac{f'(x)}{f(x)}$$

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \sum_{s \subseteq S} d(s) \sum_{a \subseteq A} \pi(s,a) R_a^s \\
&= \sum_{s \subseteq S} d(s) \sum_{a \subseteq A} \nabla_\theta \pi(s,a) R_a^s \\
&= \sum_{s \subseteq S} d(s) \sum_{a \subseteq A} \nabla_\theta \pi(s,a) (\log(\pi(s,a) R_a^s)) = E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) R_a^s]
\end{aligned}$$

Policy base Reinforcement Learning

$$E_{\pi_\theta}[\nabla_\theta log \pi_\theta(s,a) R_a^s]$$

Policy base Reinforcement Learning

function REINFORCE
    Initialise θ arbitrarily
    for each episode $\{s_1, a_1, r_2, ..., s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**
        for $t = 1$ to $T - 1$ **do**
            $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$
        end for
    end for
    return θ
end function

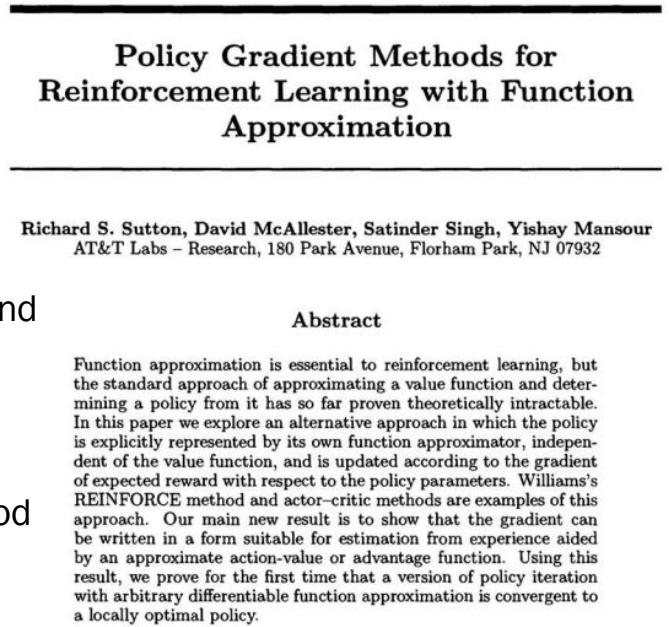
Policy base Reinforcement Learning

- Policy Gradient
- Advantage Actor Critic
- Natural Policy Gradient
- Trust Region Policy Gradient
- Proximal Policy Gradient

Policy Gradient

- Policy Gradient Methods for Reinforcement Learning with Function Approximation
  - Policy can be trained by iterable method
  - Policy can be trained by Policy Gradient Method

- Policy Gradient
  - Optimal Policy can be obtained by Gradient Ascend Method
- Iterable Method
  - Optimal Policy Can be obtained by iterable method like deep learning method



Policy Gradient Methods for Reinforcement Learning with Function Approximation

Richard S. Sutton, David McAllester, Satinder Singh, Yishay Mansour
AT&T Labs – Research, 180 Park Avenue, Florham Park, NJ 07932

Abstract

Function approximation is essential to reinforcement learning, but the standard approach of approximating a value function and determining a policy from it has so far proven theoretically intractable. In this paper we explore an alternative approach in which the policy is explicitly represented by its own function approximator, independent of the value function, and is updated according to the gradient of expected reward with respect to the policy parameters. Williams's REINFORCE method and actor–critic methods are examples of this approach. Our main new result is to show that the gradient can be written in a form suitable for estimation from experience aided by an approximate action-value or advantage function. Using this result, we prove for the first time that a version of policy iteration with arbitrary differentiable function approximation is convergent to a locally optimal policy.

Policy Gradient

$$J(\theta) = \sum_{s} d^{\pi_\theta}(s) \sum_{a} \pi_\theta(s,a) R_a^s$$

Advantage Actor Critic

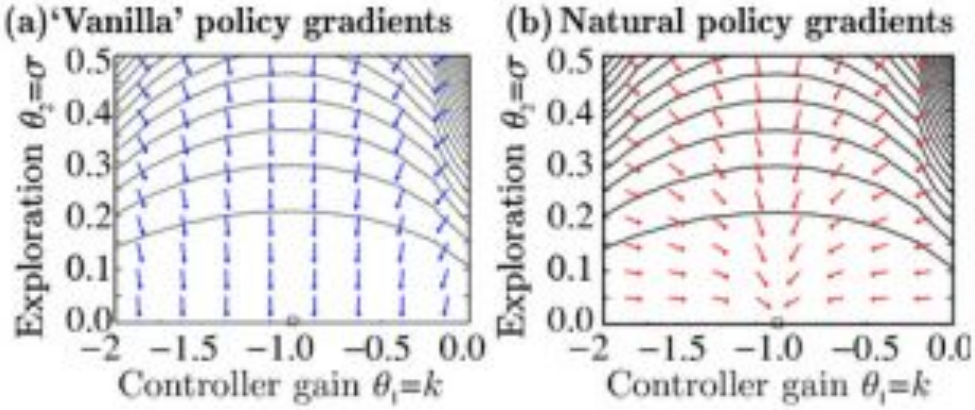
- Why advantage is needed?
  - Reduce variance, without changing expectation
  - $A(s,a)=Q(s,a)-V(s)$
  - $Q(s,a)=E[R_{t+1}+\gamma max_{a'}Q(s_{t+1},a')|s_t=s,a_t=a]$
  - $Q(s,a)=R_{t+1}+\gamma V(s_{t+1})$

$$\nabla_\theta J(\theta)=E[\nabla_\theta log\pi(s,a)A(s,a)]$$

$$(R_a^s+\gamma V(s_{t+1})-V(s_t))^2$$

Natural Policy Gradient

- The parameter update can not guarantee
  the performance improvement of the actual function



A Natural Policy Gradient

Sham Kakade
Gatsby Computational Neuroscience Unit
17 Queen Square, London, UK WC1N 3AR
http://www.gatsby.ucl.ac.uk
sham@gatsby.ucl.ac.uk

Abstract

We provide a natural gradient method that represents the steepest descent direction based on the underlying structure of the parameter space. Although gradient methods cannot make large changes in the values of the parameters, we show that the natural gradient is moving toward choosing a greedy optimal action rather than just a better action. These greedy optimal actions are those that would be chosen under one improvement step of policy iteration with approximate, *compatible* value functions, as defined by Sutton *et al.* [9]. We then show drastic performance improvements in simple MDPs and in the more challenging MDP of Tetris.

Trust Region Policy Optimization

- Trust Region

$$\max_{\theta} \quad E_{s\sim\rho_{\theta_{\text{old}}},a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta_{\text{old}}}(s,a)\right]$$
$$\text{s.t.} \quad E_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}\left(\pi_{\theta_{\text{old}}}(\cdot|s)\parallel\pi_\theta(\cdot|s)\right)\right]\le\delta$$

Trust Region Policy Optimization

John Schulman JOSCHU@EECS.BERKELEY.EDU
Sergey Levine SLEVINE@EECS.BERKELEY.EDU
Philipp Moritz PCMORITZ@EECS.BERKELEY.EDU
Michael Jordan JORDAN@CS.BERKELEY.EDU
Pieter Abbeel PABBEEL@CS.BERKELEY.EDU
University of California, Berkeley, Department of Electrical Engineering and Computer Sciences

Abstract

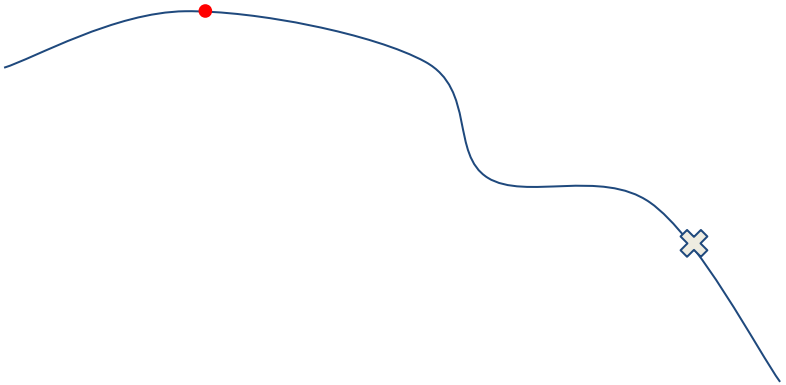
We describe an iterative procedure for optimizing policies, with guaranteed monotonic improvement. By making several approximations to the theoretically-justified procedure, we develop a practical algorithm, called Trust Region Policy Optimization (TRPO). This algorithm is similar to natural policy gradient methods and is effective for optimizing large nonlinear policies such as neural networks. Our experiments demonstrate its robust performance on a wide variety of tasks: learning simulated robotic swimming, hopping, and walking gaits; and playing Atari games using images of the screen as input. Despite its approximations that deviate from the theory, TRPO tends to give monotonic improvement, with little tuning of hyperparameters.

Tetris is a classic benchmark problem for approximate dynamic programming (ADP) methods, stochastic optimization methods are difficult to beat on this task (Gabillon et al., 2013). For continuous control problems, methods like CMA have been successful at learning control policies for challenging tasks like locomotion when provided with hand-engineered policy classes with low-dimensional parameterizations (Wampler & Popović, 2009). The inability of ADP and gradient-based methods to consistently beat gradient-free random search is unsatisfying, since gradient-based optimization algorithms enjoy much better sample complexity guarantees than gradient-free methods (Nemirovski, 2005). Continuous gradient-based optimization has been very successful at learning function approximators for supervised learning tasks with huge numbers of parameters, and extending their success to reinforcement learning would allow for efficient training of complex and powerful policies.

Trust Region Policy Optimization

- Trust Region

$$\max_{\theta} \quad E_{s\sim\rho_{\theta_{\text{old}}},a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta_{\text{old}}}(s,a)\right]$$
$$\text{s.t.} \quad E_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}\left(\pi_{\theta_{\text{old}}}(\cdot|s)\parallel\pi_\theta(\cdot|s)\right)\right]\le\delta$$



Trust Region Policy Optimization

John Schulman JOSCHU@EECS.BERKELEY.EDU
Sergey Levine SLEVINE@EECS.BERKELEY.EDU
Philipp Moritz PCMORITZ@EECS.BERKELEY.EDU
Michael Jordan JORDAN@CS.BERKELEY.EDU
Pieter Abbeel PABBEEL@CS.BERKELEY.EDU
University of California, Berkeley, Department of Electrical Engineering and Computer Sciences

Abstract

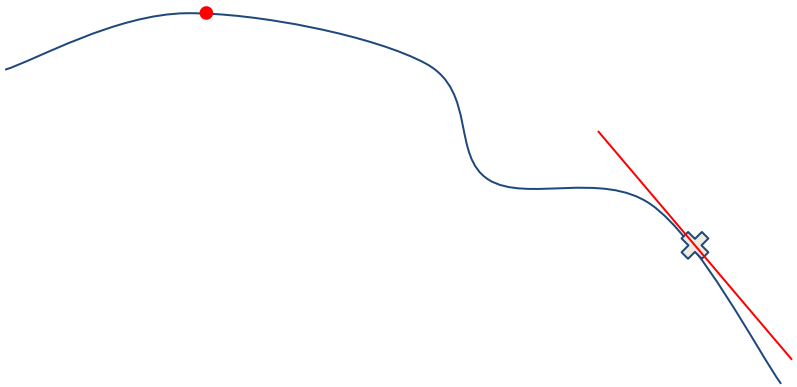
We describe an iterative procedure for optimizing policies, with guaranteed monotonic improvement. By making several approximations to the theoretically-justified procedure, we develop a practical algorithm, called Trust Region Policy Optimization (TRPO). This algorithm is similar to natural policy gradient methods and is effective for optimizing large nonlinear policies such as neural networks. Our experiments demonstrate its robust performance on a wide variety of tasks: learning simulated robotic swimming, hopping, and walking gaits; and playing Atari games using images of the screen as input. Despite its approximations that deviate from the theory, TRPO tends to give monotonic improvement, with little tuning of hyperparameters.

Tetris is a classic benchmark problem for approximate dynamic programming (ADP) methods, stochastic optimization methods are difficult to beat on this task (Gabillon et al., 2013). For continuous control problems, methods like CMA have been successful at learning control policies for challenging tasks like locomotion when provided with hand-engineered policy classes with low-dimensional parameterizations (Wampler & Popović, 2009). The inability of ADP and gradient-based methods to consistently beat gradient-free random search is unsatisfying, since gradient-based optimization algorithms enjoy much better sample complexity guarantees than gradient-free methods (Nemirovski, 2005). Continuous gradient-based optimization has been very successful at learning function approximators for supervised learning tasks with huge numbers of parameters, and extending their success to reinforcement learning would allow for efficient training of complex and powerful policies.

Trust Region Policy Optimization

- Trust Region

$$\max_{\theta} \quad E_{s\sim\rho_{\theta_{\text{old}}},a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta_{\text{old}}}(s,a)\right]$$
$$\text{s.t.} \quad E_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}\left(\pi_{\theta_{\text{old}}}(\cdot|s)\parallel\pi_\theta(\cdot|s)\right)\right]\le\delta$$



Trust Region Policy Optimization

John Schulman JOSCHU@EECS.BERKELEY.EDU
Sergey Levine SLEVINE@EECS.BERKELEY.EDU
Philipp Moritz PCMORITZ@EECS.BERKELEY.EDU
Michael Jordan JORDAN@CS.BERKELEY.EDU
Pieter Abbeel PABBEEL@CS.BERKELEY.EDU
University of California, Berkeley, Department of Electrical Engineering and Computer Sciences

Abstract

We describe an iterative procedure for optimizing policies, with guaranteed monotonic improvement. By making several approximations to the theoretically-justified procedure, we develop a practical algorithm, called Trust Region Policy Optimization (TRPO). This algorithm is similar to natural policy gradient methods and is effective for optimizing large nonlinear policies such as neural networks. Our experiments demonstrate its robust performance on a wide variety of tasks: learning simulated robotic swimming, hopping, and walking gaits; and playing Atari games using images of the screen as input. Despite its approximations that deviate from the theory, TRPO tends to give monotonic improvement, with little tuning of hyperparameters.

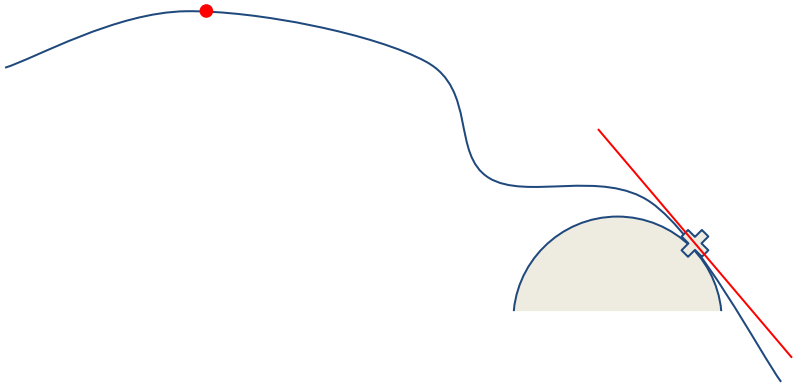
Tetris is a classic benchmark problem for approximate dynamic programming (ADP) methods, stochastic optimization methods are difficult to beat on this task (Gabillon et al., 2013). For continuous control problems, methods like CMA have been successful at learning control policies for challenging tasks like locomotion when provided with hand-engineered policy classes with low-dimensional parameterizations (Wampler & Popović, 2009). The inability of ADP and gradient-based methods to consistently beat gradient-free random search is unsatisfying, since gradient-based optimization algorithms enjoy much better sample complexity guarantees than gradient-free methods (Nemirovski, 2005). Continuous gradient-based optimization has been very successful at learning function approximators for supervised learning tasks with huge numbers of parameters, and extending their success to reinforcement learning would allow for efficient training of complex and powerful policies.

Trust Region Policy Optimization

- Trust Region

$$\max_{\theta} \quad E_{s\sim\rho_{\theta_{\text{old}}},a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta_{\text{old}}}(s,a)\right]$$

$$\text{s.t.} \quad E_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}\left(\pi_{\theta_{\text{old}}}(\cdot|s)\parallel\pi_\theta(\cdot|s)\right)\right]\le\delta$$



Trust Region Policy Optimization

John Schulman JOSCHU@EECS.BERKELEY.EDU
Sergey Levine SLEVINE@EECS.BERKELEY.EDU
Philipp Moritz PCMORITZ@EECS.BERKELEY.EDU
Michael Jordan JORDAN@CS.BERKELEY.EDU
Pieter Abbeel PABBEEL@CS.BERKELEY.EDU
University of California, Berkeley, Department of Electrical Engineering and Computer Sciences

Abstract

We describe an iterative procedure for optimizing policies, with guaranteed monotonic improvement. By making several approximations to the theoretically-justified procedure, we develop a practical algorithm, called Trust Region Policy Optimization (TRPO). This algorithm is similar to natural policy gradient methods and is effective for optimizing large nonlinear policies such as neural networks. Our experiments demonstrate its robust performance on a wide variety of tasks: learning simulated robotic swimming, hopping, and walking gaits; and playing Atari games using images of the screen as input. Despite its approximations that deviate from the theory, TRPO tends to give monotonic improvement, with little tuning of hyperparameters.

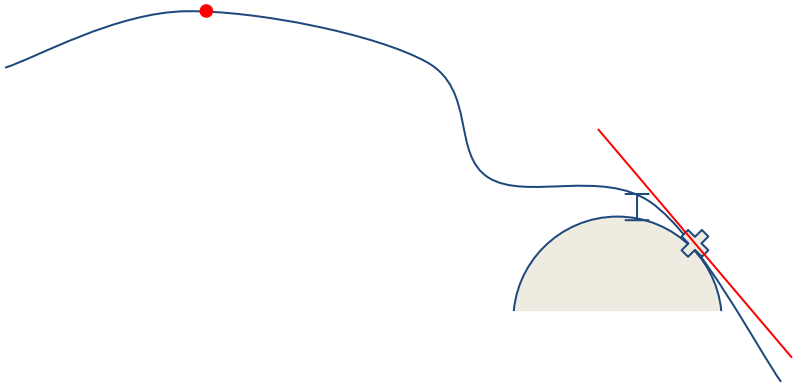
Tetris is a classic benchmark problem for approximate dynamic programming (ADP) methods, stochastic optimization methods are difficult to beat on this task (Gabillon et al., 2013). For continuous control problems, methods like CMA have been successful at learning control policies for challenging tasks like locomotion when provided with hand-engineered policy classes with low-dimensional parameterizations (Wampler & Popović, 2009). The inability of ADP and gradient-based methods to consistently beat gradient-free random search is unsatisfying, since gradient-based optimization algorithms enjoy much better sample complexity guarantees than gradient-free methods (Nemirovski, 2005). Continuous gradient-based optimization has been very successful at learning function approximators for supervised learning tasks with huge numbers of parameters, and extending their success to reinforcement learning would allow for efficient training of complex and powerful policies.

Trust Region Policy Optimization

- Trust Region

$$\max_{\theta} \quad E_{s\sim\rho_{\theta_{\text{old}}},a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta_{\text{old}}}(s,a)\right]$$

$$\text{s.t.} \quad E_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}\left(\pi_{\theta_{\text{old}}}(\cdot|s)\parallel\pi_\theta(\cdot|s)\right)\right]\le\delta$$



Trust Region Policy Optimization

John Schulman JOSCHU@EECS.BERKELEY.EDU
Sergey Levine SLEVINE@EECS.BERKELEY.EDU
Philipp Moritz PCMORITZ@EECS.BERKELEY.EDU
Michael Jordan JORDAN@CS.BERKELEY.EDU
Pieter Abbeel PABBEEL@CS.BERKELEY.EDU
University of California, Berkeley, Department of Electrical Engineering and Computer Sciences

Abstract

We describe an iterative procedure for optimizing policies, with guaranteed monotonic improvement. By making several approximations to the theoretically-justified procedure, we develop a practical algorithm, called Trust Region Policy Optimization (TRPO). This algorithm is similar to natural policy gradient methods and is effective for optimizing large nonlinear policies such as neural networks. Our experiments demonstrate its robust performance on a wide variety of tasks: learning simulated robotic swimming, hopping, and walking gaits; and playing Atari games using images of the screen as input. Despite its approximations that deviate from the theory, TRPO tends to give monotonic improvement, with little tuning of hyperparameters.

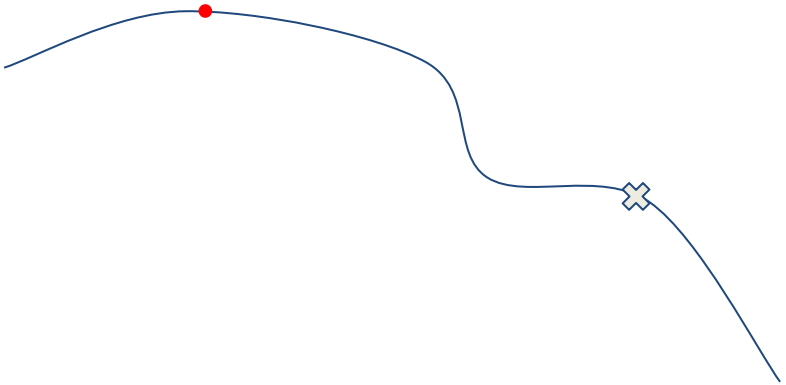
Tetris is a classic benchmark problem for approximate dynamic programming (ADP) methods, stochastic optimization methods are difficult to beat on this task (Gabillon et al., 2013). For continuous control problems, methods like CMA have been successful at learning control policies for challenging tasks like locomotion when provided with hand-engineered policy classes with low-dimensional parameterizations (Wampler & Popović, 2009). The inability of ADP and gradient-based methods to consistently beat gradient-free random search is unsatisfying, since gradient-based optimization algorithms enjoy much better sample complexity guarantees than gradient-free methods (Nemirovski, 2005). Continuous gradient-based optimization has been very successful at learning function approximators for supervised learning tasks with huge numbers of parameters, and extending their success to reinforcement learning would allow for efficient training of complex and powerful policies.

Trust Region Policy Optimization

- Trust Region

$$\max_{\theta} \quad E_{s\sim\rho_{\theta_{\text{old}}},a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta_{\text{old}}}(s,a)\right]$$

$$\text{s.t.} \quad E_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}\left(\pi_{\theta_{\text{old}}}(\cdot|s)\parallel\pi_\theta(\cdot|s)\right)\right]\le\delta$$



Trust Region Policy Optimization

John Schulman JOSCHU@EECS.BERKELEY.EDU
Sergey Levine SLEVINE@EECS.BERKELEY.EDU
Philipp Moritz PCMORITZ@EECS.BERKELEY.EDU
Michael Jordan JORDAN@CS.BERKELEY.EDU
Pieter Abbeel PABBEEL@CS.BERKELEY.EDU
University of California, Berkeley, Department of Electrical Engineering and Computer Sciences

Abstract

We describe an iterative procedure for optimizing policies, with guaranteed monotonic improvement. By making several approximations to the theoretically-justified procedure, we develop a practical algorithm, called Trust Region Policy Optimization (TRPO). This algorithm is similar to natural policy gradient methods and is effective for optimizing large nonlinear policies such as neural networks. Our experiments demonstrate its robust performance on a wide variety of tasks: learning simulated robotic swimming, hopping, and walking gaits; and playing Atari games using images of the screen as input. Despite its approximations that deviate from the theory, TRPO tends to give monotonic improvement, with little tuning of hyperparameters.

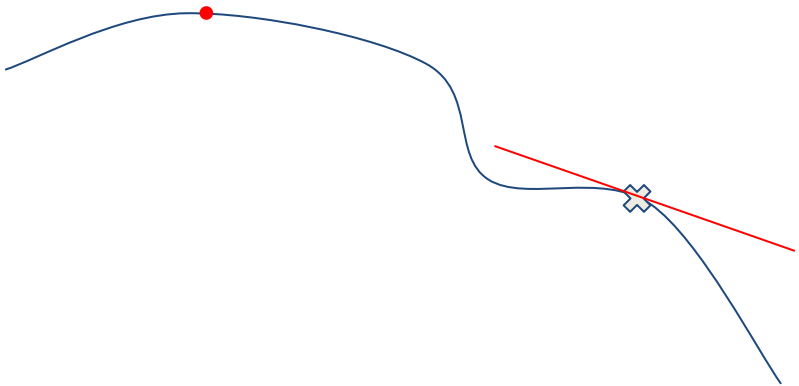
Tetris is a classic benchmark problem for approximate dynamic programming (ADP) methods, stochastic optimization methods are difficult to beat on this task (Gabillon et al., 2013). For continuous control problems, methods like CMA have been successful at learning control policies for challenging tasks like locomotion when provided with hand-engineered policy classes with low-dimensional parameterizations (Wampler & Popović, 2009). The inability of ADP and gradient-based methods to consistently beat gradient-free random search is unsatisfying, since gradient-based optimization algorithms enjoy much better sample complexity guarantees than gradient-free methods (Nemirovski, 2005). Continuous gradient-based optimization has been very successful at learning function approximators for supervised learning tasks with huge numbers of parameters, and extending their success to reinforcement learning would allow for efficient training of complex and powerful policies.

Trust Region Policy Optimization

- Trust Region

$$\max_{\theta} \quad E_{s\sim\rho_{\theta_{\text{old}}},a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta_{\text{old}}}(s,a)\right]$$

$$\text{s.t.} \quad E_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}\left(\pi_{\theta_{\text{old}}}(\cdot|s)\parallel\pi_\theta(\cdot|s)\right)\right]\le\delta$$



Trust Region Policy Optimization

John Schulman JOSCHU@EECS.BERKELEY.EDU
Sergey Levine SLEVINE@EECS.BERKELEY.EDU
Philipp Moritz PCMORITZ@EECS.BERKELEY.EDU
Michael Jordan JORDAN@CS.BERKELEY.EDU
Pieter Abbeel PABBEEL@CS.BERKELEY.EDU
University of California, Berkeley, Department of Electrical Engineering and Computer Sciences

Abstract

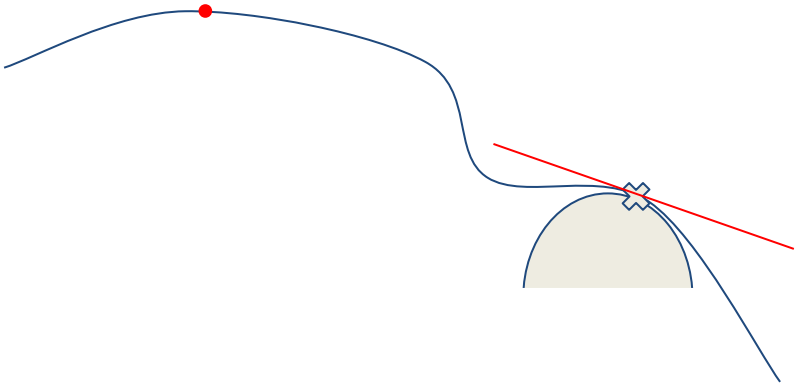
We describe an iterative procedure for optimizing policies, with guaranteed monotonic improvement. By making several approximations to the theoretically-justified procedure, we develop a practical algorithm, called Trust Region Policy Optimization (TRPO). This algorithm is similar to natural policy gradient methods and is effective for optimizing large nonlinear policies such as neural networks. Our experiments demonstrate its robust performance on a wide variety of tasks: learning simulated robotic swimming, hopping, and walking gaits; and playing Atari games using images of the screen as input. Despite its approximations that deviate from the theory, TRPO tends to give monotonic improvement, with little tuning of hyperparameters.

Tetris is a classic benchmark problem for approximate dynamic programming (ADP) methods, stochastic optimization methods are difficult to beat on this task (Gabillon et al., 2013). For continuous control problems, methods like CMA have been successful at learning control policies for challenging tasks like locomotion when provided with hand-engineered policy classes with low-dimensional parameterizations (Wampler & Popović, 2009). The inability of ADP and gradient-based methods to consistently beat gradient-free random search is unsatisfying, since gradient-based optimization algorithms enjoy much better sample complexity guarantees than gradient-free methods (Nemirovski, 2005). Continuous gradient-based optimization has been very successful at learning function approximators for supervised learning tasks with huge numbers of parameters, and extending their success to reinforcement learning would allow for efficient training of complex and powerful policies.

Trust Region Policy Optimization

- Trust Region

$$\max_{\theta} \quad E_{s\sim\rho_{\theta_{\text{old}}},a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta_{\text{old}}}(s,a)\right]$$
$$\text{s.t.} \quad E_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}\left(\pi_{\theta_{\text{old}}}(\cdot|s)\parallel\pi_\theta(\cdot|s)\right)\right]\le\delta$$



Trust Region Policy Optimization

John Schulman JOSCHU@EECS.BERKELEY.EDU
Sergey Levine SLEVINE@EECS.BERKELEY.EDU
Philipp Moritz PCMORITZ@EECS.BERKELEY.EDU
Michael Jordan JORDAN@CS.BERKELEY.EDU
Pieter Abbeel PABBEEL@CS.BERKELEY.EDU
University of California, Berkeley, Department of Electrical Engineering and Computer Sciences

Abstract

We describe an iterative procedure for optimizing policies, with guaranteed monotonic improvement. By making several approximations to the theoretically-justified procedure, we develop a practical algorithm, called Trust Region Policy Optimization (TRPO). This algorithm is similar to natural policy gradient methods and is effective for optimizing large nonlinear policies such as neural networks. Our experiments demonstrate its robust performance on a wide variety of tasks: learning simulated robotic swimming, hopping, and walking gaits; and playing Atari games using images of the screen as input. Despite its approximations that deviate from the theory, TRPO tends to give monotonic improvement, with little tuning of hyperparameters.

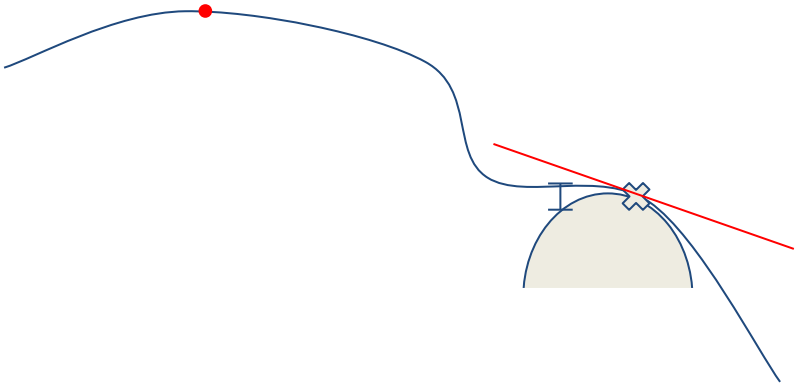
Tetris is a classic benchmark problem for approximate dynamic programming (ADP) methods, stochastic optimization methods are difficult to beat on this task (Gabillon et al., 2013). For continuous control problems, methods like CMA have been successful at learning control policies for challenging tasks like locomotion when provided with hand-engineered policy classes with low-dimensional parameterizations (Wampler & Popović, 2009). The inability of ADP and gradient-based methods to consistently beat gradient-free random search is unsatisfying, since gradient-based optimization algorithms enjoy much better sample complexity guarantees than gradient-free methods (Nemirovski, 2005). Continuous gradient-based optimization has been very successful at learning function approximators for supervised learning tasks with huge numbers of parameters, and extending their success to reinforcement learning would allow for efficient training of complex and powerful policies.

Trust Region Policy Optimization

- Trust Region

$$\max_{\theta} \quad E_{s\sim\rho_{\theta_{\text{old}}},a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta_{\text{old}}}(s,a)\right]$$

$$\text{s.t.} \quad E_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}\left(\pi_{\theta_{\text{old}}}(\cdot|s)\parallel\pi_\theta(\cdot|s)\right)\right]\le\delta$$



Trust Region Policy Optimization

John Schulman JOSCHU@EECS.BERKELEY.EDU
Sergey Levine SLEVINE@EECS.BERKELEY.EDU
Philipp Moritz PCMORITZ@EECS.BERKELEY.EDU
Michael Jordan JORDAN@CS.BERKELEY.EDU
Pieter Abbeel PABBEEL@CS.BERKELEY.EDU
University of California, Berkeley, Department of Electrical Engineering and Computer Sciences

Abstract

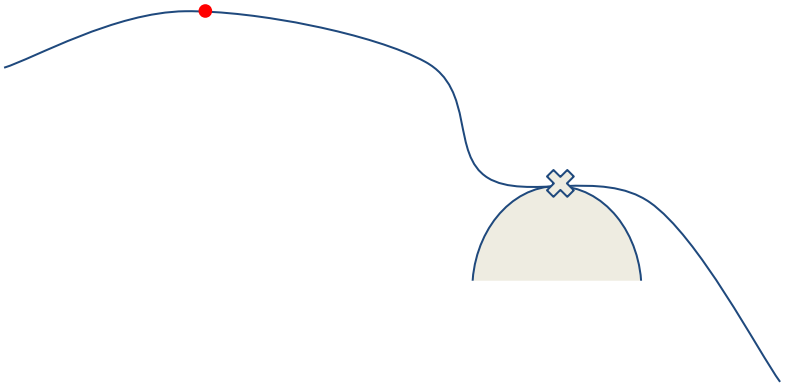
We describe an iterative procedure for optimizing policies, with guaranteed monotonic improvement. By making several approximations to the theoretically-justified procedure, we develop a practical algorithm, called Trust Region Policy Optimization (TRPO). This algorithm is similar to natural policy gradient methods and is effective for optimizing large nonlinear policies such as neural networks. Our experiments demonstrate its robust performance on a wide variety of tasks: learning simulated robotic swimming, hopping, and walking gaits; and playing Atari games using images of the screen as input. Despite its approximations that deviate from the theory, TRPO tends to give monotonic improvement, with little tuning of hyperparameters.

Tetris is a classic benchmark problem for approximate dynamic programming (ADP) methods, stochastic optimization methods are difficult to beat on this task (Gabillon et al., 2013). For continuous control problems, methods like CMA have been successful at learning control policies for challenging tasks like locomotion when provided with hand-engineered policy classes with low-dimensional parameterizations (Wampler & Popović, 2009). The inability of ADP and gradient-based methods to consistently beat gradient-free random search is unsatisfying, since gradient-based optimization algorithms enjoy much better sample complexity guarantees than gradient-free methods (Nemirovski, 2005). Continuous gradient-based optimization has been very successful at learning function approximators for supervised learning tasks with huge numbers of parameters, and extending their success to reinforcement learning would allow for efficient training of complex and powerful policies.

Trust Region Policy Optimization

- Trust Region

$$\max_{\theta} \quad E_{s\sim\rho_{\theta_{\text{old}}},a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta_{\text{old}}}(s,a)\right]$$
$$\text{s.t.} \quad E_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}\left(\pi_{\theta_{\text{old}}}(\cdot|s)\parallel\pi_\theta(\cdot|s)\right)\right]\le\delta$$



Trust Region Policy Optimization

John Schulman JOSCHU@EECS.BERKELEY.EDU
Sergey Levine SLEVINE@EECS.BERKELEY.EDU
Philipp Moritz PCMORITZ@EECS.BERKELEY.EDU
Michael Jordan JORDAN@CS.BERKELEY.EDU
Pieter Abbeel PABBEEL@CS.BERKELEY.EDU
University of California, Berkeley, Department of Electrical Engineering and Computer Sciences

Abstract

We describe an iterative procedure for optimizing policies, with guaranteed monotonic improvement. By making several approximations to the theoretically-justified procedure, we develop a practical algorithm, called Trust Region Policy Optimization (TRPO). This algorithm is similar to natural policy gradient methods and is effective for optimizing large nonlinear policies such as neural networks. Our experiments demonstrate its robust performance on a wide variety of tasks: learning simulated robotic swimming, hopping, and walking gaits; and playing Atari games using images of the screen as input. Despite its approximations that deviate from the theory, TRPO tends to give monotonic improvement, with little tuning of hyperparameters.

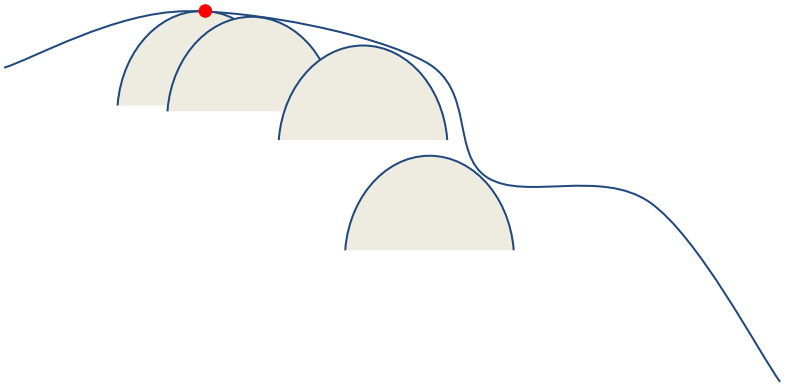
Tetris is a classic benchmark problem for approximate dynamic programming (ADP) methods, stochastic optimization methods are difficult to beat on this task (Gabillon et al., 2013). For continuous control problems, methods like CMA have been successful at learning control policies for challenging tasks like locomotion when provided with hand-engineered policy classes with low-dimensional parameterizations (Wampler & Popović, 2009). The inability of ADP and gradient-based methods to consistently beat gradient-free random search is unsatisfying, since gradient-based optimization algorithms enjoy much better sample complexity guarantees than gradient-free methods (Nemirovski, 2005). Continuous gradient-based optimization has been very successful at learning function approximators for supervised learning tasks with huge numbers of parameters, and extending their success to reinforcement learning would allow for efficient training of complex and powerful policies.

Trust Region Policy Optimization

- Trust Region

$$\max_{\theta} \quad E_{s\sim\rho_{\theta_{\text{old}}},a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta_{\text{old}}}(s,a)\right]$$

$$\text{s.t.} \quad E_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}\left(\pi_{\theta_{\text{old}}}(\cdot|s)\parallel\pi_\theta(\cdot|s)\right)\right]\le\delta$$



Trust Region Policy Optimization

John Schulman JOSCHU@EECS.BERKELEY.EDU
Sergey Levine SLEVINE@EECS.BERKELEY.EDU
Philipp Moritz PCMORITZ@EECS.BERKELEY.EDU
Michael Jordan JORDAN@CS.BERKELEY.EDU
Pieter Abbeel PABBEEL@CS.BERKELEY.EDU
University of California, Berkeley, Department of Electrical Engineering and Computer Sciences

Abstract

We describe an iterative procedure for optimizing policies, with guaranteed monotonic improvement. By making several approximations to the theoretically-justified procedure, we develop a practical algorithm, called Trust Region Policy Optimization (TRPO). This algorithm is similar to natural policy gradient methods and is effective for optimizing large nonlinear policies such as neural networks. Our experiments demonstrate its robust performance on a wide variety of tasks: learning simulated robotic swimming, hopping, and walking gaits; and playing Atari games using images of the screen as input. Despite its approximations that deviate from the theory, TRPO tends to give monotonic improvement, with little tuning of hyperparameters.

Tetris is a classic benchmark problem for approximate dynamic programming (ADP) methods, stochastic optimization methods are difficult to beat on this task (Gabillon et al., 2013). For continuous control problems, methods like CMA have been successful at learning control policies for challenging tasks like locomotion when provided with hand-engineered policy classes with low-dimensional parameterizations (Wampler & Popović, 2009). The inability of ADP and gradient-based methods to consistently beat gradient-free random search is unsatisfying, since gradient-based optimization algorithms enjoy much better sample complexity guarantees than gradient-free methods (Nemirovski, 2005). Continuous gradient-based optimization has been very successful at learning function approximators for supervised learning tasks with huge numbers of parameters, and extending their success to reinforcement learning would allow for efficient training of complex and powerful policies.

Proximal Policy Optimization

- Trust Region

$$\max_{\theta} \quad E_{s\sim\rho_{\theta_{\text{old}}},a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta_{\text{old}}}(s,a)\right]$$
$$\text{s. t.} \quad E_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}\left(\pi_{\theta_{\text{old}}}(\cdot|s)\parallel\pi_\theta(\cdot|s)\right)\right]\leq\delta$$

Proximal Policy Optimization

- Trust Region

$$\max_{\theta} \quad E_{s\sim\rho_{\theta_{\text{old}}},a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta_{\text{old}}}(s,a)\right]$$

$$\text{s. t.} \quad E_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}\left(\pi_{\theta_{\text{old}}}(\cdot|s)\parallel\pi_\theta(\cdot|s)\right)\right]\leq\delta$$

Proximal Policy Optimization Algorithms

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

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.

Proximal Policy Optimization

- Trust Region

$$\max_{\theta} \quad E_{s\sim\rho_{\theta_{\text{old}}},a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta_{\text{old}}}(s,a)\right]$$

$$\text{s. t.} \quad E_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}\left(\pi_{\theta_{\text{old}}}(\cdot|s)\parallel\pi_\theta(\cdot|s)\right)\right]\leq\delta$$

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

Proximal Policy Optimization Algorithms

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

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.

Proximal Policy Optimization

- Trust Region

$$\max_{\theta} \quad E_{s\sim\rho_{\theta_{\text{old}}},a\sim q}\left[\frac{\pi_\theta(a|s)}{q(a|s)}Q_{\theta_{\text{old}}}(s,a)\right]$$

$$\text{s. t.} \quad E_{s\sim\rho_{\theta_{\text{old}}}}\left[D_{\text{KL}}\left(\pi_{\theta_{\text{old}}}(\cdot|s)\parallel\pi_\theta(\cdot|s)\right)\right]\leq\delta$$

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

Proximal Policy Optimization Algorithms

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

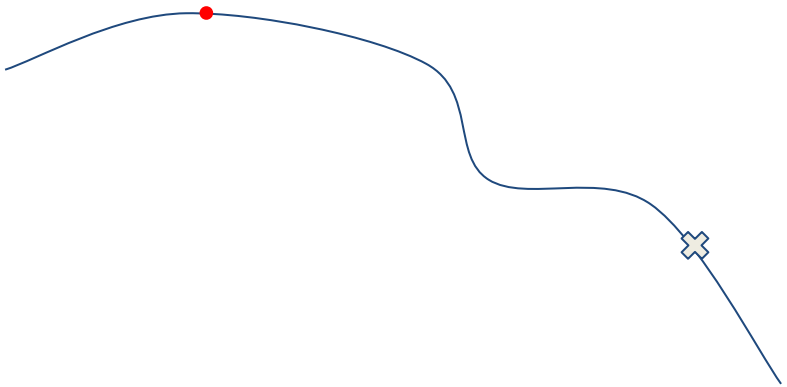
Abstract

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.

Proximal Policy Optimization

- Trust Region

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



Proximal Policy Optimization Algorithms

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

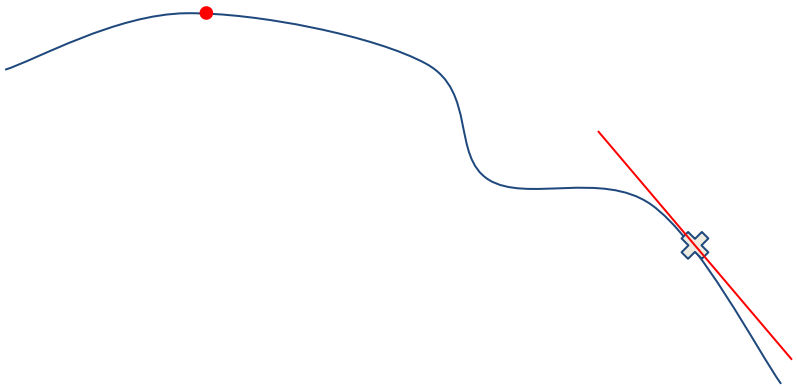
Abstract

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.

Proximal Policy Optimization

- Trust Region

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



Proximal Policy Optimization Algorithms

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

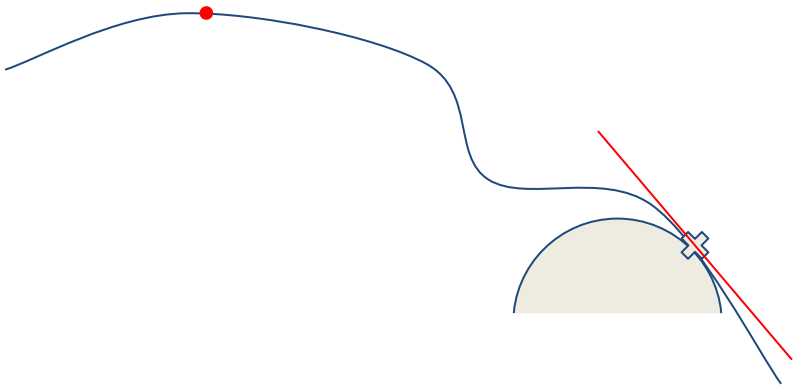
Abstract

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.

Proximal Policy Optimization

- Trust Region

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



Proximal Policy Optimization Algorithms

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

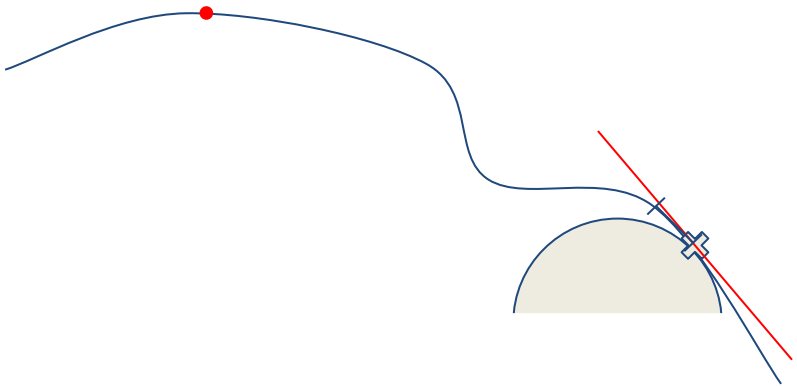
Abstract

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.

Proximal Policy Optimization

- Trust Region

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



Proximal Policy Optimization Algorithms

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

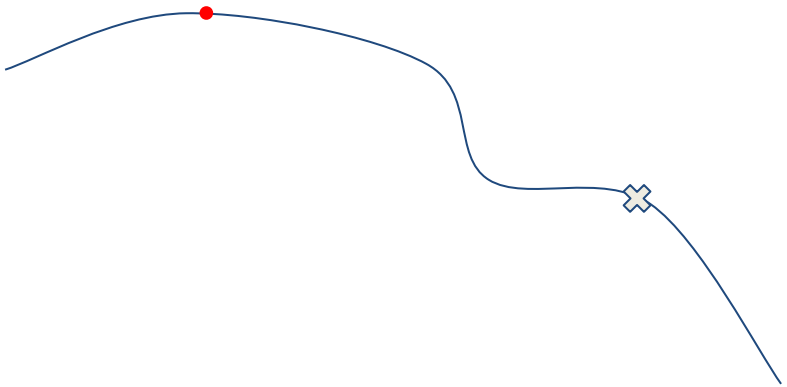
Abstract

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.

Proximal Policy Optimization

- Trust Region

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



Proximal Policy Optimization Algorithms

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

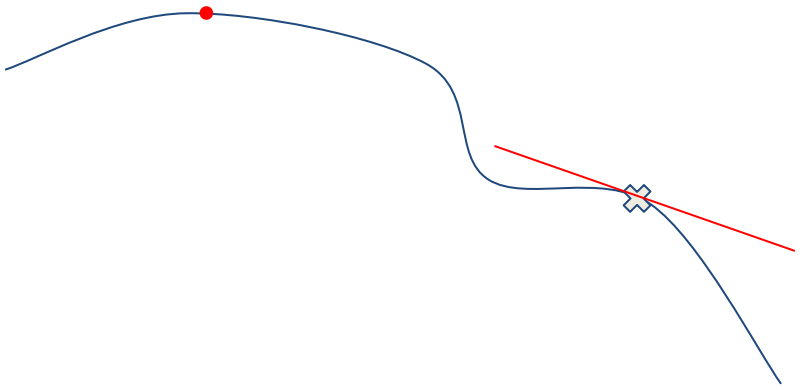
Abstract

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.

Proximal Policy Optimization

- Trust Region

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



Proximal Policy Optimization Algorithms

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

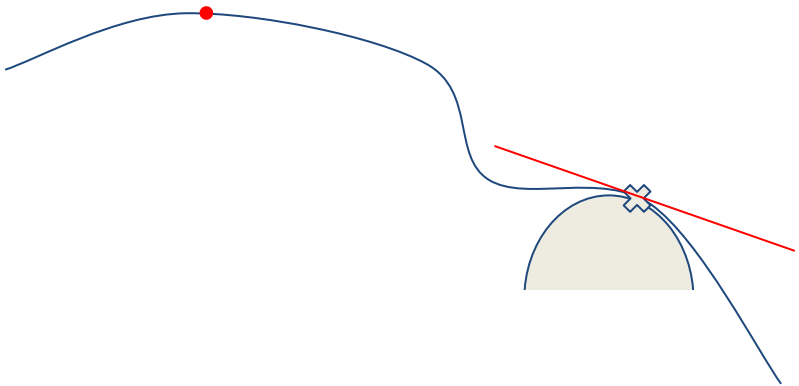
Abstract

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.

Proximal Policy Optimization

- Trust Region

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



Proximal Policy Optimization Algorithms

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

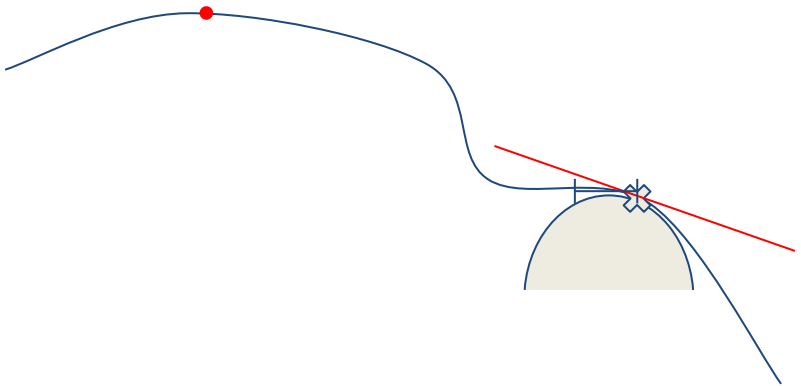
Abstract

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.

Proximal Policy Optimization

- Trust Region

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



Proximal Policy Optimization Algorithms

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

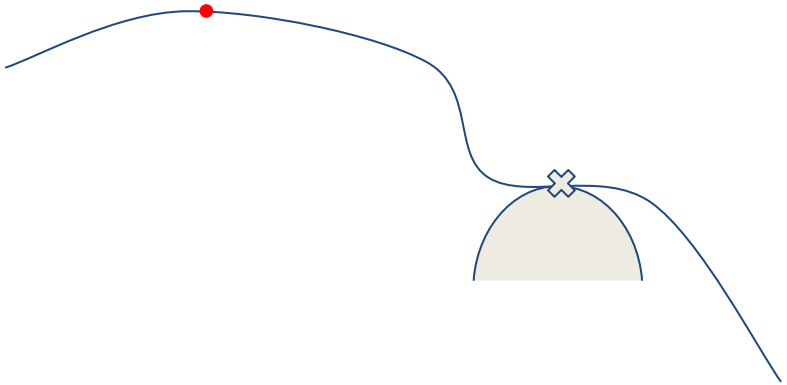
Abstract

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.

Proximal Policy Optimization

- Trust Region

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



Proximal Policy Optimization Algorithms

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

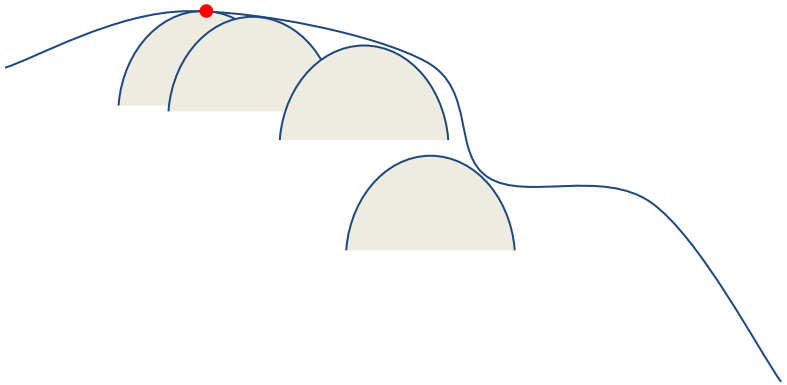
Abstract

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.

Proximal Policy Optimization

- Trust Region

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



Proximal Policy Optimization Algorithms

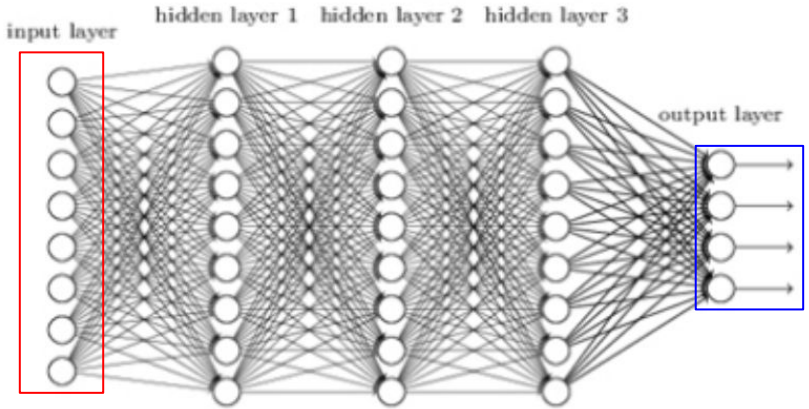
John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov
OpenAI
{joschu, filip, prafulla, alec, oleg}@openai.com

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.

Value base Reinforcement Learning

Value base Reinforcement Learning



Value base Reinforcement Learning

- Playing Atari with Deep Reinforcement Learning(NIPS 2013)
- Human-level control through deep reinforcement learning(Nature 2015)
- Deep Reinforcement Learning with Double Q-Learning
- Dueling Network Architectures for Deep Reinforcement Learning
- Prioritized Experience Replay
- .....

Value base Reinforcement Learning

Distance : 2 min

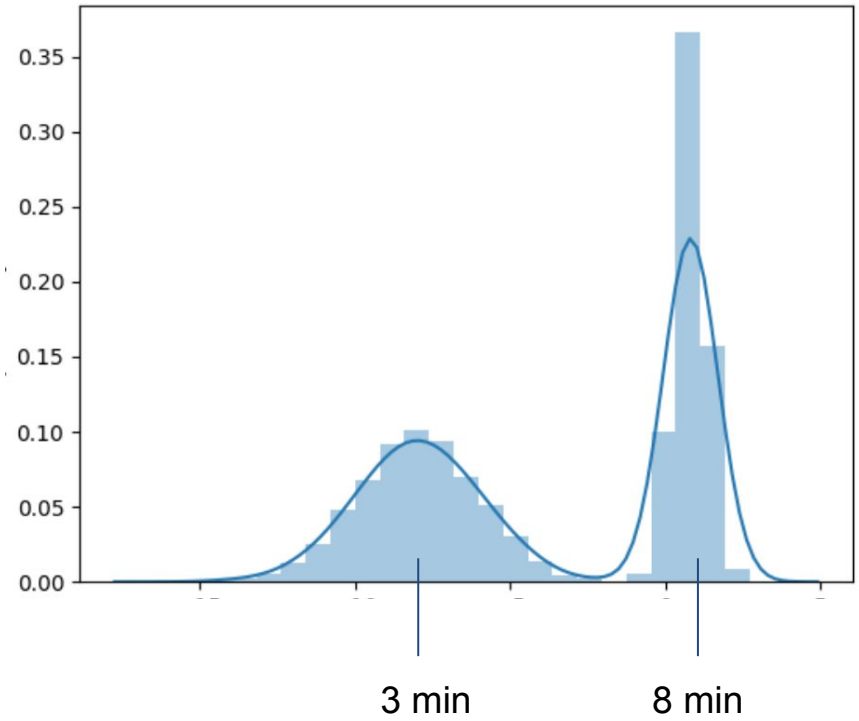
Dispatch Time : 5 min

Value base Reinforcement Learning

Distan 2 min

Dispatch Time : 5 min

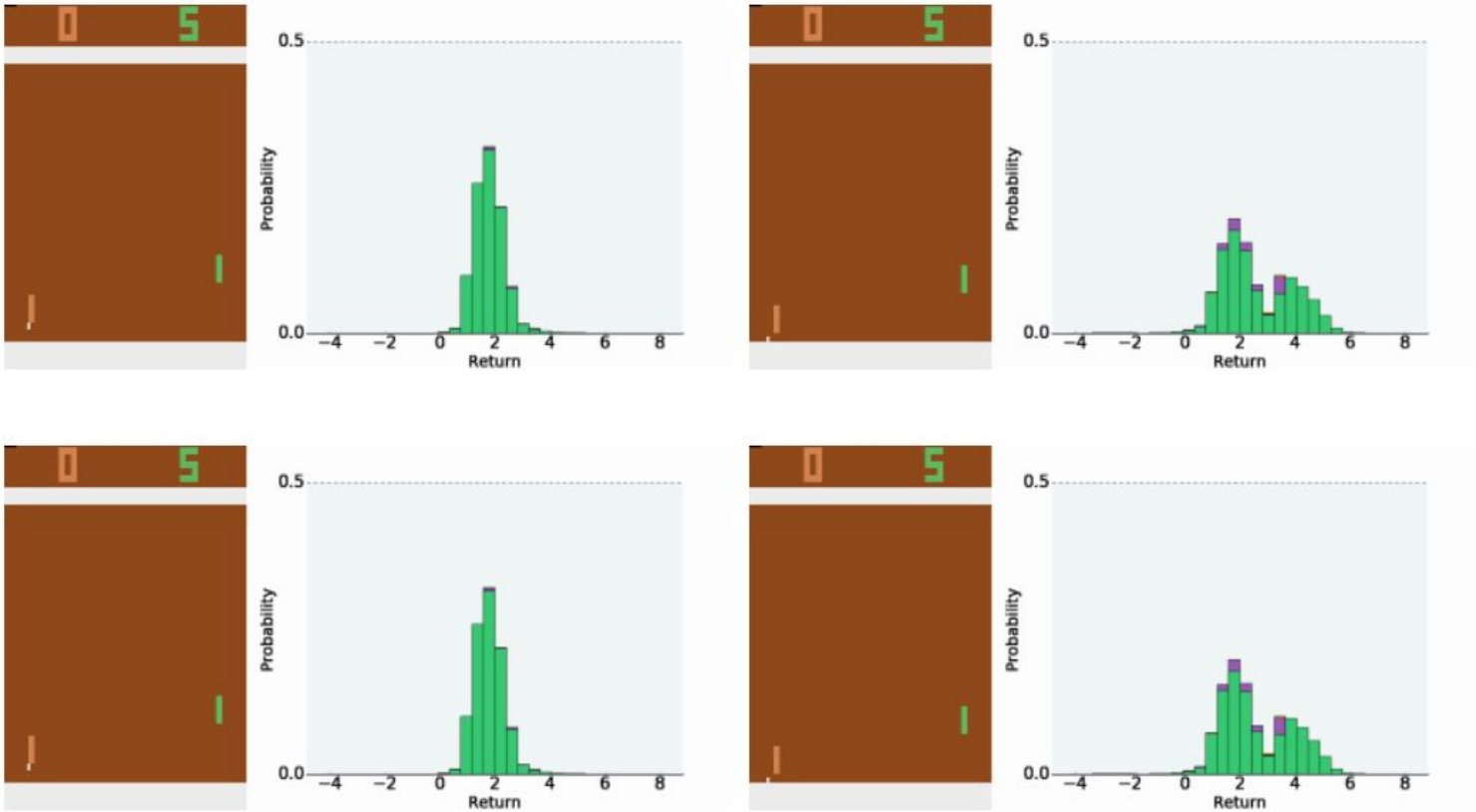
Value base Reinforcement Learning



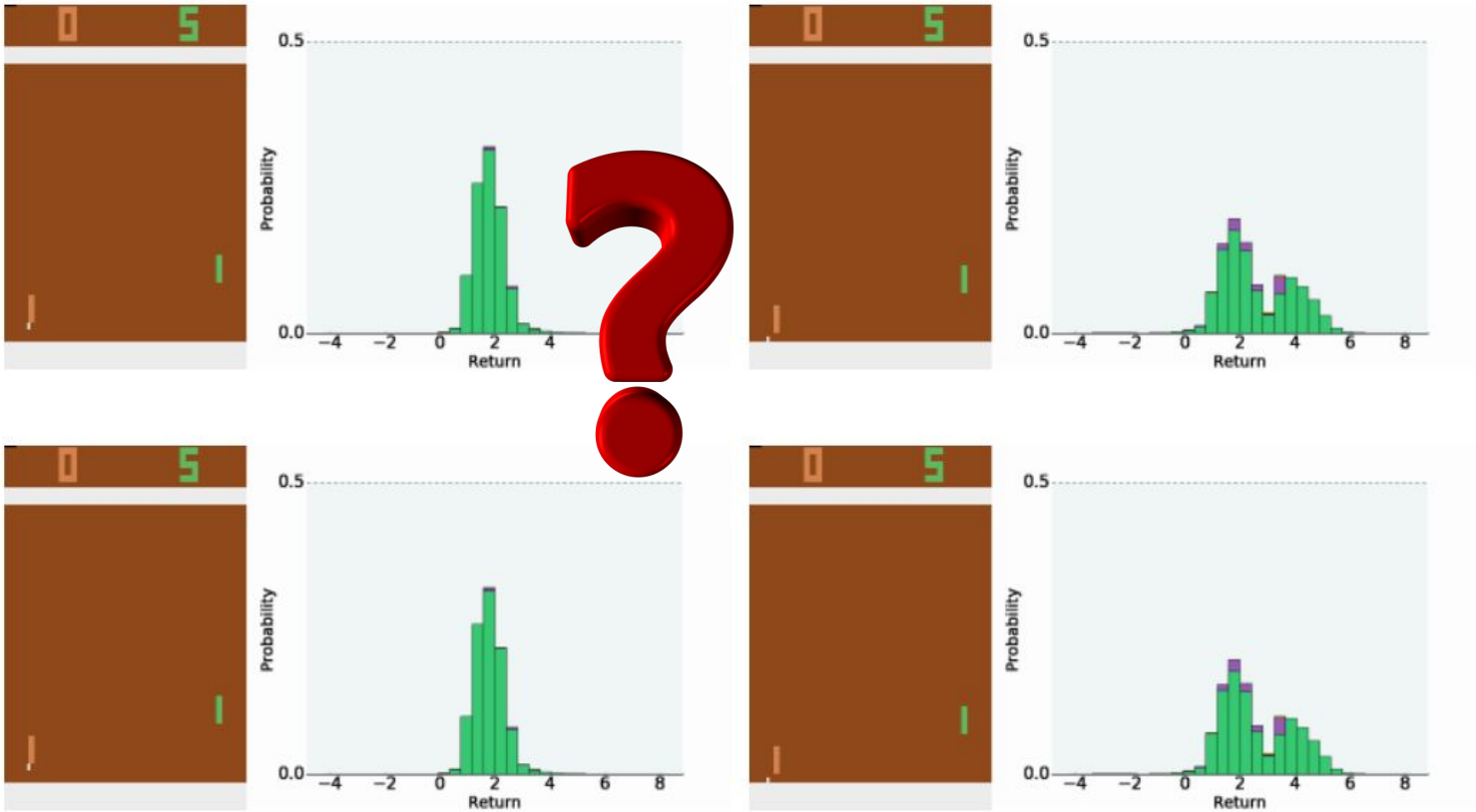
Value base Reinforcement Learning

- A Distributional perspective on Reinforcement Learning(2017)
- Distributional Reinforcement Learning with Quantile Regression(2017)
- Implicit Quantile Networks for Distributional Reinforcement Learning(2018)

Value base Reinforcement Learning



Value base Reinforcement Learning



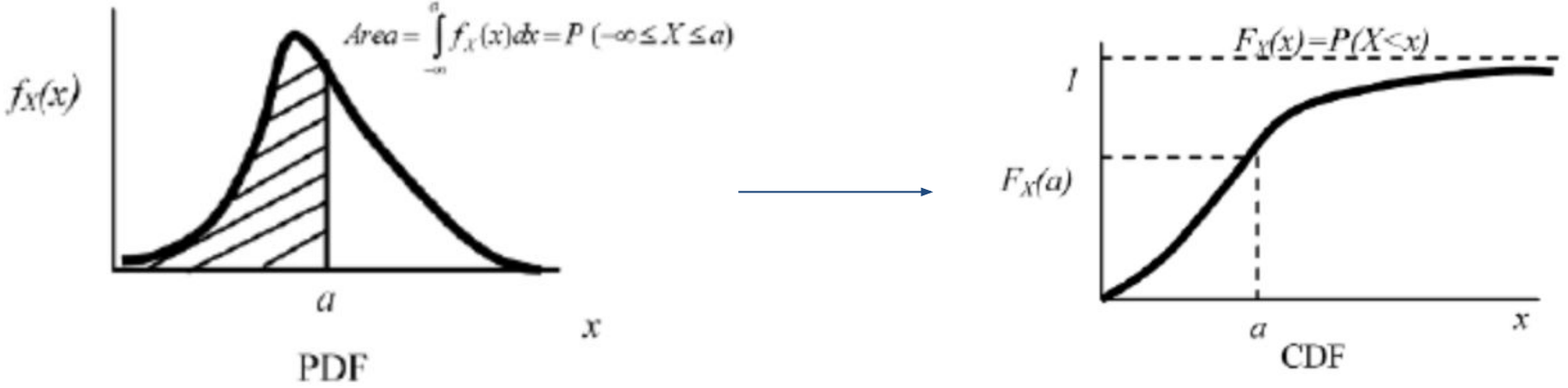
Distributional Reinforcement Learning with Quantile Regrssion

- Histogram cannot meet condition

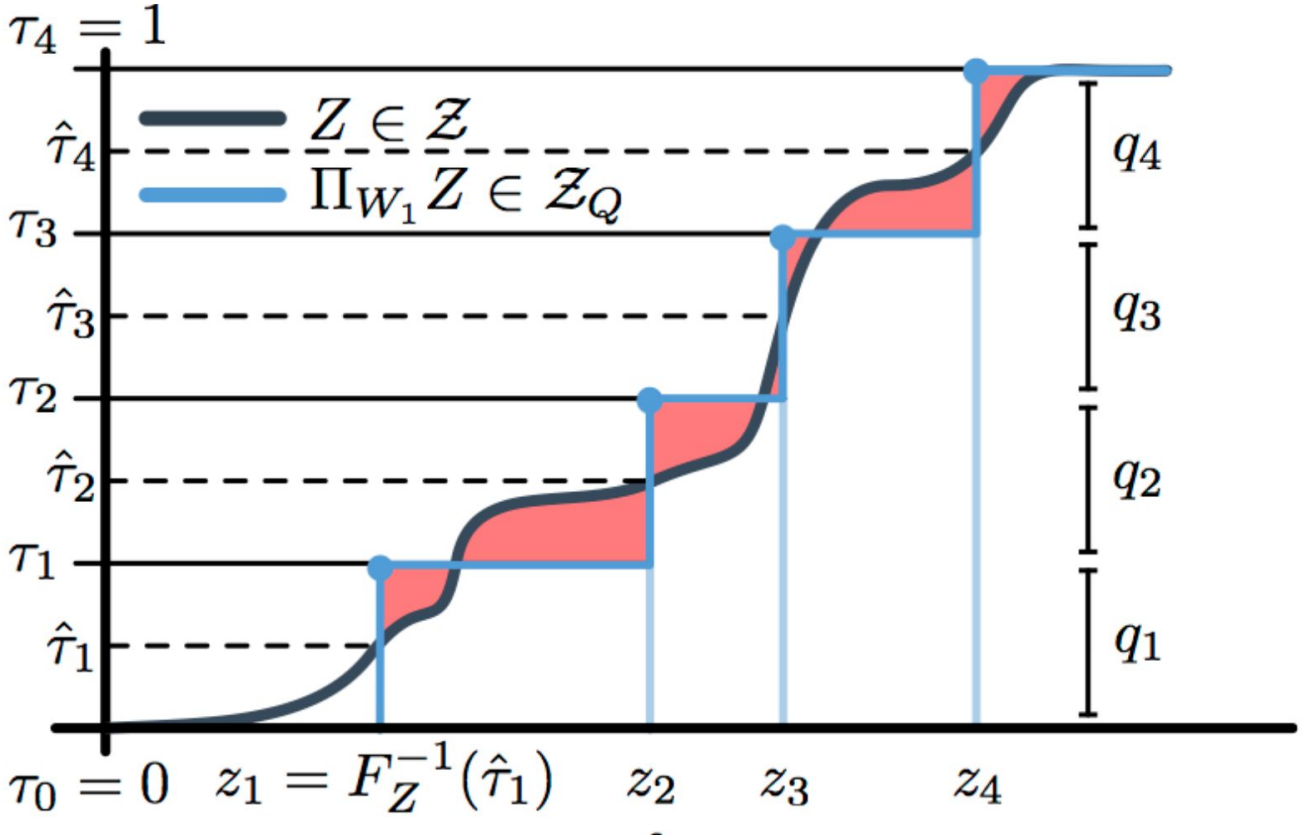
$$\bar{d}_p(\mathcal{T}^\pi Z_1, \mathcal{T}^\pi Z_2) \leq \gamma \bar{d}_p(Z_1, Z_2).$$

- Wasserstein Distance

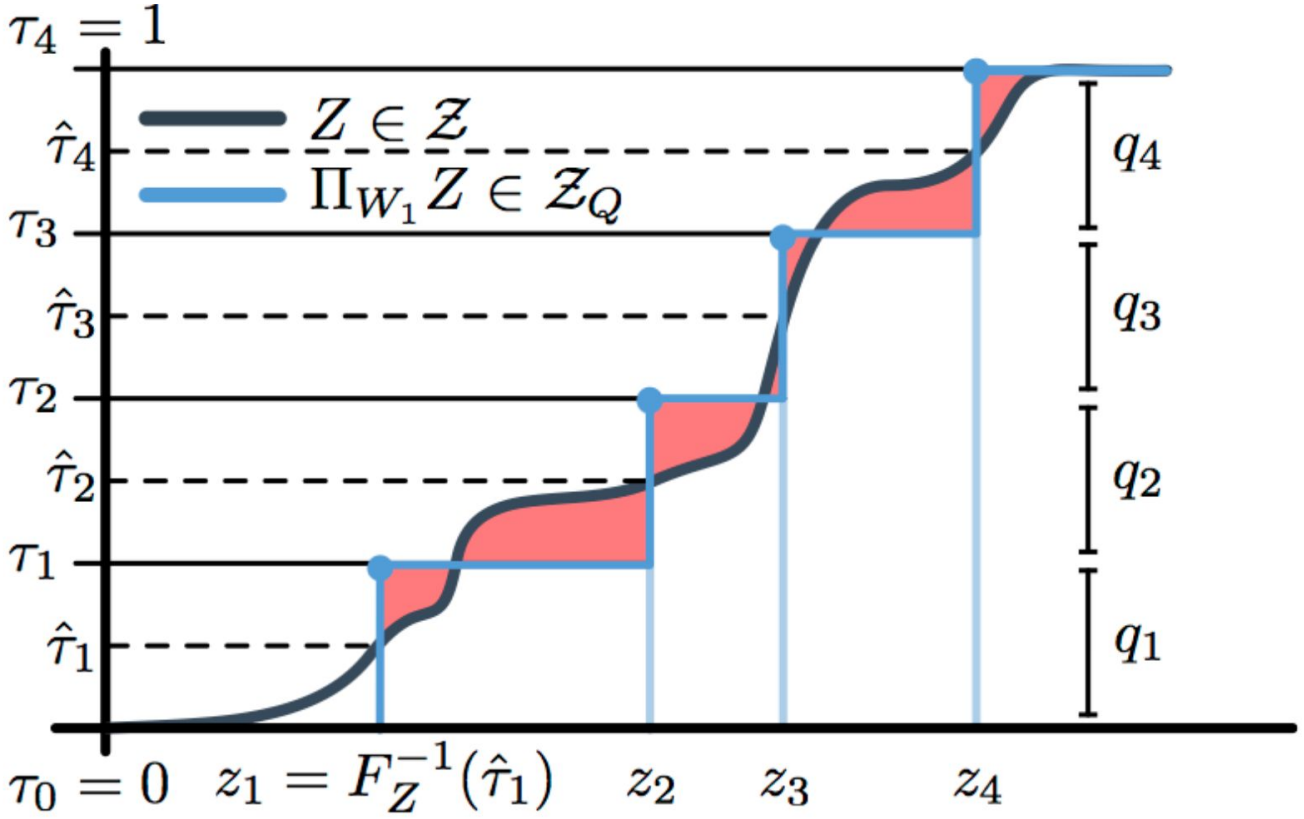
$$W_p(U, Y) = \left(\int_0^1 |F_Y^{-1}(\omega) - F_U^{-1}(\omega)|^p d\omega \right)^{1/p}$$



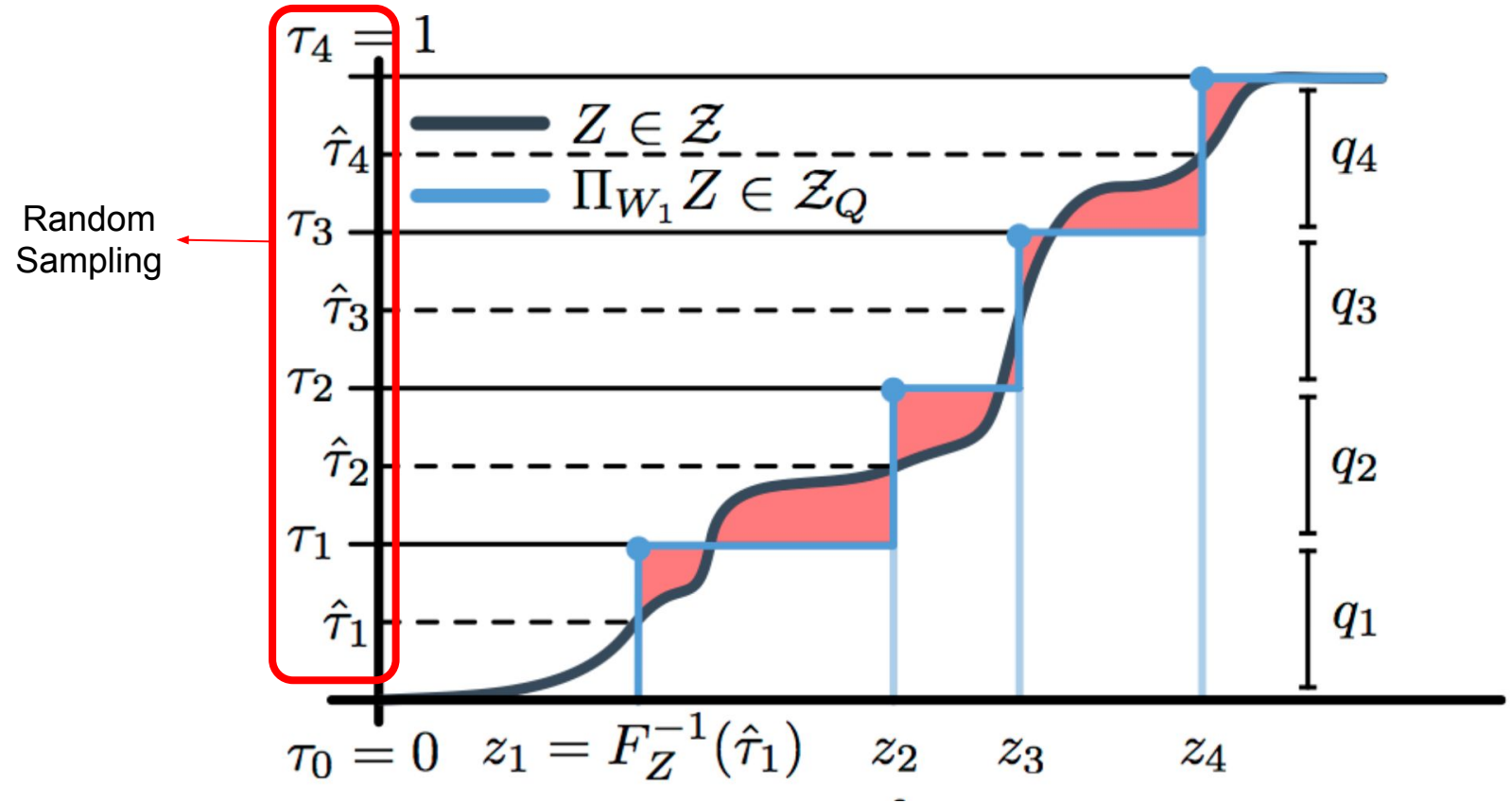
Distributional Reinforcement Learning with Quantile Regrssion



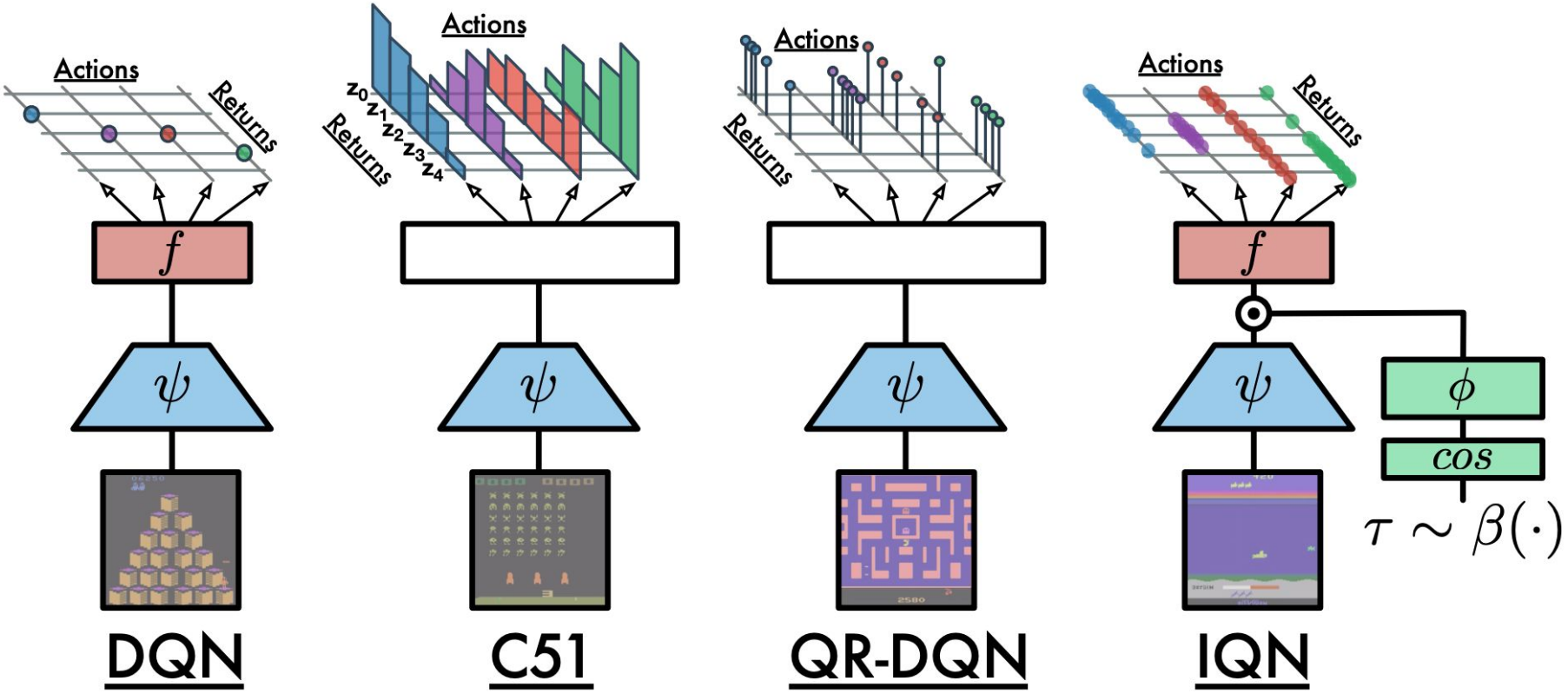
Implicit Quantile Network for Distributional Reinforcement Learning



Implicit Quantile Network for Distributional Reinforcement Learning



Value base Reinforcement Learning



Famous open source for Reinforcement Learning

Famous open source for Reinforcement Learning

- Deepminds

Famous open source for Reinforcement Learning

- Deepminds
- Open AI

Famous open source for Reinforcement Learning

- Deepminds
- Open AI

Famous open source for Reinforcement Learning

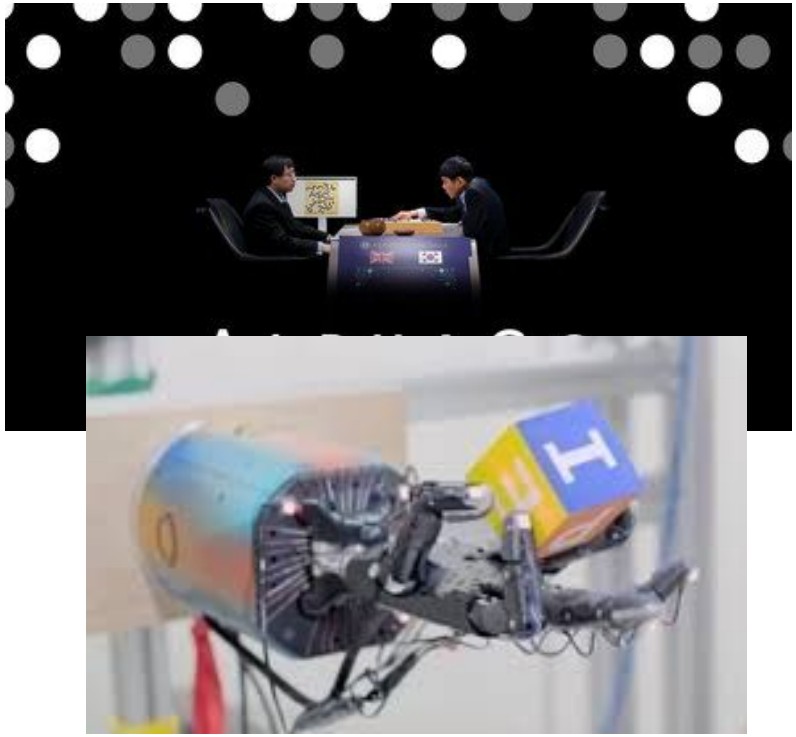
- Deepminds
- Open AI

Famous open source for Reinforcement Learning

- Deepminds
- Open AI

Famous open source for Reinforcement Learning

- Deepminds
- Open AI



Famous open source for Reinforcement Learning

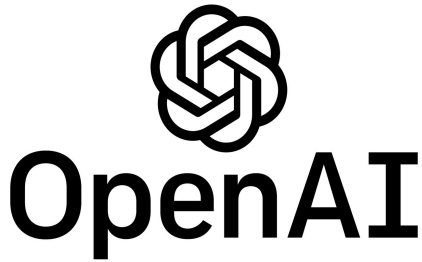
- Deepminds
- Open AI

Famous open source for Reinforcement Learning

- Deepminds
  - Dopamine
    - Value based Reinforcement Learning opensource baseline
    - https://github.com/google/dopamine
- Open AI

Famous open source for Reinforcement Learning

- Deepminds
  - Dopamine
    - Value based Reinforcement Learning open source baseline
    - https://github.com/google/dopamine
- Open AI
  - Baselines
    - Policy based Reinforcement Learning open source baseline
    - https://github.com/openai/baselines



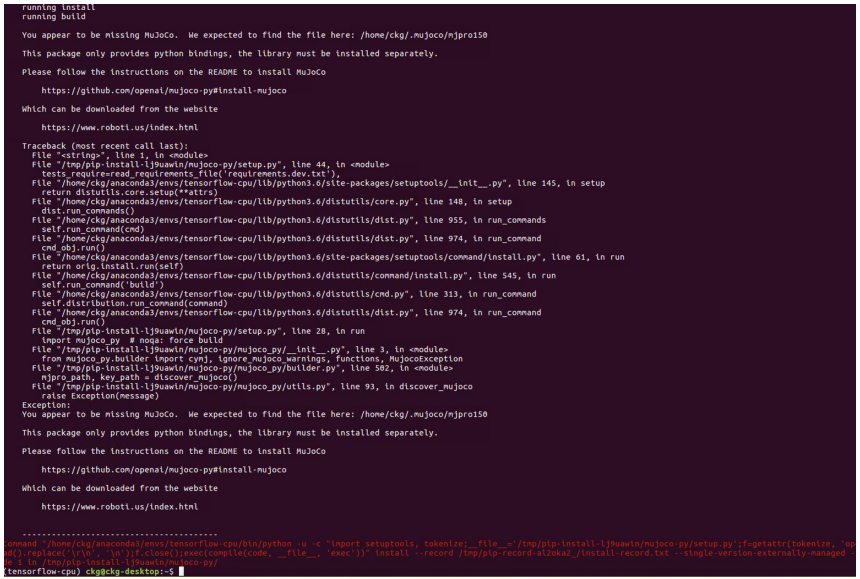
Why?

Why?

- Hard to Install

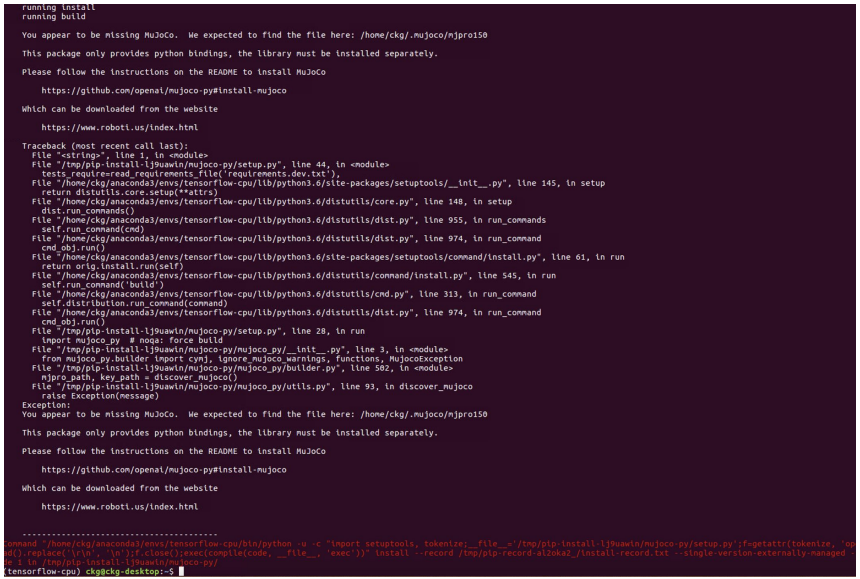
Why?

- Hard to Install
  - baselines : mp4py, mujoco…
  - dopamine : easy



Why?

- Hard to Install
  - baselines : mp4py, mujoco…
  - dopamine : easy



Why?

- Hard to Install
  - baselines : mp4py, mujoco…
  - dopamine : easy
- Hard to use to your environment

Why?

- Hard to Install
  - baselines : mp4py, mujoco…
  - dopamine : easy
- Hard to use to your environment
  - manipulate the api
  - https://github.com/google/dopamine
  - https://github.com/openai/baselines

Why?

- Hard to Install
  - baselines : mp4py, mujoco…
  - dopamine : easy
- Hard to use to your environment
  - manipulate the api
  - https://github.com/google/dopamine
  - https://github.com/openai/baselines
- Korean

Requirements

Requirements

- Hard to Install

Requirements

- Easy to Install

Requirements

- Easy to Install

Installation

cpu version

```
pip install tensorflow-rl[tf-cpu]
```

gpu version

```
pip install tensorflow-rl[tf-gpu]
```

Requirements

- Easy to Install
- Hard to use to your environment

Requirements

- Easy to Install
- Easy to use to your environment

Requirements

- Easy to Install
- Easy to use to your environment
  - Provide many tutorial code(discrete, continuous action)

Requirements

- Easy to Install
- Easy to use to your environment
  - Provide many tutorial code(discrete, continuous action)
  - Keep the flow of other reinforcement learning code

Requirements

- Easy to Install
- Easy to use to your environment
  - Provide many tutorial code(discrete, continuous action)
  - Keep the flow of other reinforcement learning code

```
env = gym.make('Breakout-v2')

for i in range(episode):
    state = env.reset()
    done = False
    while not done:
        action = inference(state)
        next_state, reward, done, _ = env.step(action)
        state = next_state
    train_model(data)
```

Requirements

- Easy to Install
- Easy to use to your environment
- Korean

Supported algorithms

Supported algorithms

- Vanilla Policy Gradient
- Advantage Actor Critic
- Proximal Policy Optimization
- Deep Deterministic Policy Gradient

Supported algorithms

- Vanilla Policy Gradient(Discrete Action)
- Advantage Actor Critic(Discrete Action)
- Proximal Policy Optimization(Discrete, Continuous Action)
- Deep Deterministic Policy Gradient(Discrete, Continuous Action)

To do List

- Vanilla Policy Gradient(Discrete Action)
- Advantage Actor Critic(Discrete Action)
- Proximal Policy Optimization(Discrete, Continuous Action)
- Deep Deterministic Policy Gradient(Discrete, Continuous Action)

To do List

- Vanilla Policy Gradient(Discrete Action)
- Advantage Actor Critic(Discrete Action)
- Proximal Policy Optimization(Discrete, Continuous Action)
- Deep Deterministic Policy Gradient(Discrete, Continuous Action)

- Applicate LSTM to Vanilla Policy Gradient, Advantage Actor Critic, Proximal Policy Optimization
- Actor Critic Experience Replay
- Soft actor critic
- Value based Reinforcement Learning
    - DQN, Double DQN, Deuling DQN to Rainbow
    - Distributional RL

How to use

How to use

```
from agent.utils import get_gaes
from agent.discrete.seperate.ppo import PPO
import gym
import tensorflow as tf
import numpy as np
from model import MLPActor, MLPCritic

env = gym.make('CartPole-v1')
state_size, output_size = 4, 2
sess = tf.Session()
actor = MLPActor('actor', state_size, output_size)
critic = MLPCritic('critic', state_size)
agent = PPO(sess, output_size, None, None, actor, critic)
sess.run(tf.global_variables_initializer())

last_score = 0

while True:
    total_state, total_next_state, total_reward, total_done, total_action = [], [], [], [], []
    state = env.reset()
    done = False
    score = 0
    while not done:
        if last_score > 300:
            env.render()
        action = agent.get_action([state])
        action = action[0]

        next_state, step, done, _ = env.step(action)
        score += step
        reward = 0

        if done:
            if score == 500:
                reward = 1
            else:
                reward = -1


        total_state.append(state)
        total_next_state.append(next_state)
        total_done.append(done)
        total_action.append(action)
        total_reward.append(reward)

        state = next_state
```

```
total_state = np.stack(total_state)
total_next_state = np.stack(total_next_state)
total_action = np.stack(total_action)
total_reward = np.stack(total_reward)
total_done = np.stack(total_done)

value, next_value = agent.get_value(total_state, total_next_state)
adv, target = get_gaes(total_reward, total_done, value, next_value, agent.gamma, agent.lamda, True)

agent.train_model(total_state, total_action, target, adv)
print(score)
last_score = score
```

License

License

Free

We are hiring

Support us in any form

https://github.com/RLOpensource/tensorflow_RL/blob/master/model.py