

535514: Reinforcement Learning

Lecture 15 — DDPG, TD3, and TRPO

Ping-Chun Hsieh

April 15, 2024

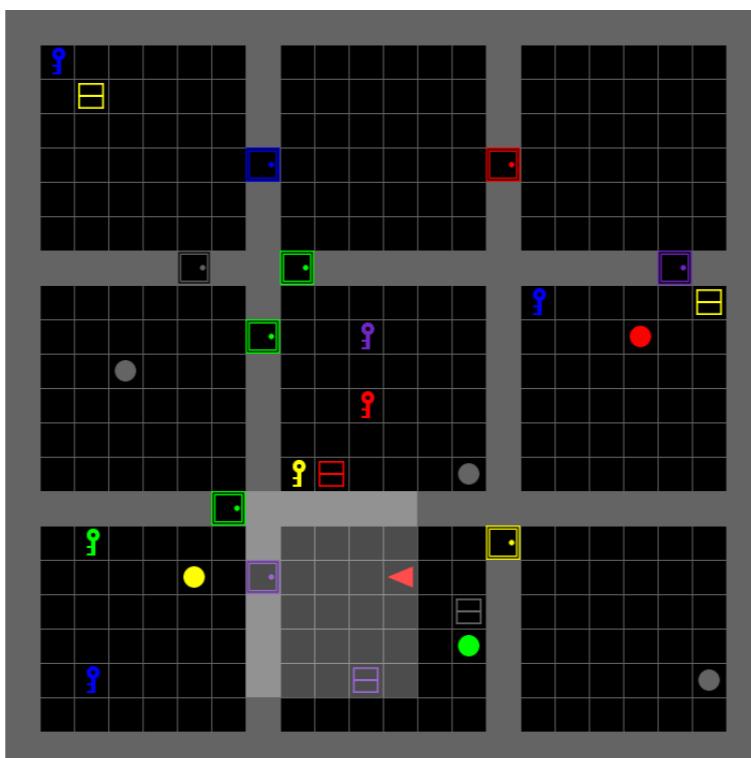
When Is “Exploration” Needed?

- Learning deterministic policies: For sure! Yes!
- How about “stochastic policies”? Do we need exploration in this case?
- How about “on-policy” learning? Do we need exploration in this case?

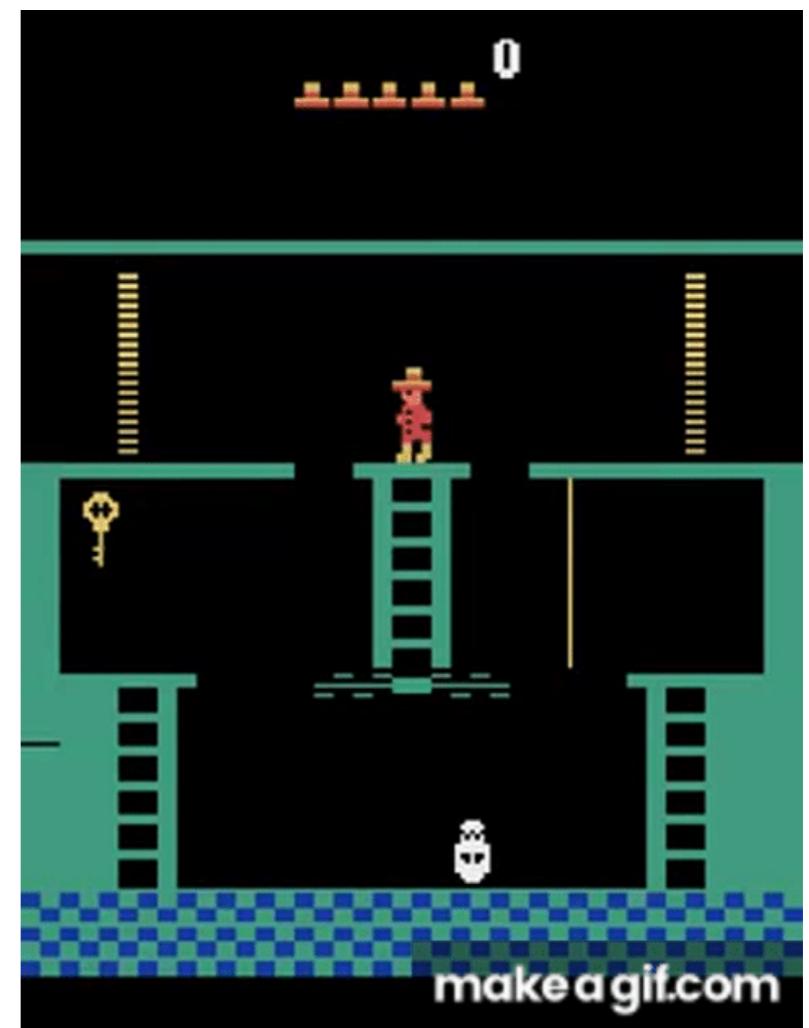
Some benchmark environments for exploration

*Sparse-reward
environment*

Minigrid

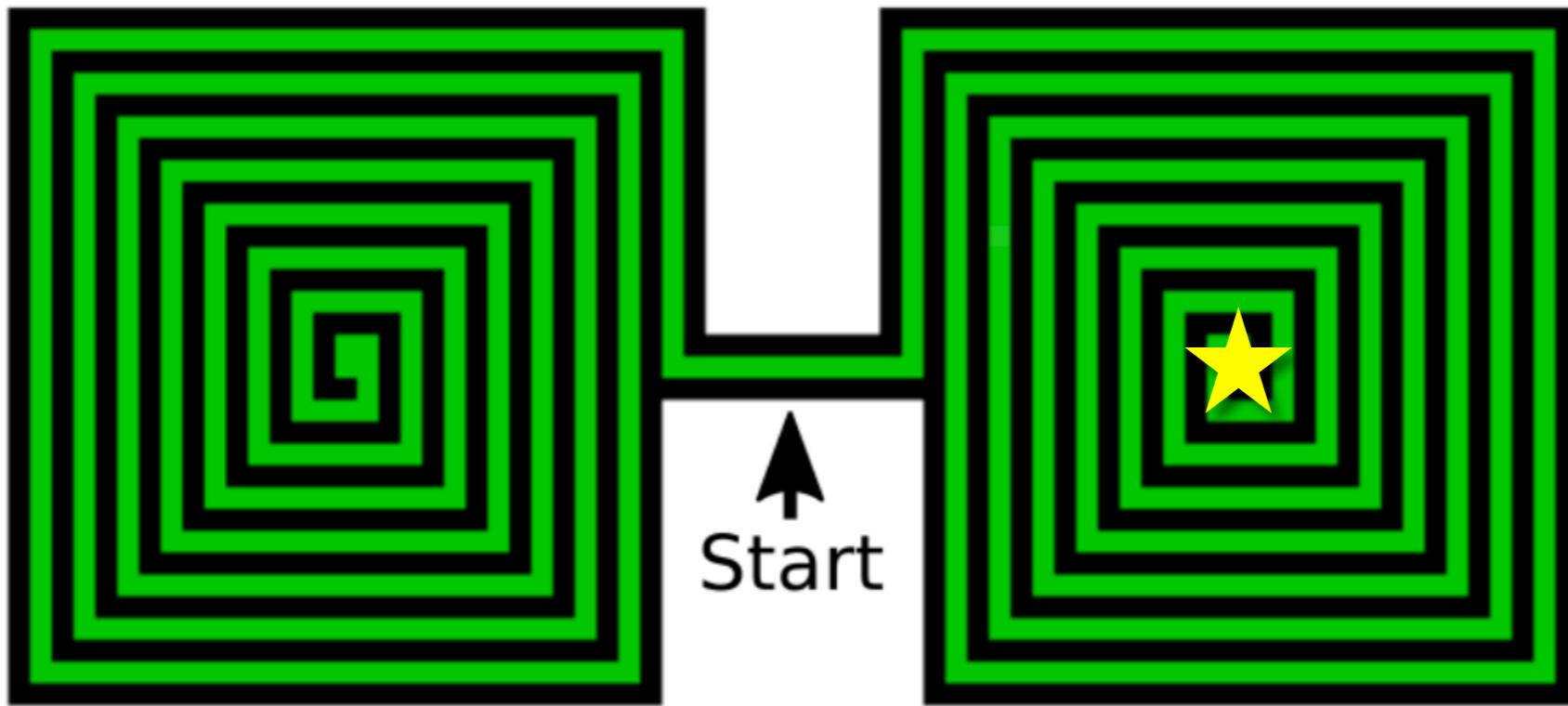


Montezuma’s Revenge in Atari



Does “random exploration” work here?

Issues With Random Exploration



Random exploration could be problematic for environments with sparse rewards

Figure Source: Ecoffet et al., “Go-Explore: a New Approach for Hard-Exploration Problems,” NeurIPS 2019

Question: Critic Design in OPDAC?

- Off-Policy Deterministic Actor-Critic (OPDAC):

Step 1: Initialize θ_0 , w_0 and step sizes α_θ , α_w

Step 2: Sample a trajectory $\tau = (s_0, a_0, r_1, \dots) \sim P_\mu^\beta$
For each step of the current trajectory $t = 0, 1, 2, \dots$

$$\Delta w_k \leftarrow \Delta w_k + \alpha_w (r_t + \gamma Q_{w_k}(s_{t+1}, \pi_\theta(s_{t+1})) - Q_{w_k}(s_t, a_t)) \nabla_w Q_w(s_t, a_t)|_{w=w_k}$$

$$\Delta \theta_k \leftarrow \Delta \theta_k + \alpha_\theta \gamma^t \left(\nabla_\theta \pi_\theta(s_t) \nabla_a Q_{w_k}(s_t, a)|_{a=\pi_\theta(s_t)} \right)$$

$$\theta_{k+1} \leftarrow \theta_k + \Delta \theta_k, w_{k+1} \leftarrow w_k + \Delta w_k = \nabla_\theta Q_{w_k}(s_t, \pi_\theta(s_t))|_{\theta=\theta_k}$$

- Question: Why no change on the critic in Off-Policy DAC?

Learning a Critic for OPDAC under VFA

- **Goal:** find \mathbf{w} that minimizes MSE between $Q^\pi(s, a)$ and $\hat{Q}(s, a; \mathbf{w})$

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \mathbb{E}_{s \sim \rho(s)} \left[(Q^\pi(s, a) - \hat{Q}(s, a; \mathbf{w}))^2 \right]$$

$\underbrace{\phantom{(Q^\pi(s, a) - \hat{Q}(s, a; \mathbf{w}))^2}}_{=:F(\mathbf{w})}$

- Find \mathbf{w}^* by iterative GD updates

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \mathbb{E}_{s \sim \rho(s)} \left[(Q^\pi(s, a) - \hat{Q}(s, a; \mathbf{w})) \nabla_{\mathbf{w}} \hat{Q}(s, a; \mathbf{w}) \right]$$

- Since true Q^π is unknown, let's use **bootstrapping** (e.g.TD(0))

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \mathbb{E}_{s \sim \rho(s)} \left[(r + \gamma \hat{Q}_{\mathbf{w}}(s', \pi(s')) - \hat{Q}(s, a; \mathbf{w})) \nabla_{\mathbf{w}} \hat{Q}(s, a; \mathbf{w}) \right]$$

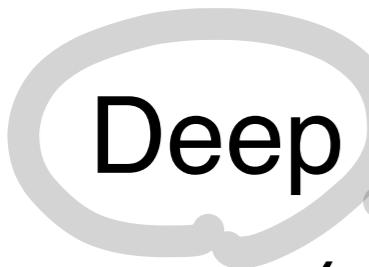
- **Question:** To learn a critic for OPDAC, what distribution $\rho(s)$ shall we choose?

Option 1: μ^π
Option 2: μ_β

Option 3: Uniform

On-Policy vs Off-Policy Methods

| | Policy Optimization | Value-Based | Model-Based | Imitation-Based |
|-------------------|---|--|--|---------------------------------|
| On-policy | Exact PG REINFORCE (w/i baseline) A2C On-policy DAC TRPO Natural PG (NPG) PPO-KL & PPO-Clip | Epsilon-Greedy MC Sarsa Expected Sarsa | Model-Predictive Control (MPC) PETS | IRL GAIL IQ-Learn RLHF |
| Off-policy | Off-policy DPG & DDPG Twin Delayed DDPG (TD3) | Q-learning Double Q-learning DQN & DDQN C51 / QR-DQN / IQN Soft Actor-Critic (SAC) | | |



Deep Deterministic Policy Gradient (DDPG)

(= OPDAC with Deep Neural Nets)

Lillicrap et al., “Continuous control with deep reinforcement learning”, ICLR 2016

What is DDPG?

- ▶ **DDPG**: Combine OPDAC with NN nonlinear VFA
 - ▶ **Off-policy**: Exploration
 - ▶ **Nonlinear VFA**: Convergence issue
- ▶ To tackle the above issues, DDPG applies several techniques:
 - (T1) Experience replay (for data-efficient off-policy learning)
 - (T2) Ornstein-Uhlenbeck process for exploration (optional)
 - (T3) Target networks

(T1) Experience Replay

- ▶ Main idea:

1. Store the previous experiences (s, a, s', r) into a buffer
2. Sample a mini-batch from the buffer at each step $\approx \frac{d\beta}{d\mu}$

(similar to mini-batch SGD in supervised learning)

- ▶ Purposes:

1. Better estimate of DPG: Break correlations between successive steps in a trajectory (“more stable learning”, as stated in many papers)

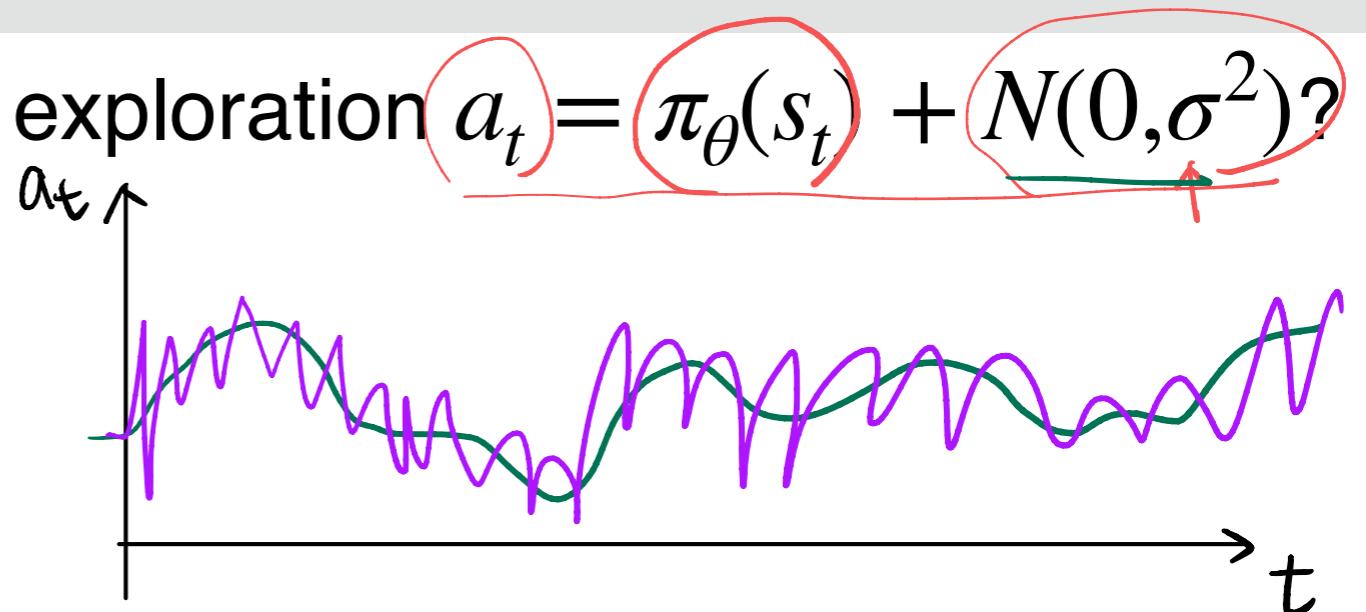
Off-policy DPG:

$$\nabla_{\theta} J_{\beta}(\pi_{\theta}) \approx E_{\frac{d\beta}{d\mu}} [\nabla_{\theta} \text{TL}_{\theta}(s) \cdot \nabla_{\theta} Q^{\pi_{\theta}}(s, a)]$$

2. Better data efficiency: Fewer interactions with environment needed for convergence

(T2) Ornstein-Uhlenbeck Process for Exploration

- Issue with Gaussian noise exploration $a_t = \pi_\theta(s_t) + N(0, \sigma^2)$?



- Ornstein-Uhlenbeck (OU) process: Similar to Gaussian policies, but with temporal correlation

$$dx_t = \theta(\mu - x_t)dt + \sigma \cdot dW_t$$

Brownian motion

If $\theta=0$:

X_t is a Brownian Motion

- Discrete-time approximation of OU:

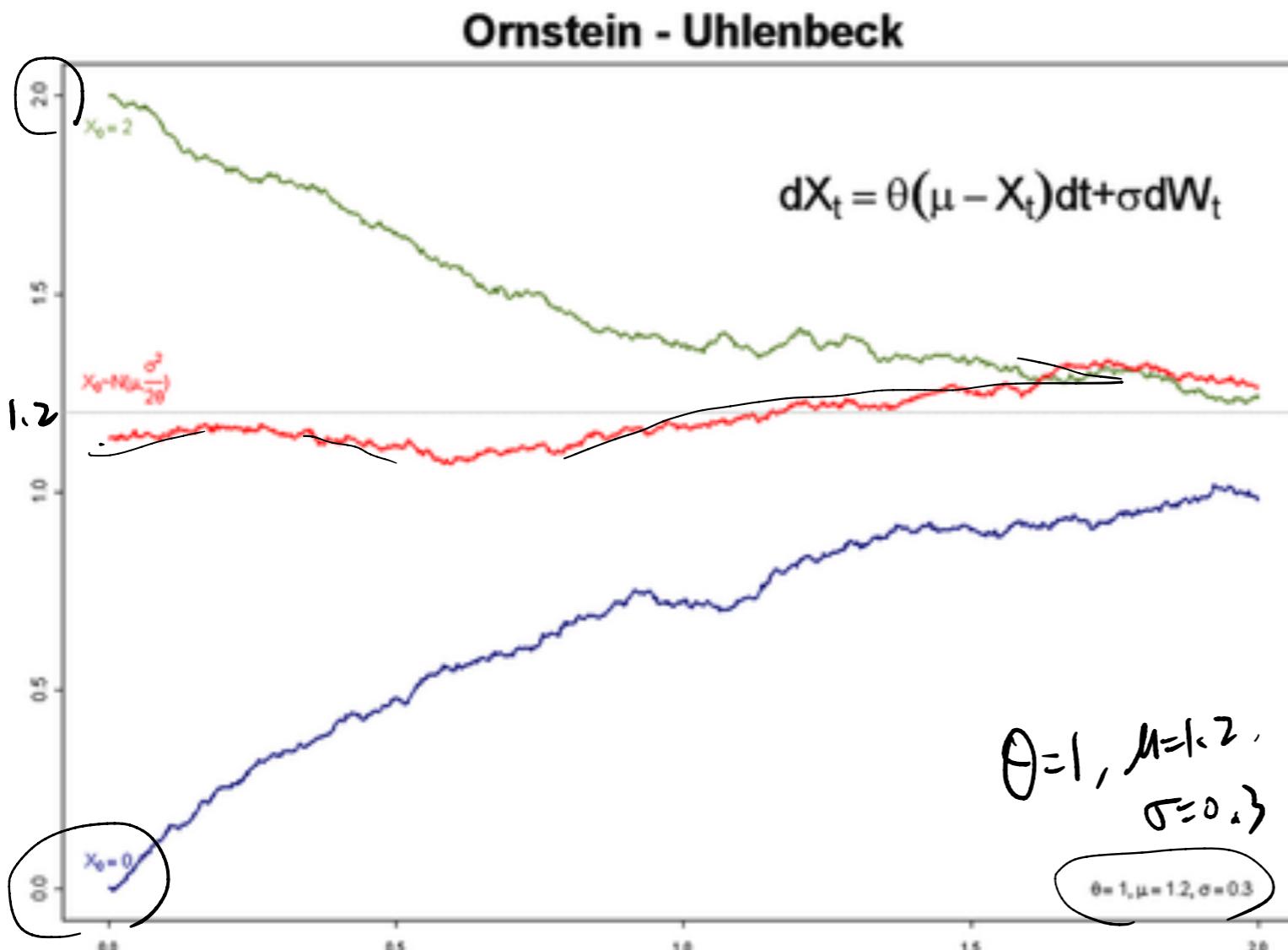
$$X_{t+1} - X_t = \theta(\mu - X_t)\Delta t + \sigma \cdot \Delta W_t$$

i.i.d. normal random variables $\sim \mathcal{N}(0, \Delta t)$



Example of OU Process

$$X_{t+1} - X_t = \underbrace{1 \cdot (1.2 - X_t)}_{N(0, 1)} + \sigma dW$$



(Same OU process with 3 different initial conditions)

How about a sequence of i.i.d. Gaussian random variables?

(T3) Target Networks

① Soft update for target

$$\bar{Q} \leftarrow 0.99\bar{Q} + 0.01 \cdot Q$$

② $\bar{Q} \leftarrow Q$ every N time steps

- ▶ Idea: Use separate *target networks* ($\bar{\pi}_\theta$ for actor, \bar{Q}_w for critic) that are updated only periodically
- ▶ For DDPG, the critic update with target networks

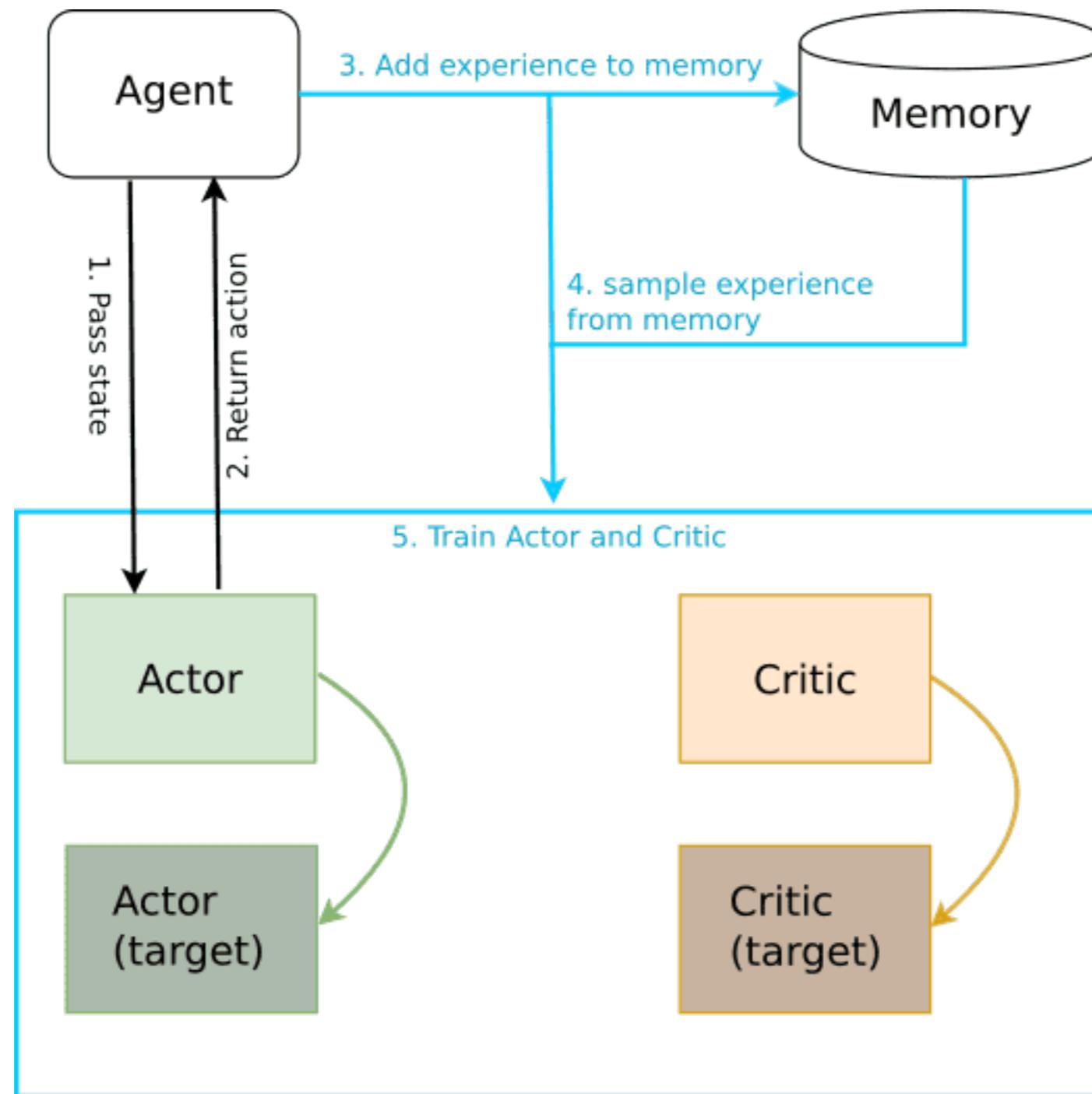
$$\Delta w_k \leftarrow \Delta w_k + \alpha_w \left(r_t + \gamma \bar{Q}_{w_k}(s_{t+1}, \bar{\pi}_\theta(s_{t+1})) - Q_{w_k}(s_t, a_t) \right) \nabla_w Q_w(s_t, a_t)|_{w=w_k}$$

- ▶ Similar to value iteration:

$$V(s) \leftarrow \max_a R(s, a) + \gamma \sum_{s'} P(s'|s, a) \bar{V}(s)$$

- ▶ Purpose: Mitigate divergence

DDPG Architecture



Pseudo Code of DDPG Algorithm

Algorithm 1 DDPG algorithm

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

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

Initialize replay buffer R

for episode = 1, M **do**

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

 Receive initial observation state s_1

for t = 1, T **do**

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

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

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

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

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

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

 Update the actor policy using the sampled policy gradient:

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

 Update the target networks:

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

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

end for

end for

2 evaluation networks
and 2 target networks

action drawn from a deterministic
policy with exploration

experience replay

Update actor and critic

→ This can be viewed as
the gradient of Q w.r.t. θ

Update target networks
(small τ for stability)

Twin Delayed DDPG (TD3)

Fujimoto et al., “Addressing Function Approximation Error in Actor-Critic Methods,”
ICML 2018

Overestimation Bias in DDPG

- ▶ **Overestimation bias:** Estimated Q values are larger than the true ones

$$\mathbb{E}_s[Q_w(s, \pi_\theta(s))] \geq \mathbb{E}_s[Q^{\pi_\theta}(s, \pi_\theta(s))]$$

- ▶ **Question:** How to empirically measure such bias?
 - ▶ **Empirical evidence:** 

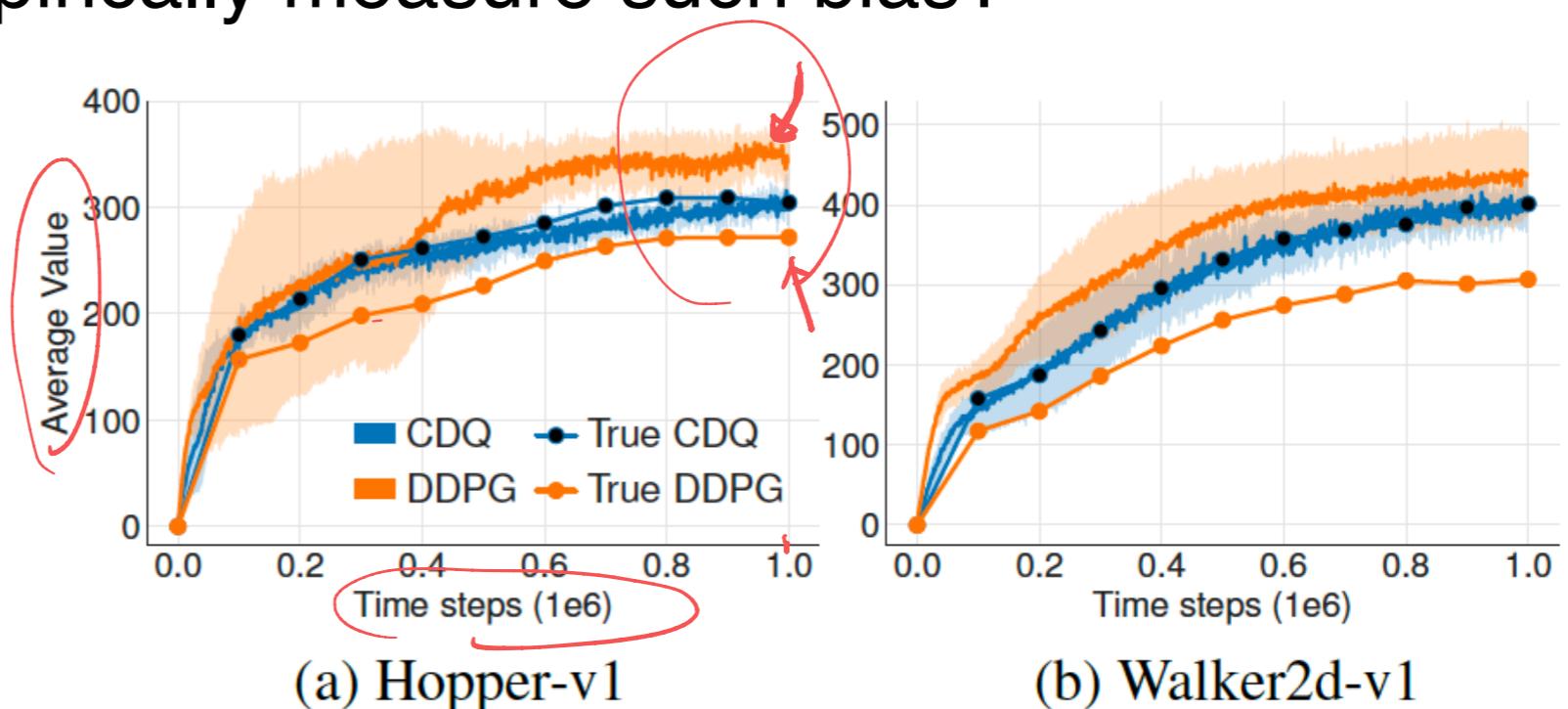
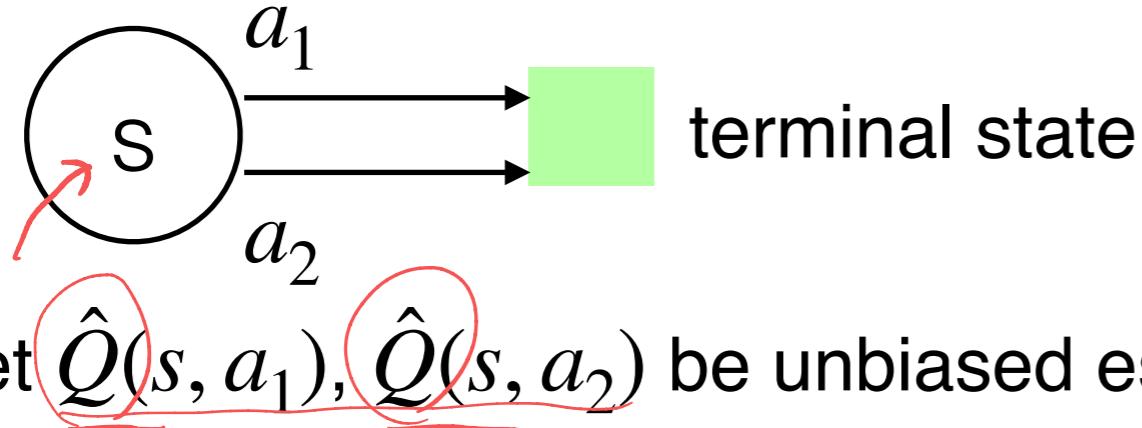


Figure 1. Measuring overestimation bias in the value estimates of DDPG and our proposed method, Clipped Double Q-learning (CDQ), on MuJoCo environments over 1 million time steps.

Overestimation Bias: A Motivating Example

- Example: Consider a 1-state MDP with 2 actions



$$\pi Q(s, a_1) = 0.5$$

$$\pi Q(s, a_2) = 0.5$$

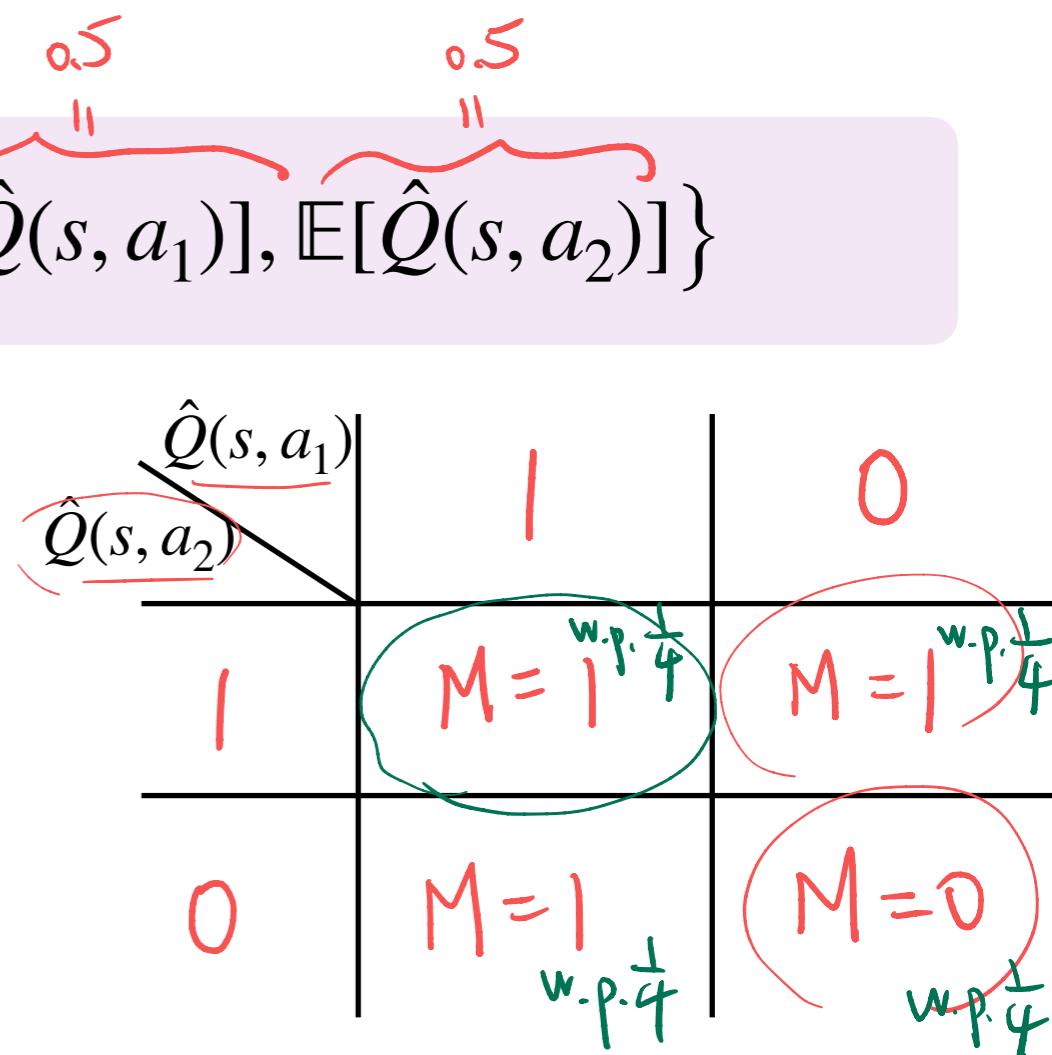
- Let $\hat{Q}(s, a_1)$, $\hat{Q}(s, a_2)$ be unbiased estimators (each based on 1 sample)

- An Interesting Fact:

$$\underbrace{\mathbb{E}[\max\{\hat{Q}(s, a_1), \hat{Q}(s, a_2)\}]}_{=:M} > \max\left\{\mathbb{E}[\hat{Q}(s, a_1)], \mathbb{E}[\hat{Q}(s, a_2)]\right\}$$

$$\max\left\{\mathbb{E}[\hat{Q}(s, a_1)], \mathbb{E}[\hat{Q}(s, a_2)]\right\} = 0.5$$

$$\mathbb{E}\left[\max\{\hat{Q}(s, a_1), \hat{Q}(s, a_2)\}\right] = 0.75$$

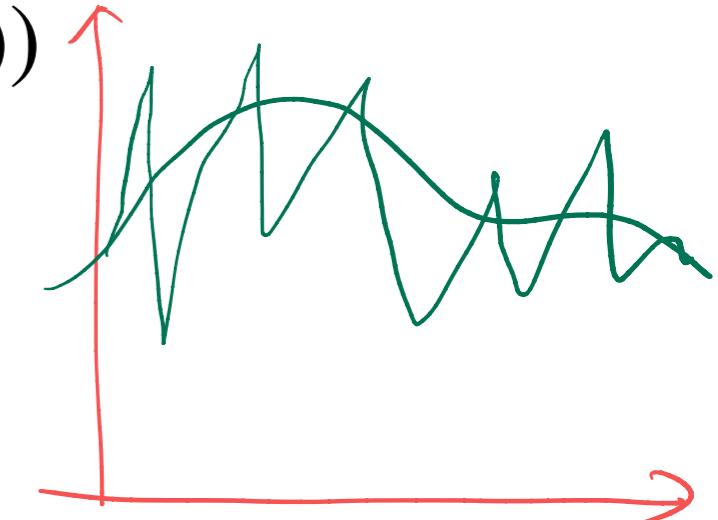


“Clipped Double-Q” (CDQ) in TD3

- ▶ Heuristic 1: Use two Q networks and take the minimum

DDPG’s TD-target: $y = r + \gamma \bar{Q}_w(s', \pi_\theta(s'))$

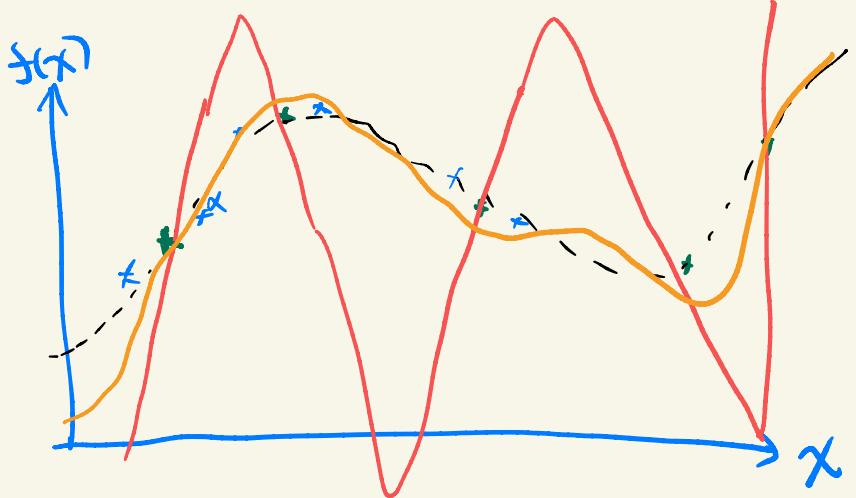
TD3’s TD-target: $y = r + \gamma \min_{i=1,2} \bar{Q}_{w_i}(s', \pi_\theta(s'))$



- ▶ Heuristic 2: Adding noise for a smooth Q

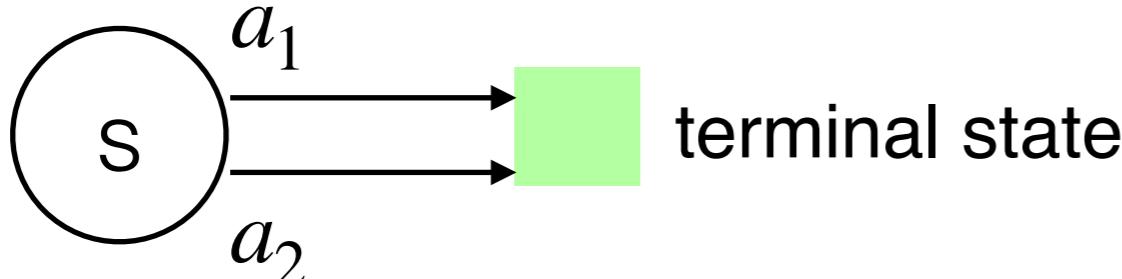
TD3’s clipped noise: $y = r + \gamma \min_{i=1,2} \bar{Q}_{w_i}(s', \pi_\theta(s') + \epsilon)$

$$\epsilon \sim \text{Clip}(N(0, \sigma^2), \epsilon_{\min}, \epsilon_{\max})$$



Mitigating Overestimation Bias via CDQ: An Example

- Example: Consider a 1-state MDP with 2 actions

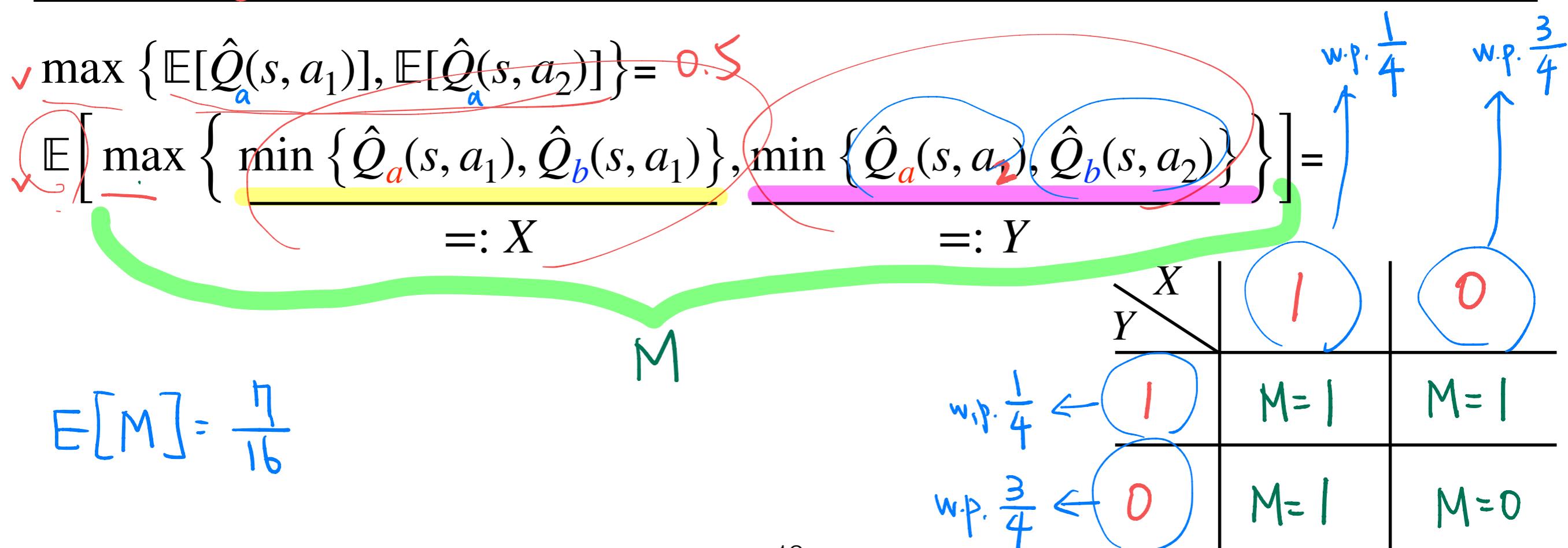


$$r(s, a_1) \sim \text{Bernoulli}(0.5)$$

$$r(s, a_2) \sim \text{Bernoulli}(0.5)$$

CDQ uses twin Q functions:

- Let $\hat{Q}_a(s, a_1), \hat{Q}_a(s, a_2)$ be the first unbiased Q estimate
- Let $\hat{Q}_b(s, a_1), \hat{Q}_b(s, a_2)$ be the second unbiased Q estimate



A2C/DPG/DDPG/TD3 \approx Policy Iteration, But With a Slight Difference

Policy iteration:

- Step 1: **Policy evaluation** for the current policy (i.e., find $Q^\pi(s, a)$)
- Step 2: **Policy improvement** based on Bellman optimality equations

A2C/DPG/DDPG/TD3:

- Step 1: Estimate $Q^{\pi_\theta}(s, a) \approx Q_w(s, a)$ for the current policy by TD
 - Step 2: Use $Q_w(s, a)$ to improve the policy by PG
(deterministic or stochastic)
-
- ▶ **Question:** Is $V^{\pi_\theta}(\mu)$ guaranteed to be improved in Step 2?

Trust Region Policy Optimization (TRPO)

Schulman et al., “Trust Region Policy Optimization”, ICML 2015

Strict Policy Improvement

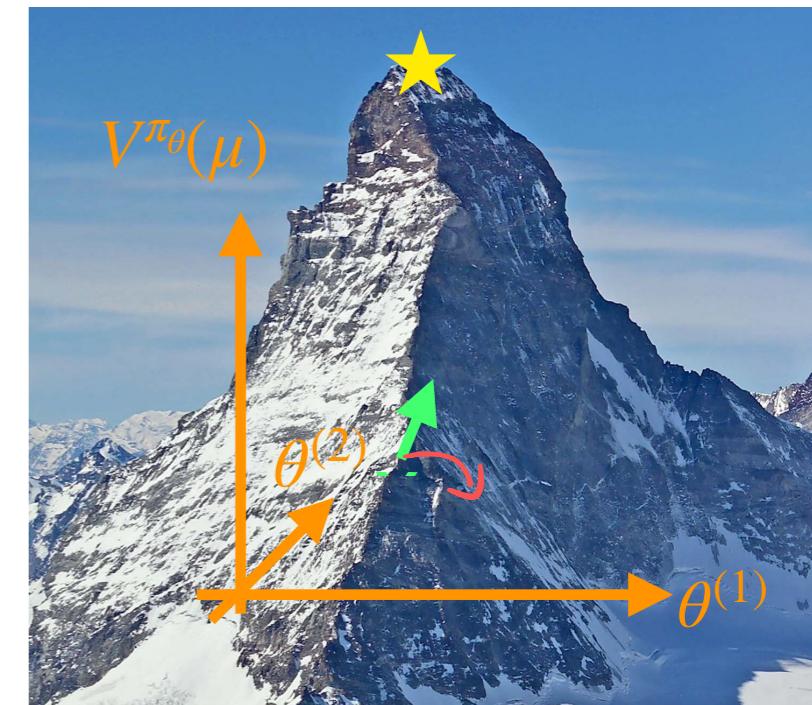
- Learning rate is critical for policy improvement

“Gradient ascent” uses first-order approximation

First-order approximation is accurate only
in a small neighborhood

Recall: Taylor expansion

$$f(\theta + \Delta\theta) = f(\theta) + \underbrace{\nabla f(\theta)^T \Delta\theta}_{\text{first-order}} + \Delta\theta^T \cdot \nabla^2 f(\theta) \Delta\theta + \dots$$



- Question: How to guarantee strict improvement?

TRPO: 1-Slide Summary

Idea: TRPO iteratively updates the policy by solving the following:

In each iteration k , the policy is updated from π_{θ_k} to $\pi_{\theta_{k+1}}$

$$\pi_{\theta_{k+1}} = \arg \max_{\theta} \mathbb{E}_{s \sim d_{\mu}^{\pi_{\theta_k}}, a \sim \pi_{\theta_k}(\cdot | s)} \left[\frac{\pi_{\theta}(a | s)}{\pi_{\theta_k}(a | s)} A^{\pi_{\theta_k}}(s, a; \cdot) \right]$$

subject to $D(\pi_{\theta_k}, \pi_{\theta}) \leq \delta$ ← “trust region”



TRPO wants to achieve policy improvement via
constrained optimization

State-Wise Performance Difference Lemma

- ▶ **State-Wise Performance Difference** [Kakade & Langford, 2002]

For any two policies π_{old}, π_{new} and any initial state s , we have

$$V^{\pi_{new}}(s) - V^{\pi_{old}}(s) = \frac{1}{1-\gamma} \mathbb{E}_{s' \sim d_s^{\pi_{new}}} \mathbb{E}_{a' \sim \pi_{new}(\cdot | s')} \left[A^{\pi_{old}}(s', a') \right]$$

- ▶ Question 1: How to interpret this result?
- ▶ Question 2: Under **tabular policies**, what will we have if π_{old} is obtained from π_{new} by “*one-step policy improvement*”?
- ▶ Question 3: How about **neural policies**? Could we still possibly achieve monotonic policy improvement?

Average Performance Difference Lemma

- ▶ **Idea:** Though we could NOT achieve monotonic policy improvement, could we resort to improvement in average performance?
 - ▶ **Question:** Could we find the “average difference”
$$\sum_s \mu(s) (V^{\pi_{new}}(s) - V^{\pi_{old}}(s))$$
 by state-wise performance difference lemma?

$$V^{\pi_{new}}(\mu) - V^{\pi_{old}}(\mu) = \frac{1}{1-\gamma} \mathbb{E}_{s' \sim d_\mu^{\pi_{new}}} \mathbb{E}_{a' \sim \pi_{new}(\cdot | s')} \left[A^{\pi_{old}}(s', a') \right]$$

To simplify notations, let's use

$$\eta(\pi_{new}) \equiv (1 - \gamma)V_\mu(\pi_{new})$$

$$\eta(\pi_{old}) \equiv (1 - \gamma)V_\mu(\pi_{old})$$

Question: How about directly optimizing

$$\sum_s d_\mu^{\pi_{new}}(s) \sum_a \pi_{new}(a | s) A^{\pi_{old}}(s, a) = (1 - \gamma)(V_\mu(\pi_{new}) - V_\mu(\pi_{old}))$$

usually difficult to get (why?)

Idea: Approximate $d_\mu^{\pi_{new}}(s)$ by $d_\mu^{\pi_{old}}(s)$

$$(1 - \gamma)(V_\mu(\pi_{new}) - V_\mu(\pi_{old})) \approx \sum_s d_\mu^{\pi_{old}}(s) \sum_a \pi_{new}(a | s) A^{\pi_{old}}(s, a)$$

(called a “surrogate function”)

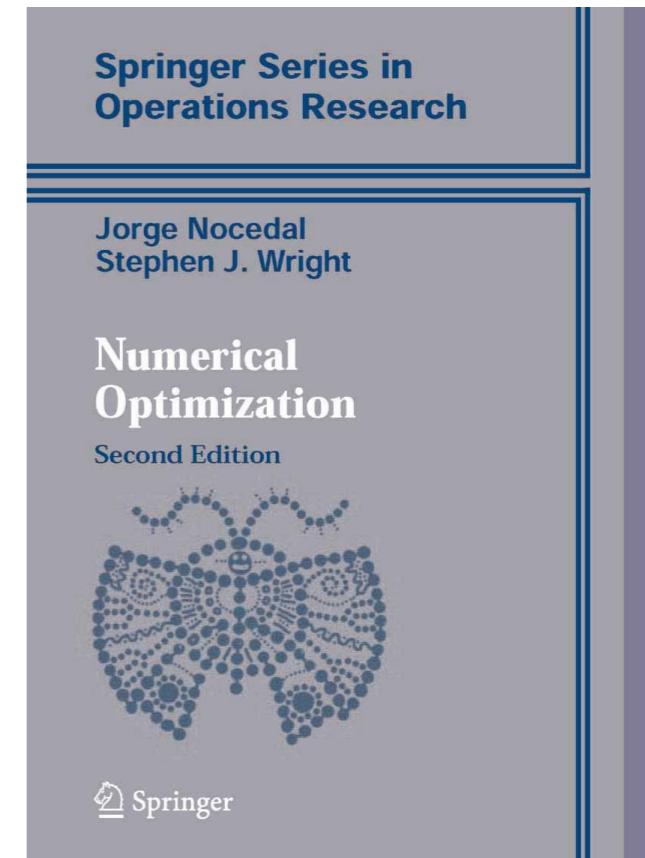
Question: How to ensure that $d_\mu^{\pi_{new}}(s)$ and $d_\mu^{\pi_{old}}(s)$ are close?

π_{new} and π_{old} need to be quite close

Trust Region Methods for Optimization!

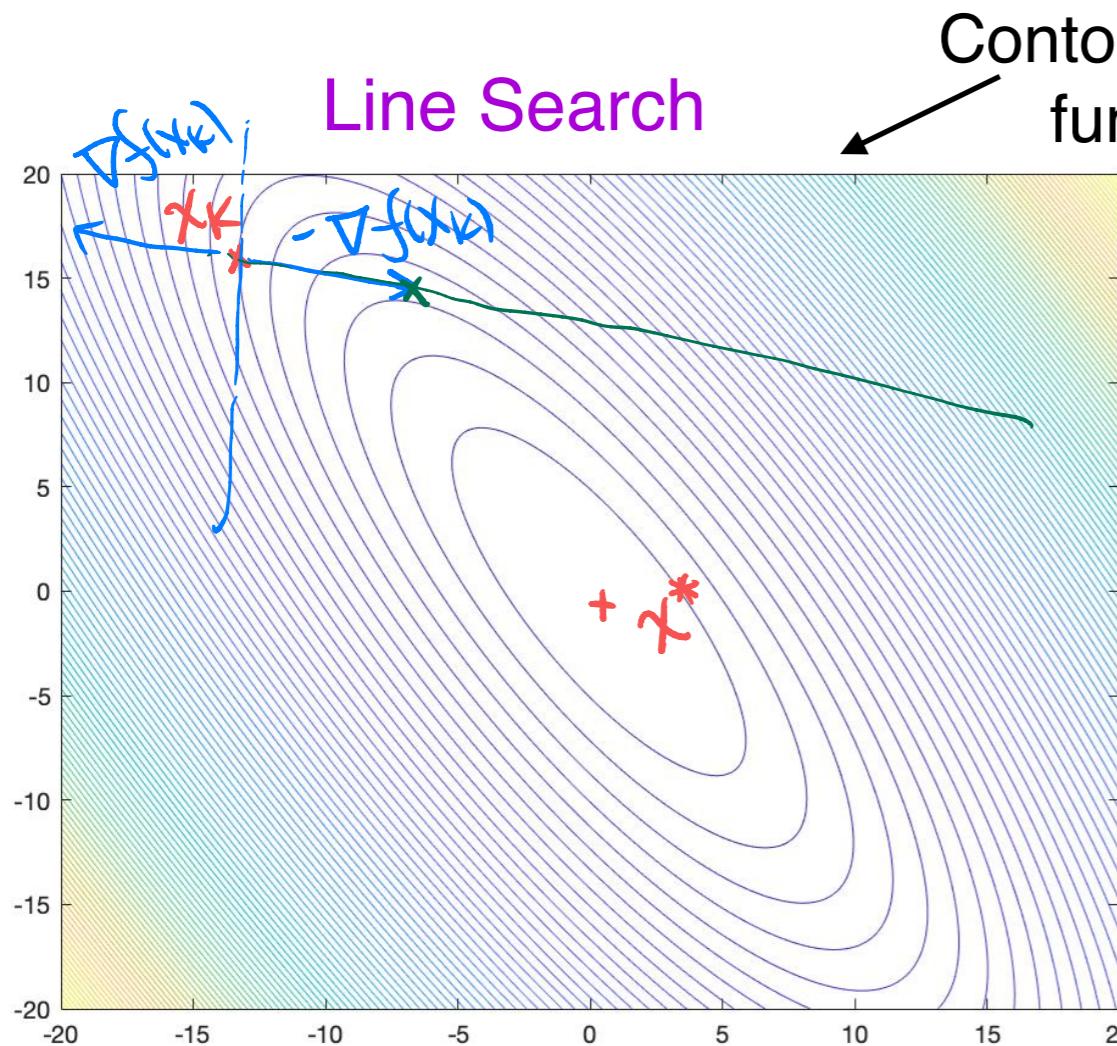
Trust Region Methods for Optimization

Jorge Nocedal and Stephen Wright, “Numerical Optimization”, 2006

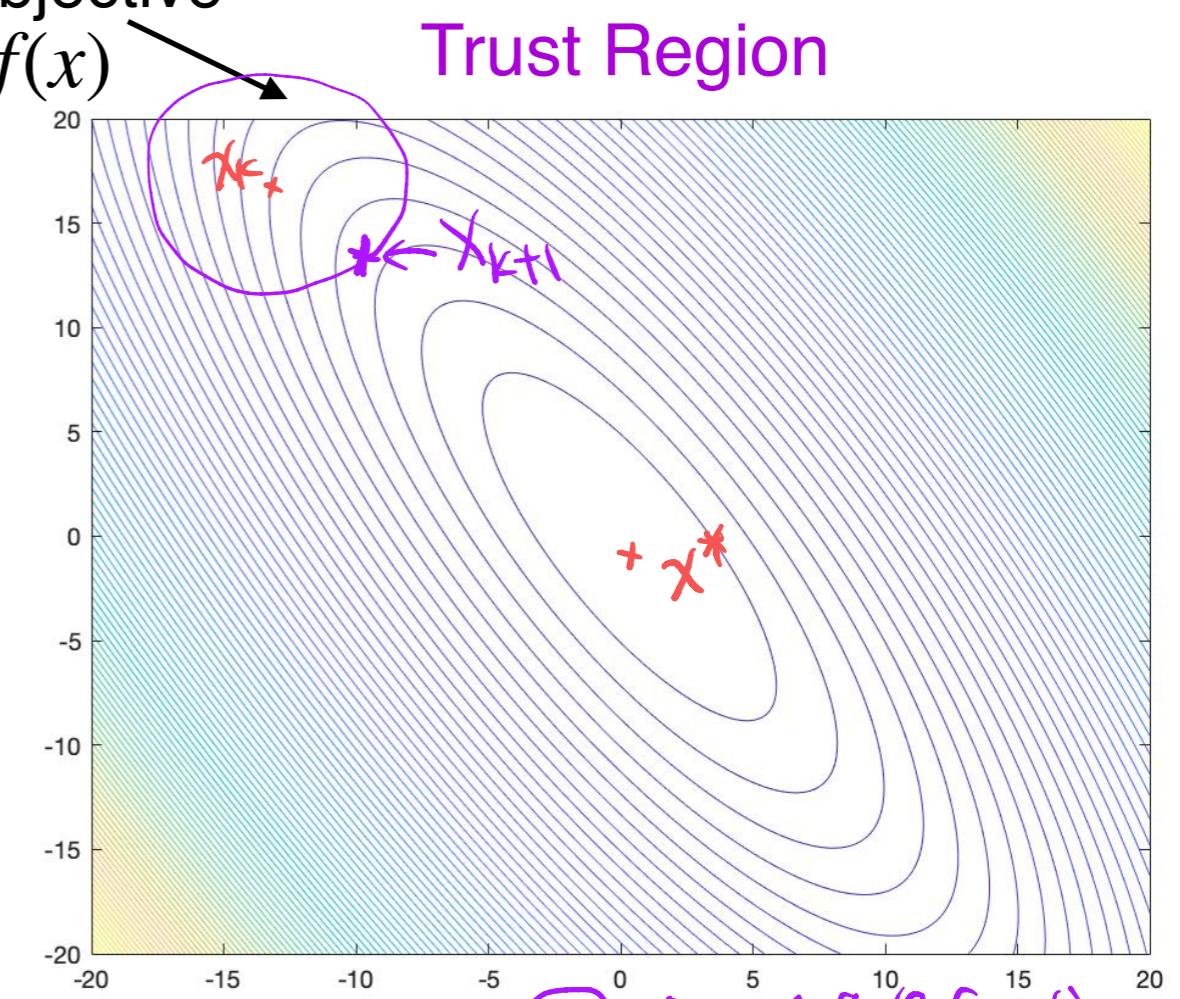


Trust Region vs Line Search

- ▶ 2 major types of iterative algorithms for numerical optimization:



$$x_{k+1} = x_k + \alpha_k \delta_k$$



surrogate function

$$\min_{\delta \in \Delta_k} \hat{f}_k(x_k + \delta)$$

trust region

- ▶ Remark: GD is a line search method

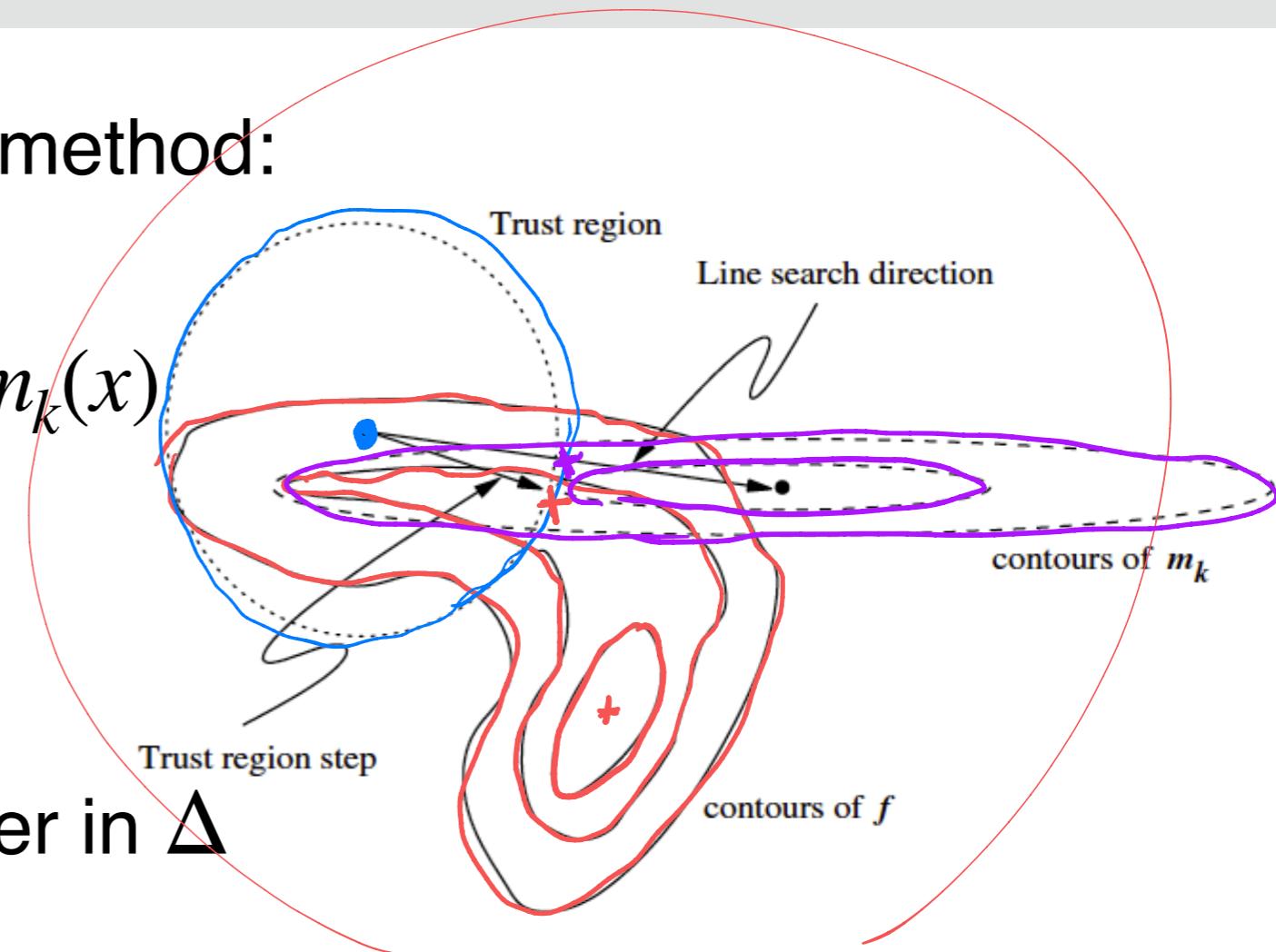
Trust Region Methods (Formally)

- ▶ 3 major steps of a trust region method:

(A1) Choose a surrogate function $m_k(x)$

(A2) Specify a trust region Δ

(A3) Find an approximate optimizer in Δ



- ▶ **Question:** How to choose the surrogate function?
- ▶ **Question:** How to specify a good trust region?

(A1) Surrogate Functions

- **Surrogate functions** are required to be easy to optimize or evaluate
- **Examples:**

Linear approx.

$$f(x_k + p) \approx m_k(p) := \underbrace{f(x_k) + \nabla f(x_k)^T p}_{\text{Taylor expansion (up to the 1st-order term)}}$$

Quadratic approx.

$$f(x_k + p) \approx m_k(p) := \underbrace{f(x_k) + \nabla f(x_k)^T p + \frac{1}{2} p^T \nabla^2 f(x_k) p}_{\text{Taylor expansion (up to the 2nd-order term)}}$$

- **Example:** Trust region optimization (TRO) subproblem with a quadratic surrogate function

$$\min m_k(p) = f(x_k) + \nabla f(x_k)^T p + \frac{1}{2} p^T \nabla^2 f(x_k) p$$

$$\text{s.t. } \|p\|_2 \leq \Delta_k$$

(If $\nabla^2 f(x_k)$ is positive semi-definite, then the problem is a QCQP and is convex)

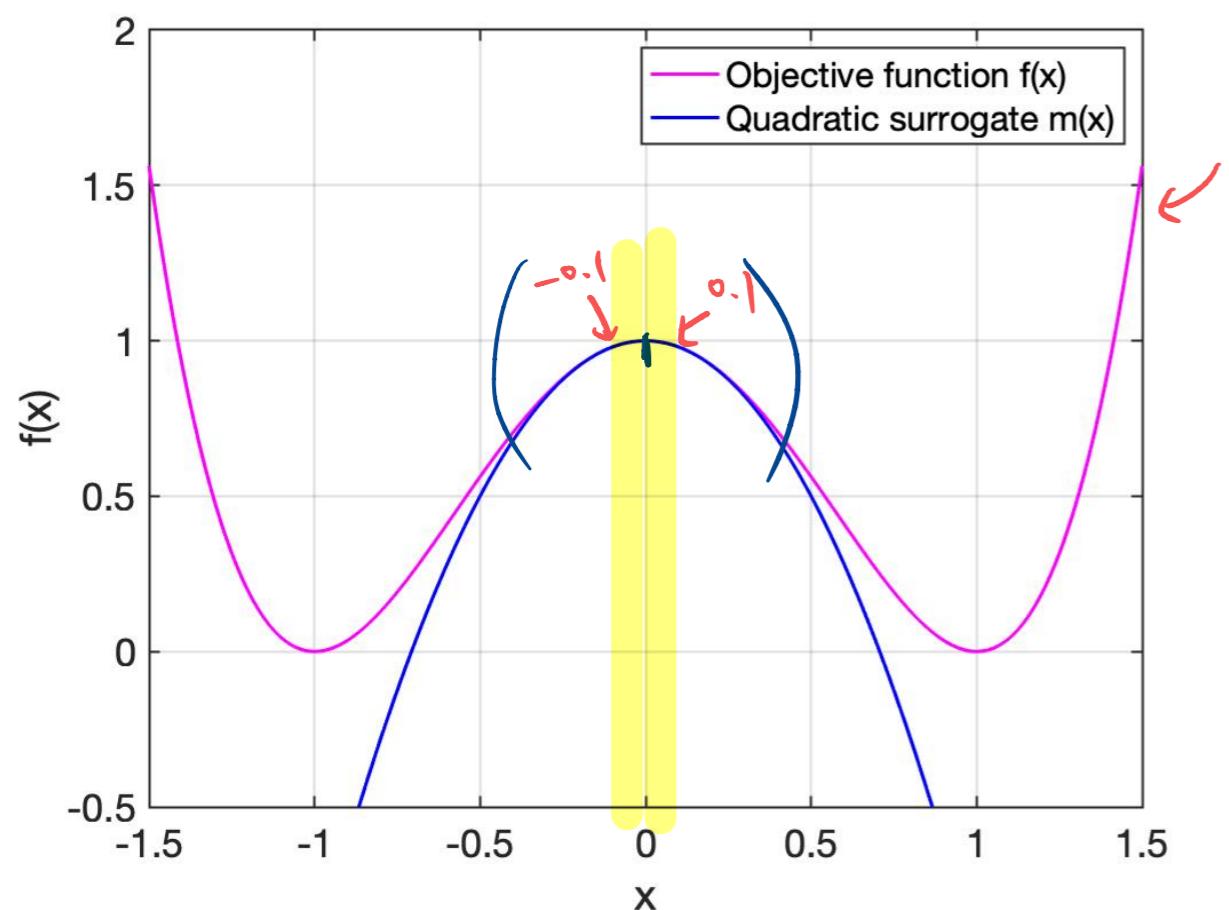
Example: Quadratic Surrogate Functions

- Example: Minimize $f(x) = (x^2 - 1)^2, x \in \mathbb{R}$

$$\min m_k(p) = f(x_k) + \nabla f(x_k)^T p + \frac{1}{2} p^T \nabla^2 f(x_k) p$$

$\frac{\partial f(x)}{\partial x} = 2(x^2 - 1) \cdot 2x$
 $= 4x^3 - 4x$
 $\frac{\partial^2 f(x)}{\partial x^2} = 12x^2 - 4$

- Question: What is the quadratic surrogate function at $x_k = 0$? Any benefit of TR compared to GD?



Trust Region Subproblem:

- $f(x_k) = 1$
- $\nabla f(x_k) = 0$
- $\nabla^2 f(x_k) = -4$

$$\begin{aligned} & \min_{p} -2p^2 + 1 \\ \text{s.t. } & \|p\|_2 \leq \Delta_k \end{aligned}$$

Suppose we choose $\Delta_k = 0.1$.

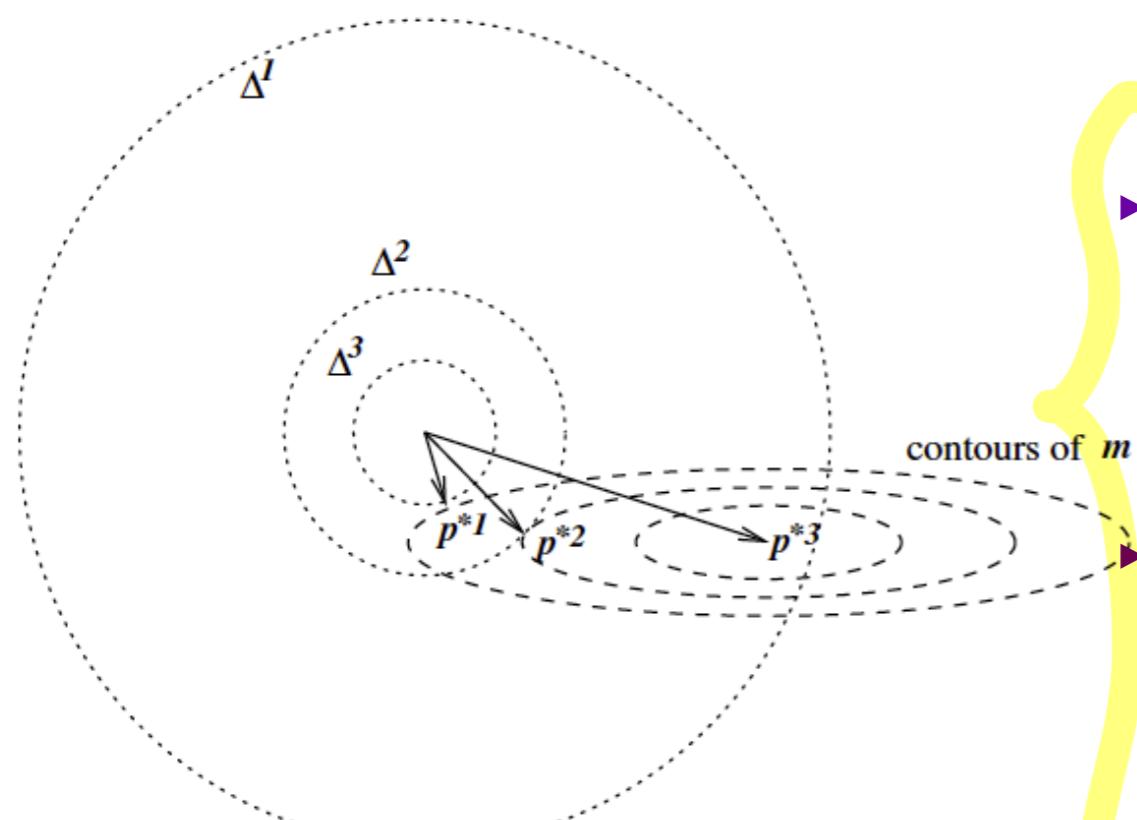
Then, what is x_{k+1} ?

(A2) Specify the Size of a Trust Region

- Trust region size is usually captured by the radius Δ_k

- Question: How to tune Δ_k ?

Define $\rho_k := \frac{f(x_k) - f(x_k + p)}{m_k(0) - m_k(p)}$



- If ρ_k is close to 0 or negative:

Decrease Δ

- If ρ_k is close to 1 and $\|p_k\| = \Delta_k$:

Increase Δ

- Otherwise: Δ unchanged

Let's apply TR method and build TRPO step by step!

Goal: The Ultimate TRPO Algorithm

- Trust-Region Policy Optimization (TRPO) Algorithm:

Step 1: Initialize θ_0

Step 2: For iteration $k = 0, 1, 2, \dots$

Step 2-1: Collect trajectories by running the current policy π_{θ_k}

Step 2-2: Obtain advantage $A^{\theta_k}(s, a)$ for the current policy π_{θ_k}

Step 2-3: Update the policy by solving

$$\theta_{k+1} = \arg \max_{\theta} \mathbb{E}_{s \sim d_{\mu}^{\pi_{\theta_k}}, a \sim \pi_{\theta_k}(\cdot | s)} \left[\frac{\pi_{\theta}(a | s)}{\pi_{\theta_k}(a | s)} A^{\theta_k}(s, a) \right]$$

subject to $\bar{D}_{KL}(\pi_{\theta_k} \| \pi_{\theta}) \leq \delta$

Recall: Trust Region Methods

- ▶ Recall: 3 major steps of a trust region method:

(A1) Choose a surrogate function $m_k(x)$

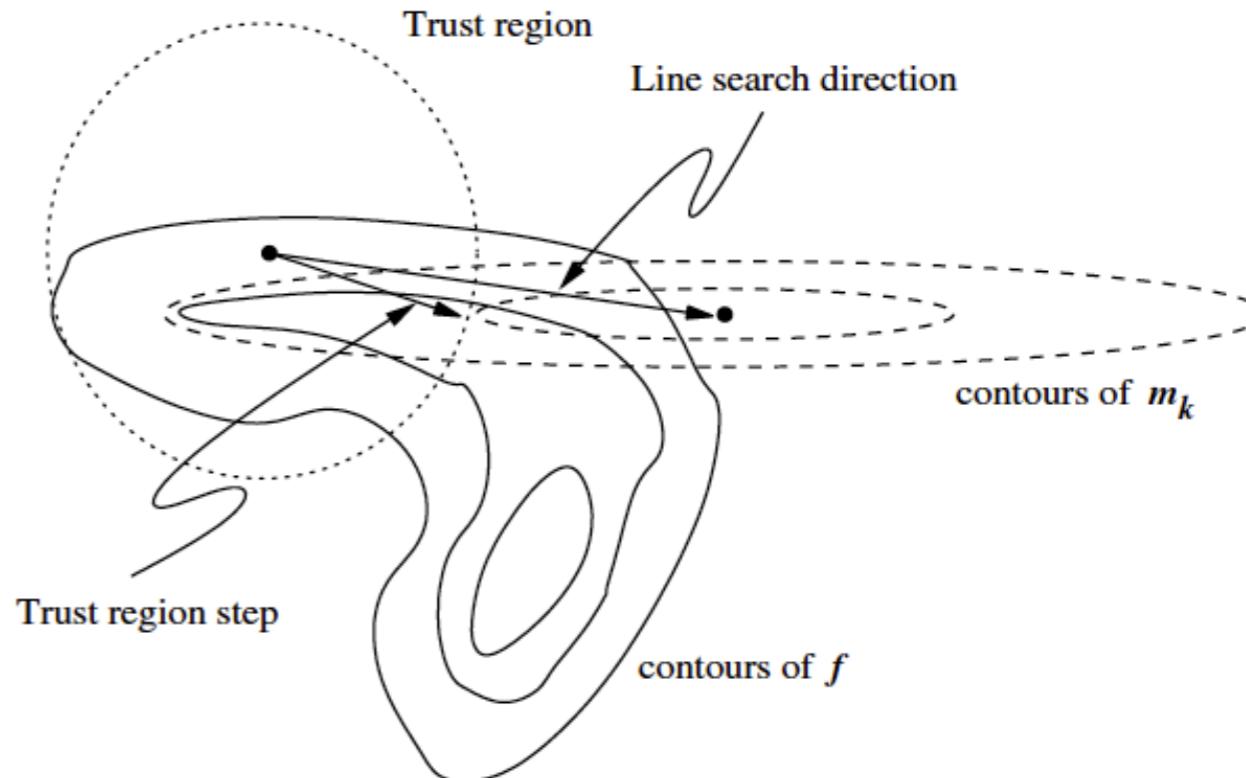
Approximate $d_\mu^{\pi_{new}}(s)$ by $d_\mu^{\pi_{old}}(s)$

(A2) Specify a trust region Δ

KL divergence between π_{old}, π_{new}

(A3) Find an approximate optimizer in Δ

Convexify the problem by 1st-order and 2nd-order approximation



(A1) Surrogate Function in TRPO

- **Recall:** Average performance difference lemma

$$\begin{aligned}\eta(\pi_{new}) &= \eta(\pi_{old}) + \mathbb{E}_{s \sim d_\mu^{\pi_{new}}, a \sim \pi_{new}(\cdot | s)} [A^{\pi_{old}}(s_t, a_t)] \\ &= \eta(\pi_{old}) + \sum_s d_\mu^{\pi_{new}}(s) \sum_a \pi_{new}(a | s) A^{\pi_{old}}(s, a)\end{aligned}$$

Approximate $d_\mu^{\pi_{new}}(s)$ by $d_\mu^{\pi_{old}}(s)$:

$$\eta(\pi_{new}) - \eta(\pi_{old}) \approx \sum_s d_\mu^{\pi_{old}}(s) \sum_a \pi_{new}(a | s) A^{\pi_{old}}(s, a)$$

- **Define:** Surrogate function $L_{\pi_{old}}(\pi_{new})$ in TRPO

$$L_{\pi_{old}}(\pi_{new}) = \eta(\pi_{old}) + \sum_s d_\mu^{\pi_{old}}(s) \sum_a \pi_{new}(a | s) A^{\pi_{old}}(s, a)$$

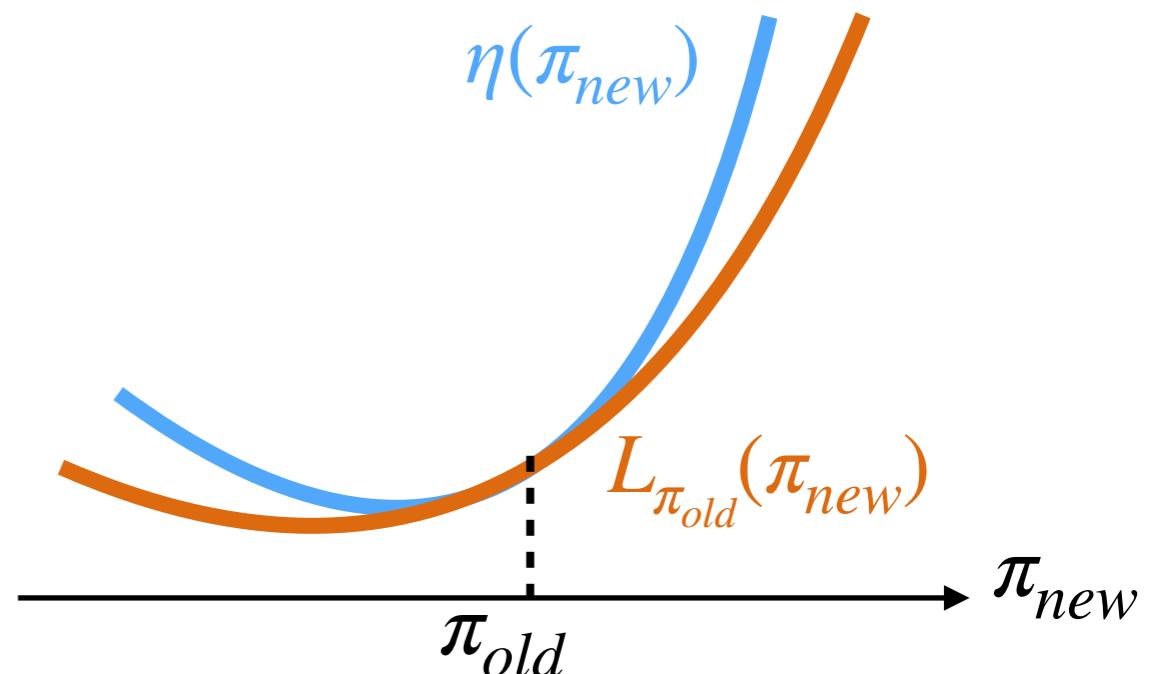
(A1) Why is $L_{\pi_{old}}(\pi_{new})$ a Good Surrogate Function?

$$L_{\pi_{old}}(\pi_{new}) = \eta(\pi_{old}) + \sum_s d_\mu^{\pi_{old}}(s) \sum_a \pi_{new}(a | s) A^{\pi_{old}}(s, a)$$

- $L_{\pi_{old}}(\pi_{new})$ satisfy two properties: $\pi_{old} \equiv \pi_{\theta_1}$, $\pi_{new} \equiv \pi_\theta$

1. $L_{\pi_{\theta_1}}(\pi_{\theta_1}) = \eta(\pi_{\theta_1})$
2. $\nabla_\theta L_{\pi_{\theta_1}}(\pi_\theta)|_{\theta=\theta_1} = \nabla_\theta \eta(\pi_\theta)|_{\theta=\theta_1}$

(HW2 problem)



- **Intuition:** If π_{old}, π_{new} are close, then improvement in $L_{\pi_{old}}(\pi_{new})$ implies improvement in $\eta(\pi_{new})$

Kullback-Leibler divergence Between Policies

- ▶ **Notation:** Kullback-Leibler (KL) divergence

$$D_{KL}(\pi(\cdot | s) \| \tilde{\pi}(\cdot | s)) := \sum_a \pi(a | s) \log\left(\frac{\pi(a | s)}{\tilde{\pi}(a | s)}\right)$$

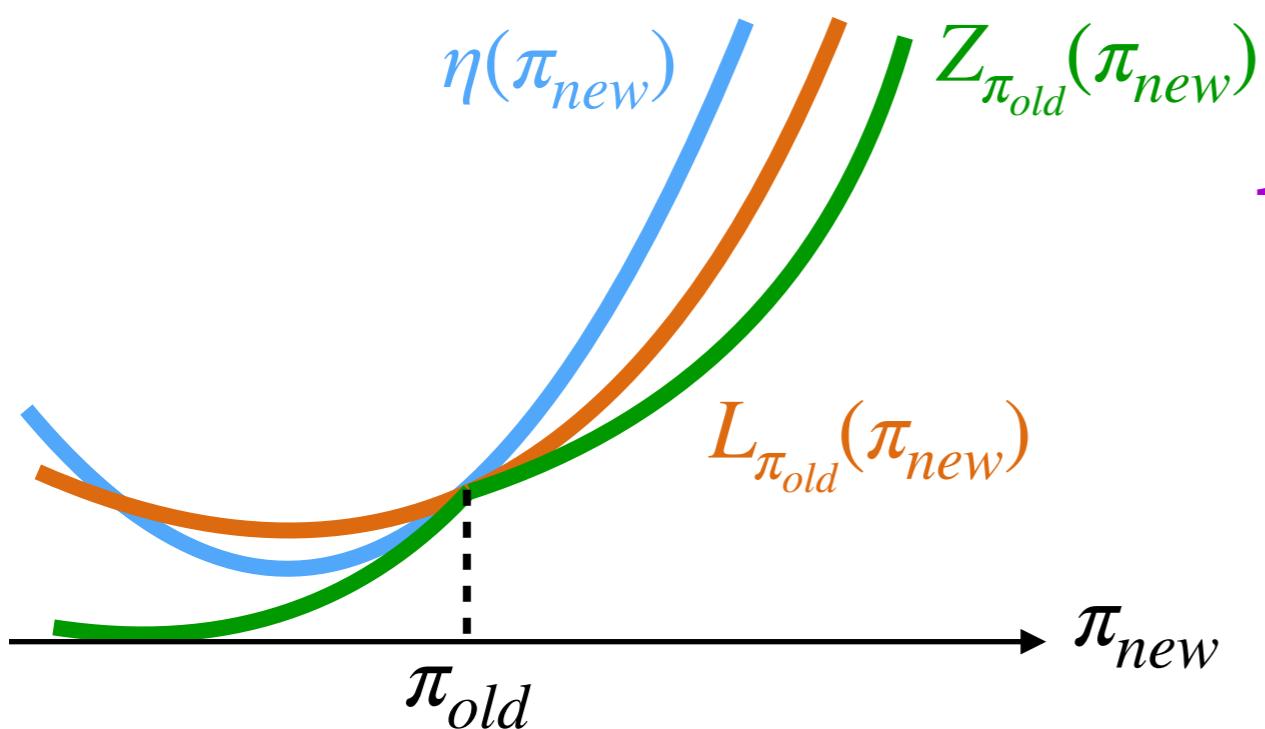
$$D_{KL}^{\max}(\pi \| \tilde{\pi}) := \max_s D_{KL}(\pi(\cdot | s) \| \tilde{\pi}(\cdot | s))$$

(A2) How to Specify a Trust Region Δ ?

- ▶ **Recall:** $L_{\pi_{old}}(\pi_{new})$ and $\eta(\pi_{new})$ are close if π_{old}, π_{new} are close
- ▶ **Policy Improvement Bound (PIB):** Let $\varepsilon := \max |A^{\pi_{old}}(s, a)|$

$$\eta(\pi_{new}) \geq L_{\pi_{old}}(\pi_{new}) - \underbrace{\frac{4\varepsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\pi_{old} \parallel \pi_{new})}_{=: Z_{\pi_{old}}(\pi_{new})}$$

- ▶ **Question:** How to specify a trust region?



1st attempt: $D_{KL}^{\max}(\pi_{old} \parallel \pi_{new}) \leq \delta$

Is $D_{KL}^{\max}(\pi_{old} \parallel \pi_{new})$ easy to evaluate?

(A2) How to Specify a Trust Region Δ ?

- **Policy Improvement Bound (PIB):** Let $\varepsilon := \max |A^{\pi_{old}}(s, a)|$

$$\eta(\pi_{new}) \geq L_{\pi_{old}}(\pi_{new}) - \underbrace{\frac{4\varepsilon\gamma}{(1-\gamma)^2} D_{KL}^{\max}(\pi_{old} \parallel \pi_{new})}_{=: Z_{\pi_{old}}(\pi_{new})}$$

2nd attempt: Use $\bar{D}_{KL}(\pi_{old} \parallel \pi_{new})$ instead of $D_{KL}^{\max}(\pi_{old} \parallel \pi_{new})$

$$\bar{D}_{KL}(\pi_{old} \parallel \pi_{new}) := \mathbb{E}_{s \sim \pi_{old}} [D_{KL}(\pi_{old}(\cdot \mid s) \parallel \pi_{new}(\cdot \mid s))]$$

Trust region in TRPO: $\bar{D}_{KL}(\pi_{old} \parallel \pi_{new}) \leq \delta$

Put Everything Together

- ▶ Trust-Region Policy Optimization (TRPO) Algorithm:

Step 1: Initialize θ_0

Step 2: For iteration $k = 0, 1, 2, \dots$

 Step 2-1: Collect trajectories by running the current policy π_{θ_k}

 Step 2-2: Obtain advantage $A^{\theta_k}(s, a)$ for the current policy π_{θ_k}

 Step 2-3: Update the policy by solving

$$\theta_{k+1} = \arg \max_{\theta} L_{\pi_{\theta_k}}(\pi_{\theta}) \quad (\equiv \arg \max_{\theta} \mathbb{E}_{s \sim d_{\mu}^{\pi_{\theta_k}}, a \sim \pi_{\theta_k}(\cdot | s)} \left[\frac{\pi_{\theta}(a | s)}{\pi_{\theta_k}(a | s)} A^{\theta_k}(s, a) \right])$$

subject to $\bar{D}_{KL}(\pi_{\theta_k} \| \pi_{\theta}) \leq \delta$

One remaining practical issue with TRPO...

$$\theta_{k+1} = \arg \max_{\theta} L_{\pi_{\theta_k}}(\pi_{\theta}) \quad (\equiv \arg \max_{\theta} \mathbb{E}_{s \sim d_{\mu}^{\pi_{\theta_k}}, a \sim \pi_{\theta_k}(\cdot | s)} \left[\frac{\pi_{\theta}(a | s)}{\pi_{\theta_k}(a | s)} A^{\theta_k}(s, a) \right])$$

subject to $\bar{D}_{KL}(\pi_{\theta_k} \| \pi_{\theta}) \leq \delta$

How to efficiently solve this constrained problem?

(A3) Find an approximate optimizer in Δ

$$\theta_{k+1} = \arg \max_{\theta} L_{\pi_{\theta_k}}(\pi_{\theta}) \quad (\equiv \arg \max_{\theta} \mathbb{E}_{s \sim d_{\mu}^{\pi_{\theta_k}}, a \sim \pi_{\theta_k}(\cdot | s)} \left[\frac{\pi_{\theta}(a | s)}{\pi_{\theta_k}(a | s)} A^{\theta_k}(s, a) \right])$$

subject to $\bar{D}_{KL}(\pi_{\theta_k} \| \pi_{\theta}) \leq \delta$

► **Idea:** Approximation

1. Linear approximation to the objective $L_{\theta_k}(\theta)$
2. Quadratic approximation to the KL constraint

► The problem under approximation:

Hessian of $\bar{D}_{KL}(\pi_{\theta_k} \| \pi_{\theta})$

$$\begin{aligned} & \text{Maximize} && (\theta - \theta_k)^T \nabla_{\theta} L_{\theta_k}(\theta) \Big|_{\theta=\theta_k} \\ & \text{subject to} && \frac{1}{2} (\theta - \theta_k)^T H(\theta - \theta_k) \leq \delta \end{aligned}$$

► **Question:** Why is this approximation helpful?

- The problem under approximation: Hessian of $\bar{D}_{KL}(\pi_{\theta_k} \parallel \pi_\theta)$

$$\begin{aligned} & \text{Maximize} && (\theta - \theta_k)^\top \nabla_{\theta} L_{\theta_k}(\theta) |_{\theta=\theta_k} \\ & \text{subject to} && \frac{1}{2} (\theta - \theta_k)^\top H(\theta - \theta_k) \leq \delta \end{aligned}$$

- The Hessian of $\bar{D}_{KL}(\pi_{\theta_k} \parallel \pi_\theta)$ is positive semi-definite
- The constraint is therefore convex (and can be easily analyzed)
- The solution is:

usually called “natural policy gradient”

$$\theta = \theta_k + \alpha H^{-1} \nabla_{\theta} L_{\theta_k}(\theta) |_{\theta=\theta_k} \quad (\text{HW2 problem})$$

Quick Summary: What is TRPO?

TRPO = TR Method on RL

With 3 key steps...

1. Approximate $d_\mu^{\pi_{new}}(s)$ by $d_\mu^{\pi_{old}}(s)$
2. Trust region by KL divergence between π_{old}, π_{new}
3. Simplify the problem by linear and quadratic approximation

Assignment for this lecture:

- ▶ Spend 30 minutes going through the idea of TRPO again
- ▶ Spend 30 minutes reading the code of TRPO
 - ▶ <https://github.com/ikostrikov/pytorch-trpo>

- ▶ Could you explain the purpose of each line?

- ▶ Could you find any part of the code that we have not discussed in this lecture?

```
51  def trpo_step(model, get_loss, get_kl, max_kl, damping):
52      loss = get_loss()
53      grads = torch.autograd.grad(loss, model.parameters())
54      loss_grad = torch.cat([grad.view(-1) for grad in grads]).data
55
56      def Fvp(v):
57          kl = get_kl()
58          kl = kl.mean()
59
60          grads = torch.autograd.grad(kl, model.parameters(), create_graph=True)
61          flat_grad_kl = torch.cat([grad.view(-1) for grad in grads])
62
63          kl_v = (flat_grad_kl * Variable(v)).sum()
64          grads = torch.autograd.grad(kl_v, model.parameters())
65          flat_grad_grad_kl = torch.cat([grad.contiguous().view(-1) for grad in grads]).data
66
67          return flat_grad_grad_kl + v * damping
68
69      stepdir = conjugate_gradients(Fvp, -loss_grad, 10)
70
71      shs = 0.5 * (stepdir * Fvp(stepdir)).sum(0, keepdim=True)
72
73      lm = torch.sqrt(shs / max_kl)
74      fullstep = stepdir / lm[0]
75
76      neggdotstepdir = (-loss_grad * stepdir).sum(0, keepdim=True)
77      print(("lagrange multiplier:", lm[0], "grad_norm:", loss_grad.norm()))
78
79      prev_params = get_flat_params_from(model)
80      success, new_params = linesearch(model, get_loss, prev_params, fullstep,
81                                      neggdotstepdir / lm[0])
82      set_flat_params_to(model, new_params)
83
84      return loss
```

We will discuss this next time!