

Reinforcement Learning

Exercise 8: Actor-Critic Methods

Nico Meyer

Overview

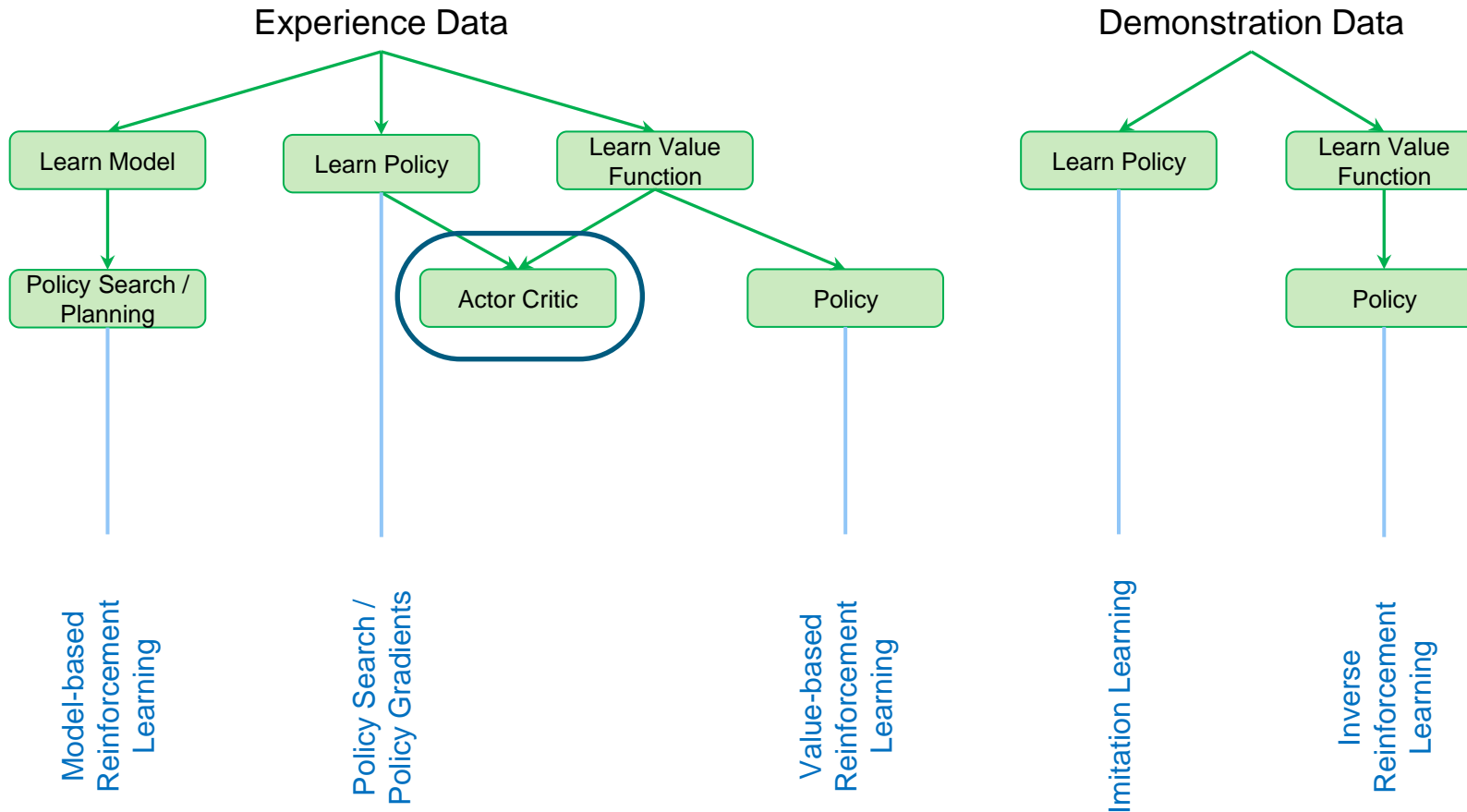
Exercise Content

<i>Week</i>	<i>Date</i>	<i>Topic</i>	<i>Material</i>	<i>Who?</i>
1	22.04.		<i>no exercises</i>	
2	29.04.	MDPs (slides)	ex1.pdf	Nico
3	06.05.	T.B.D.		
4	13.05.	Dynamic Programming (slides)	ex2.pdf, ex2_skeleton.zip	Alex
5	20.05.	OpenAI Gym, PyTorch-Intro (slides) TD-Learning (slides)		Nico
6	27.05.	TD-Control (slides)		Nico
7	03.06.	Intermediate exam		
8	10.06.		<i>no exercises</i>	
9	17.06.	DQN (slides)		Nico
10	24.06.	VPG (slides)		Alex
11	01.07.	A2C (slides)		Nico
12	08.07.	Multi-armed Bandits (slides)		Alex
13	15.07.	RND/ICM (slides)		Alex
14	22.07.	MCTS (slides)		Alex



Overview

General Picture



Actor-Critic Methods

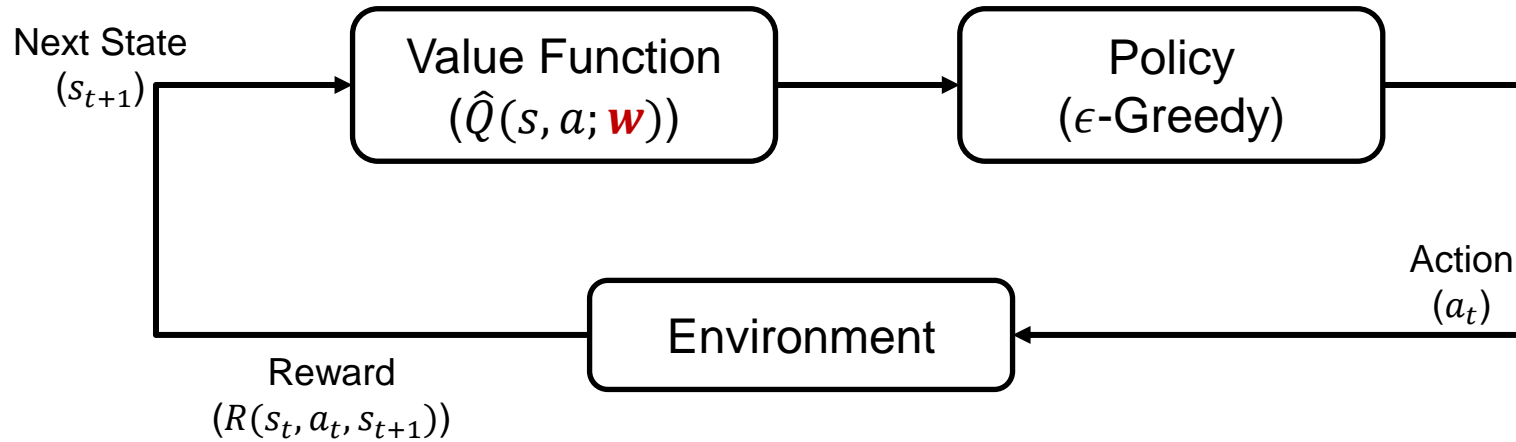
Advantage Actor Critic



Brief Recap

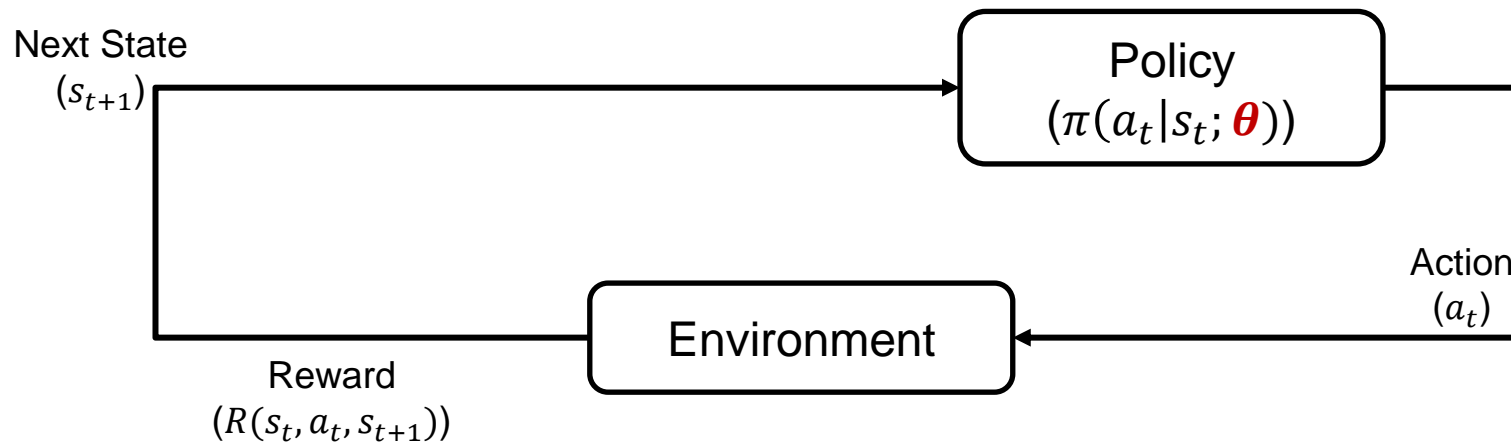
Policy-based Reinforcement Learning

Value-based:



Goal: find w that approximates the true Q -function

Policy-based:



Goal: find θ that maximizes long term reward

Recap

Policy Gradients

- Our goal is to maximize the expected reward:

$$G(\tau) := \sum_{t=0}^{T-1} \gamma^t R(s_t, a_t)$$
$$\max_{\theta} \mathbb{E}_{\pi_{\theta}} G(\tau)$$

(where π_{θ} is a parameterized policy, e.g., a neural network)

- But how do we maximize this?

→ **Gradient Ascent!** Suppose we know how to calculate the gradient w.r.t. the parameters:

$$\nabla_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} G(\tau)$$

- Then we can update our parameters θ in the direction of the gradient:

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} G(\tau)$$

Policy Gradient

often in literature
referred to as $\nabla_{\theta} J(\pi_{\theta})$

Policy-based Reinforcement Learning

Baselines and Advantage Functions

- We can subtract **baseline functions** from our policy gradient without changing its expectation:

$$\nabla_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} G(\tau) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left(\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) (G_t - b(s_t)) \right) = \sum_{t'=t}^T \gamma^{t'-t} R(s_{t'}, a_{t'}) = Q^{\pi}(s_t, a_t)$$

- We can define *advantage functions* that reduce variance
- For instance, we can use $b(s_t) = v^{\pi}(s_t)$ to reduce variance in the sample estimate of the policy gradient:

$$A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$$

→ faster and more stable policy learning!

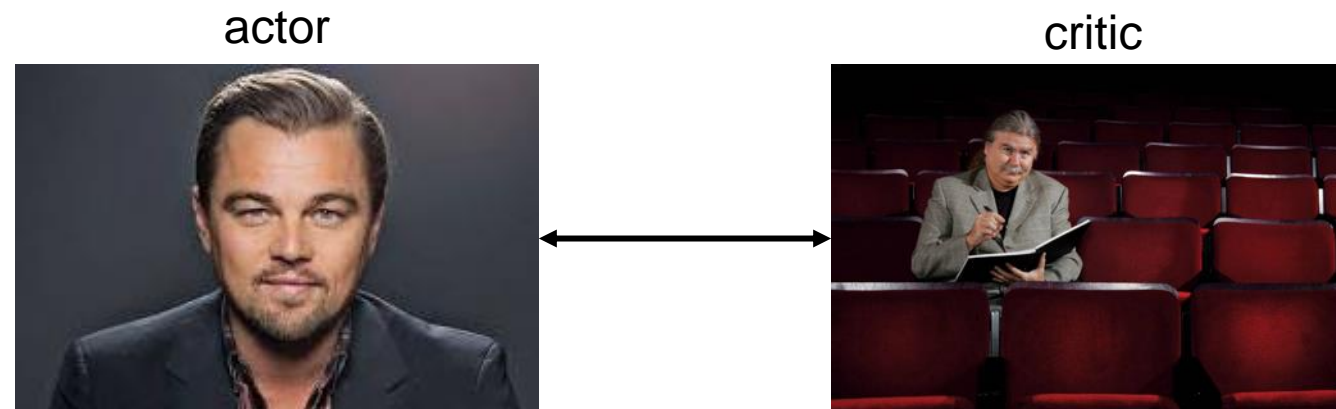
* You can find one version of the proof here: https://spinningup.openai.com/en/latest/spinningup/rl_intro3.html#id1

Actor-Critic Methods

Reducing Variance

$$\nabla_{\theta} \mathbb{E}_{\pi_{\theta}} G(\tau) \approx \frac{1}{L} \sum_{\tau} \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \underbrace{\sum_{t'=t}^T \gamma^{t'-t} R(s_{t'}, a_{t'}) - b(s_t)}_{= Q^{\pi}(s_t, a_t)}$$

- Monte-Carlo policy gradient is sampled and has high variance
- Idea: we can use a critic that estimates the Q



Actor-Critic Methods

Introduce critic that estimates Q

- The policy gradient we used so far (without baseline to begin with):

$$\begin{aligned}\nabla_{\theta} \mathbb{E}_{\pi_{\theta}} G(\tau) &\approx \frac{1}{L} \sum_{\tau} \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) G(\tau) \\ &\approx \frac{1}{L} \sum_{\tau} \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \underbrace{\sum_{t'=t}^T \gamma^{t'-t} R(s_{t'}, a_{t'})}_{= Q^{\pi}(s_t, a_t)}\end{aligned}$$

MC samples this, i.e. high variance
-> Use Critic to estimate Q

- In practice, (as we already know) $Q^{\pi}(s_t, a_t)$ cannot be “computed”
- we instead need to approximate it with a neural network with parameters ϕ and standard SGD over k epochs minimizing the MSE:

$$\phi_k = \arg \min_{\phi} \mathbb{E}_{s_t, a_t, \hat{R}_t \sim \pi_k} \left[(Q_{\phi}(s_t, a_t) - \hat{R}_t)^2 \right]$$

Actor-Critic Methods

Introduce critic that estimates Q

- The policy gradient we used so far (without baseline to begin with):

$$\begin{aligned}\nabla_{\theta} \mathbb{E}_{\pi_{\theta}} G(\tau) &\approx \frac{1}{L} \sum_{\tau} \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) G(\tau) \\ &\approx \frac{1}{L} \sum_{\tau} \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \underbrace{\sum_{t'=t}^T \gamma^{t'-t} R(s_{t'}, a_{t'})}_{= Q^{\pi}(s_t, a_t)}\end{aligned}$$

- Idea: we can use a critic (e.g., a NN as in DQN) to estimate the Q and learn two sets of parameters separately
 - Actor: update θ by policy gradient
 - Critic: Update parameters ϕ of v_{ϕ}^{π} , e.g., by n-step TD
- We call such algorithms *actor-critic algorithms*

Typically, we estimate $v^{\pi}(s_t; \phi)$ explicitly, and then sample

$$q^{\pi}(s_t, a_t) \approx G_t^{(n)}, \text{ i.e. } \hat{G}_t^{(1)} = R_t + \gamma v^{\pi}(s_{t+1}; \phi)$$

Actor-Critic Methods

Advantage Actor Critic (A2C)

- Introduce a baseline:

$$\begin{aligned}\nabla_{\theta} \mathbb{E}_{\pi_{\theta}} G(\tau) &\approx \frac{1}{L} \sum_{\tau} \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \underbrace{\sum_{t'=t}^T \gamma^{t'-t} R(s_{t'}, a_{t'}) - b(s_t)}_{= Q^{\pi}(s_t, a_t)} \\ &= \frac{1}{L} \sum_{\tau} \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \boxed{(\hat{G}_t - b(s_t))} \\ &\quad := A^{\pi}(s_t, a_t)\end{aligned}$$

- Calculate via MC estimation:

$$A^{\pi}(s_t, a_t) = R(s_t, a_t) - V^{\pi}(s_t)$$

Actor-Critic Methods

Advantage Actor Critic (A2C)

- Calculate via TD error:

$$\begin{aligned} A^\pi(s_t, a_t) &= Q^\pi(s_t, a_t) - V^\pi(s_t) \\ &= r + \gamma \cdot v^\pi(s'_t) - v^\pi(s_t) \end{aligned}$$

- Or multi-step TD error: "Generalized Advantage Estimation (GAE)"

$$\begin{aligned} \hat{A}_t^{(1)} &:= \delta_t^V &= -V(s_t) + r_t + \gamma V(s_{t+1}) \\ \hat{A}_t^{(2)} &:= \delta_t^V + \gamma \delta_{t+1}^V &= -V(s_t) + r_t + \gamma V(s_{t+1}) + \gamma^2 V(s_{t+2}) \\ & & \dots \\ \hat{A}_t^{(k)} &:= \sum_{l=0}^{k-1} \gamma^l \delta_{t+l}^V &= -V(s_t) + r_t + \gamma r_{t+1} + \dots + \gamma^{k-1} r_{t+k-1} + \gamma^k V(s_{t+k}) \\ \hat{A}_t^{(\infty)} &= \sum_{l=0}^{\infty} \gamma^l \delta_{t+l}^V &= -V(s_t) + \sum_{l=0}^{\infty} \gamma^l r_{t+l} \end{aligned}$$

- Less variance, more bias than MC

Actor-Critics Methods

Summary

- The policy gradient has many forms:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(s, a) G_t]$$

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(s, a) (G_t - b(s_t))]$$

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(s, a) \hat{A}_t^{(\infty)}]$$

$$\nabla_{\theta} J(\theta) = \nabla_{\theta} Q_t(s, \pi_{\theta}(s))$$

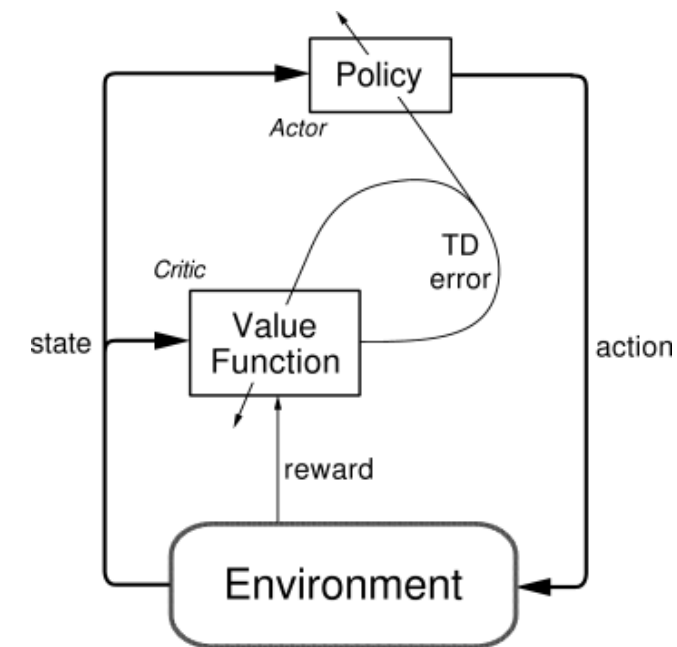
REINFORCE

REINFORCE w/ baseline

advantage actor-critic

deterministic policy gradient
(see DDPG)

- Each leads to a stochastic gradient ascent algorithm
- Critic uses policy evaluation (e.g., MC or TD) to estimate $Q^{\pi}(s, a)$ or $V^{\pi}(s)$



Sutton et al.: Reinforcement learning: An introduction. 2018.

Exercise Sheet 8

Advantage Actor Critic (A2C)



Better Control of Improvement Steps

Potential problems with gradient-based updates

- Note: the advantage function (which is a noisy estimate) may not be accurate
 - Too large steps may lead to a disaster (even *if* the gradient is *correct*)
 - Too small steps are also bad
 - Mathematical formulization:
 - First-order derivatives approximate the (parameter) surface to be flat
 - But if the surface exhibits high curvature it gets dangerous
 - Projection: small changes in parameter space might lead to large changes in policy space!
 - **Parameters θ get updated to areas too far out of the range from where previous data was collected**
- Regularize updates to the policy parameters such that the policy does not change too much



Images taken from https://medium.com/@jonathan_hui/rl-trust-region-policy-optimization-trpo-explained-a6ee04e0e0e9 and <http://www.taiwanoffthebeatentrack.com/2012/08/23/mount-hua-华山-the-most-dangerous-hike-in-the-world/>

Better Control of Improvement Steps

Natural Policy Gradients

TRPO tries to approximate inverse of Fisher information (i.e. Hessian)

- First-order derivatives approximate the (parameter) surface to be flat
- But if the surface exhibits high curvature it gets dangerous
- Small changes in parameter space might lead to large changes in policy space!

What we essentially do

(optimization perspective on 1st order gradient descent)

$$\theta' \leftarrow \arg \max_{\theta'} (\theta' - \theta)^T \nabla_{\theta} J(\theta), \text{ subject to } \|\theta' - \theta\|^2 \leq \epsilon$$

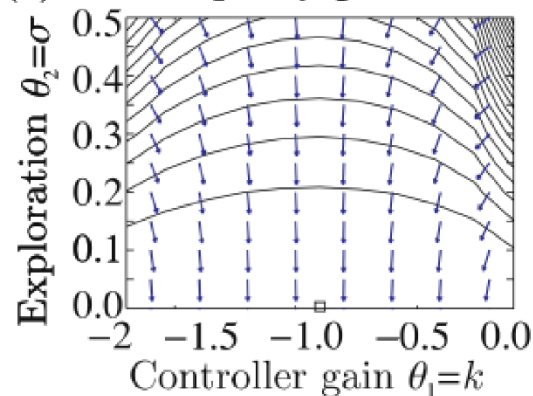
What we want to do

(incorporate 2nd order information)

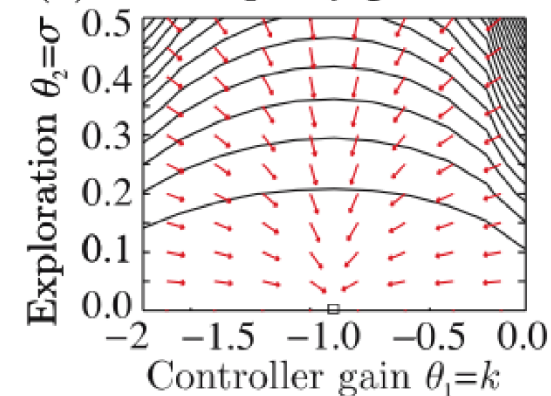
$$\theta' \leftarrow \arg \max_{\theta'} (\theta' - \theta)^T \nabla_{\theta} J(\theta), \text{ subject to } \|\theta' - \theta\|_F^2 \leq \epsilon$$

$$\rightarrow \theta \leftarrow \theta + \alpha \mathbf{F}^{-1} \nabla_{\theta} J(\theta) \text{ with e.g. KL-divergence}$$

(a) 'Vanilla' policy gradients



(b) Natural policy gradients



Peters et al.: Natural Actor-Critic. 2018

Better Control of Improvement Steps

Proximal Policy Optimization (PPO)

- The main motivation behind PPO is the same as for TRPO:
 - Make the biggest possible improvement step
 - Do not step too far such that the performance accidentally collapses
- PPO addresses the shortcomings of TRPO:
 - PPO uses 1st order methods with a few tricks
 - Significantly simpler to implement
 - Shows similar performance to TRPO (empirically)

removes the incentive for moving r_t outside of the interval $[1 - \epsilon, 1 + \epsilon]$

- PPO-Clip: The PPO objective we want to maximize is given by

$$= g(\epsilon, \hat{A}_t(s, a))$$

$$g(\epsilon, \hat{A}_t) = \begin{cases} (1 + \epsilon)A, & \text{if } A \geq 0 \\ (1 - \epsilon)A, & \text{if } A < 0 \end{cases}$$

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

where ϵ is a hyperparameter (i.e., 0.1 or 0.2) that defines how far π_{new} may go away from π_{old}

➤ the final objective is a lower bound (i.e., a pessimistic bound) on the unclipped objective

Thank you for your attention!