Policy Gradient Descent

Lucas Janson and Sham Kakade

CS/Stat 184: Introduction to Reinforcement Learning

Fall 2023

Today

- Recap ++
- Gradient Descent
- Policy Gradient
  - Likelihood ratio method
  - REINFORCE
  - Estimation

Recap++

Q-Value Dynamic Programming Algorithm:

Recall from HW1, Problem 2, the Bellman equations for $Q^\star$:

$$Q_h^\star(s,a) = r(s,a) + \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[\max_{a'} Q_{h+1}^\star(s',a')\right]$$

$V(s) = \max_a \{ r(s,a) + \mathbb{E}[\ ] \}$

Analogous Q-value DP, with same notational change as previous slide: h as argument

1. Initialization: $Q(s,a,H) = 0 \quad \forall s,a$
2. Solve (via dynamic programming):

$$Q(s,a,h) = r(s,a) + \mathbb{E}_{s' \sim P(s,a)}\left[\max_{a' \in A} Q(s',a',h+1)\right] \qquad \forall s,a,h$$

3. Return:

$$\pi_h(s) = \arg\max_a \left\{ Q(s,a,h) \right\}$$

What if we can't just evaluate the expectations?

If S and/or A are very large, computing expectations could be very expensive

We may not have a way to directly compute those expectations, but instead only have access to a simulator (or the real world), where we can collect data

Suppose:

This is now full RL!!

- We have N trajectories $\tau_1, \ldots \tau_N \sim \rho_{\pi_{data}}$

  Each trajectory is of the form $\tau_i = \{s_0^i, a_0^i, \ldots s_{H-1}^i, a_{H-1}^i, s_H^i\}$
- π_{data} is often referred to as our data collection policy.

Want: $Q(s, a, h) \approx r(s, a) + \mathbb{E}_{s' \sim P(s,a)} \left[\max_{a' \in A} Q(s', a', h+1) \right] \quad \forall s, a, h$

Since we're trying to approximate conditional expectations, seems like it kind of fits into supervised learning—can we use an approach like that? Yes!

Connection to Supervised Learning

$$Q(s,a,h) \approx r(s,a) + \mathbb{E}_{s' \sim P(s,a)} \left[\max_{a' \in A} Q(s', a', h+1) \right] \quad \forall s, a, h$$

What are the y and x?

Note that the RHS can also be written as

$$\mathbb{E}\left[r(s_h, a_h) + \max_{a'} Q(s_{h+1}, a', h+1) \,\middle|\, s_h, a_h, h \right]$$

This suggests that $y = r(s_h, a_h) + \max_{a'} Q(s_{h+1}, a', h+1)$ and $x = (s_h, a_h, h)$

Then we'd be happy if we found a

$$Q(s_h, a_h, h) = f(x) = \mathbb{E}[y \mid x] = \mathbb{E}\left[r(s_h, a_h) + \max_{a'} Q(s_{h+1}, a', h+1) \,\middle|\, s_h, a_h, h \right]$$

Connection to Supervised Learning (cont'd)

We can convert our data $\tau_1, \ldots \tau_N \sim \rho_{\pi_{data}}$, into (y, x) pairs; how many? NH

BUT, to compute each y, we need to already know Q!

Setting that aside for the moment, to fit supervised learning, we'd minimize a least-squares objective function: $\hat{f}(x) = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{NH} (y_i - f(x_i))^2$

Then if we have enough data, choose a good $\mathcal{F}$, and optimize well,

$$Q(s_h, a_h, h) := \hat{f}(x) \approx \mathbb{E}[y \,|\, x] = \mathbb{E}\left[r(s_h, a_h) + \max_{a'} Q(s_{h+1}, a', h+1) \,\middle|\, s_h, a_h, h \right]$$

Fitted (Q-)Value Iteration

To address the circularity problem of not knowing Q for computing the y, we have an algorithmic tool… what is it?
Hint: we used it for another VI algorithm before…

Fixed point iteration! Initialize, then at each step, pretend Q is known by plugging in the previous time step's Q to compute the y's, and then use that to get next Q

Input: **offline dataset** $\tau_1, \ldots \tau_N \sim \rho_{\pi_{data}}$

1. Initialize fitted Q function at f_0
2. For $k = 1, \ldots, K$:

fixed

$$f_k = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{N} \sum_{h=1}^{H-1} \left(f(s_h^i, a_h^i, h) - \left(r(s_h^i, a_h^i) + \max_a f_{k-1}(s_{h+1}^i, a, h+1) \right) \right)^2$$

3. With f_K as an estimate of $Q^\star$, return $\pi_h(s) = \arg\max_a \left\{ f^K(s, a, h) \right\}$

Q-Learning is an online version, i.e., draw new trajectories at each k based on f_k as Q-function

Bonus: Q-learning

(Tabular) Q-Learning

(Tabular) Q-Learning

- Init: $Q_h(s, a)$

(Tabular) Q-Learning

- Init: $Q_h(s, a)$
- For $k = 1,2,\ldots K$ episodes

(Tabular) Q-Learning

- Init: $Q_h(s, a)$
- For $k = 1,2,\ldots K$ episodes
  - Within each episode, for $h = 0,1,\ldots H-1$

(Tabular) Q-Learning

- Init: $Q_h(s, a)$
- For $k = 1,2,\ldots K$ episodes
  - Within each episode, for $h = 0,1,\ldots H-1$
    - Act: choose actions however you like!
      (but try to maintain exploration)

(Tabular) Q-Learning

- Init: $Q_h(s, a)$
- For $k = 1,2,\ldots K$ episodes
  - Within each episode, for $h = 0,1,\ldots H-1$
    - Act: choose actions however you like!
      (but try to maintain exploration)
    - Update:

$$Q_h(s_h, a_h) \leftarrow Q_h(s_h, a_h) + \eta^{K}\left(r_h + \max_a Q_{h+1}(s_{h+1}, a) - Q_h(s_h, a_h)\right)$$

(Tabular) Q-Learning

- Init: $Q_h(s, a)$
- For $k = 1,2,\ldots K$ episodes
  - Within each episode, for $h = 0,1,\ldots H - 1$
    - Act: choose actions however you like!
      (but try to maintain exploration)
    - Update:

$$Q_h(s_h, a_h) \leftarrow Q_h(s_h, a_h) + \eta \left(r_h + \max_a Q_{h+1}(s_{h+1}, a) - Q_h(s_h, a_h) \right)$$

- Return $Q_h(s, a)$

(Tabular) Q-Learning

- Init: $Q_h(s, a)$
- For $k = 1,2,\ldots K$ episodes
  - Within each episode, for $h = 0,1,\ldots H-1$
    - Act: choose actions however you like!
      (but try to maintain exploration)
    - Update:

$$Q_h(s_h, a_h) \leftarrow Q_h(s_h, a_h) + \eta \left(r_h + \max_a Q_{h+1}(s_{h+1}, a) - Q_h(s_h, a_h) \right)$$

- Return $Q_h(s, a)$

  - Q-learning is an "off-policy" algorithm.

(Tabular) Q-Learning

- Init: $Q_h(s, a)$
- For $k = 1,2,\ldots K$ episodes
  - Within each episode, for $h = 0,1,\ldots H - 1$
    - Act: choose actions however you like!
      (but try to maintain exploration)
    - Update:

$$Q_h(s_h, a_h) \leftarrow Q_h(s_h, a_h) + \eta \left(r_h + \max_a Q_{h+1}(s_{h+1}, a) - Q_h(s_h, a_h) \right)$$

- Return $Q_h(s, a)$

- Q-learning is an "off-policy" algorithm.
- Guarantee: Assuming states, actions visited infinitely often (which can be guaranteed with the action policy), $Q_h \to Q_h^\star$.

Q-Learning with Function Approximation (extra material: read later if interested)

- Init: $Q_h(s, a)$
- For $k = 1,2,\ldots K$ episodes
  - Within each episode, for $h = 0,1,\ldots H-1$
    - Act: choose actions however you like!
      (but try to maintain exploration)
    - Update:
      $$\theta \leftarrow \theta - \eta\left(f_\theta(s_h, a_h, h) - r_h - \gamma \max_a f_\theta(s_{h+1}, a, h+1)\right)\nabla f_\theta(s_h, a_h, h)$$
- Return $Q_h(s, a)$

- How to understand this expression?
  Consider doing a small step of SGD on the fitted-Q objective function.

Recall: Policy Iteration (PI)

- Initialization: choose a policy $\pi^0 : S \mapsto A$
- For $k = 0,1,\ldots$
  1. **Policy Evaluation**: Solve (via dynamic programming):

     $$Q^{\pi^k}(s,a,h) = r(s,a) + \mathbb{E}_{s' \sim P(\cdot|s,a)}\left[Q^{\pi^k}(s', \pi^k(s), h+1)\right] \quad \forall s,a,h$$

  2. **Policy Improvement**: set $\pi_h^{k+1}(s) := \arg\max_a Q^{\pi^k}(s,a,h)$

Again: what if we're in full RL setting where we can't just evaluate expectations?

This breaks the Policy Evaluation step, so can we do a fitted version?

Yes! RHS can be written as $\mathbb{E}\left[r(s_h, a_h) + Q^{\pi^k}(s_{h+1}, \pi^k(s_h), h+1) \,\middle|\, s_h, a_h, h\right]$

Fitted Policy Iteration:

- Initialization: choose a policy $\pi^0 : S \mapsto A$ and a sample size N
- For $k = 0,1,\ldots$
  1. **Fitted Policy Evaluation**: Using N sampled trajectories $\tau_1, \ldots \tau_N \sim \rho_{\pi^k}$, obtain approximation $\hat{Q}^{\pi^k} \approx Q^{\pi^k}$
  2. **Policy Improvement**: set $\pi_h^{k+1}(s) := \arg\max_a \hat{Q}^{\pi^k}(s, a, h)$

Direct Policy Evaluation option

Using the definition of the Q function, can do a non-iterative fitted policy evaluation

$$Q^\pi(s,a,h) = \mathbb{E}\left[\sum_{t=h}^{H-1} r(s_t,a_t) \middle| s_h, a_h, h\right]$$

Input: policy π, **dataset** $\tau_1, \ldots \tau_N \sim \rho_\pi$

Return:

$$\hat{Q}^\pi = \arg\min_{f \in \mathscr{F}} \sum_{i=1}^{N} \sum_{h=1}^{H-1} \left(f(s_h^i, a_h^i, h) - \sum_{t=h}^{H-1} r(s_t^i, a_t^i) \right)^2$$

Another Fitted Policy Evaluation

We can also use **fixed point iteration**

Input: policy π, **dataset** $\tau_1, \ldots \tau_N \sim \rho_\pi$

1. Initialize fitted Q^π function at f_0
2. For $k = 1,\ldots,K$:

$$f_k = \arg\min_{f \in \mathscr{F}} \sum_{i=1}^{N} \sum_{h=1}^{H-1} \left(f(s_h^i, a_h^i, h) - \left(r(s_h^i, a_h^i) + f_{k-1}(s_{h+1}^i, \pi(s_h^i), h+1) \right) \right)^2$$

3. Return the function f_K as an estimate of Q^π

Bonus: TD(0)

(see posted slides)

Today:

Today

- Recap
- Gradient Descent
- Policy Gradient
  - Likelihood ratio method
  - REINFORCE
  - Estimation

Gradient Descent (GD) and Stochastic Gradient Descent (SGD)

(we really do *ascent* in RL, so we should say GA and SGA)

Gradient Descent (GD) and Stochastic Gradient Descent (SGD)

(we really do *ascent* in RL, so we should say GA and SGA)

- Given an objective function

$$J(\theta) : \mathbb{R}^d \mapsto \mathbb{R}, \quad \text{(e.g., } J(\theta) = \mathbb{E}_{x,y}(f_\theta(x) - y)^2),$$

our objective is: $\max_\theta J(\theta)$

Gradient Descent (GD) and Stochastic Gradient Descent (SGD)

(we really do *ascent* in RL, so we should say GA and SGA)

- Given an objective function

  $$J(\theta) : \mathbb{R}^d \mapsto \mathbb{R}, \quad \text{(e.g., } J(\theta) = \mathbb{E}_{x,y}(f_\theta(x) - y)^2\text{)},$$

  our objective is: $\max_\theta J(\theta)$

- Gradient Descent is an iterative approach, to decrease the objective function as follows:
  - Initialize θ_0, for k = 0, … :

    $$\theta^{k+1} = \theta^k - \eta \nabla J(\theta^k)$$

Gradient Descent (GD) and Stochastic Gradient Descent (SGD)

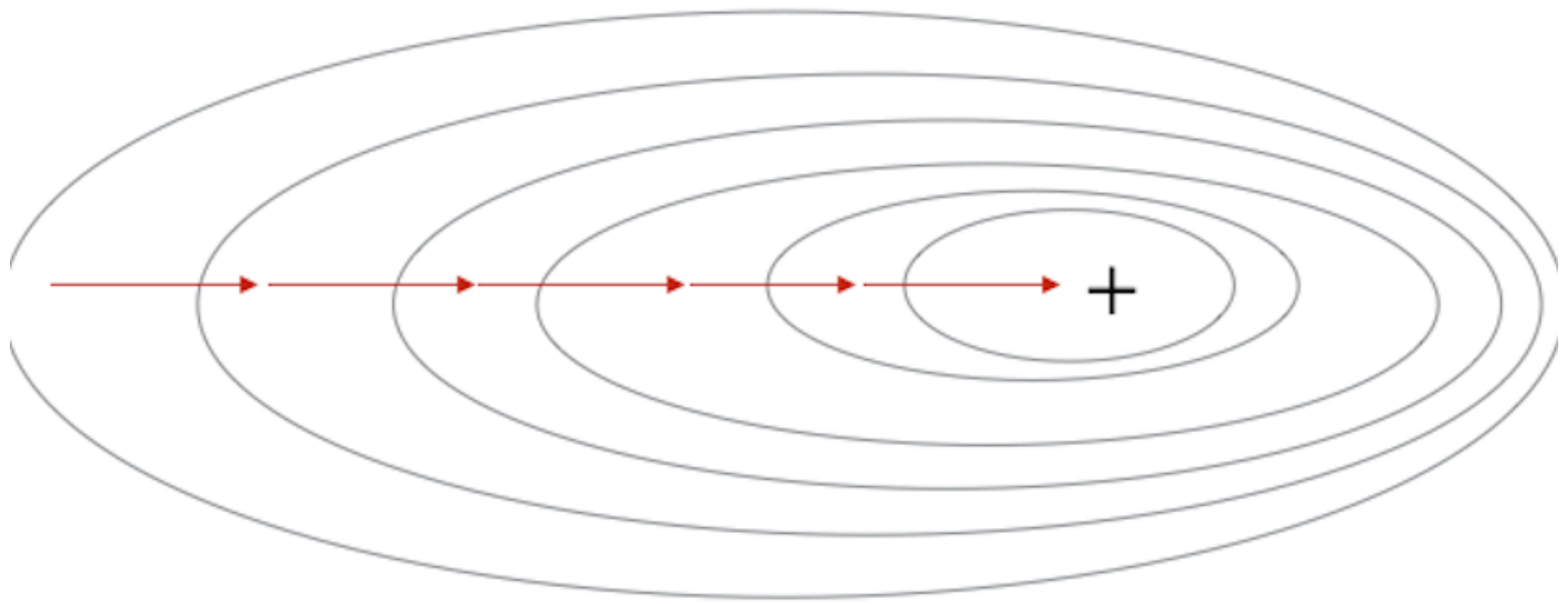
(we really do *ascent* in RL, so we should say GA and SGA)

- Given an objective function

  $$J(\theta) : \mathbb{R}^d \mapsto \mathbb{R}, \quad \text{(e.g., } J(\theta) = \mathbb{E}_{x,y}(f_\theta(x) - y)^2\text{)},$$

  our objective is: $\max_\theta J(\theta)$

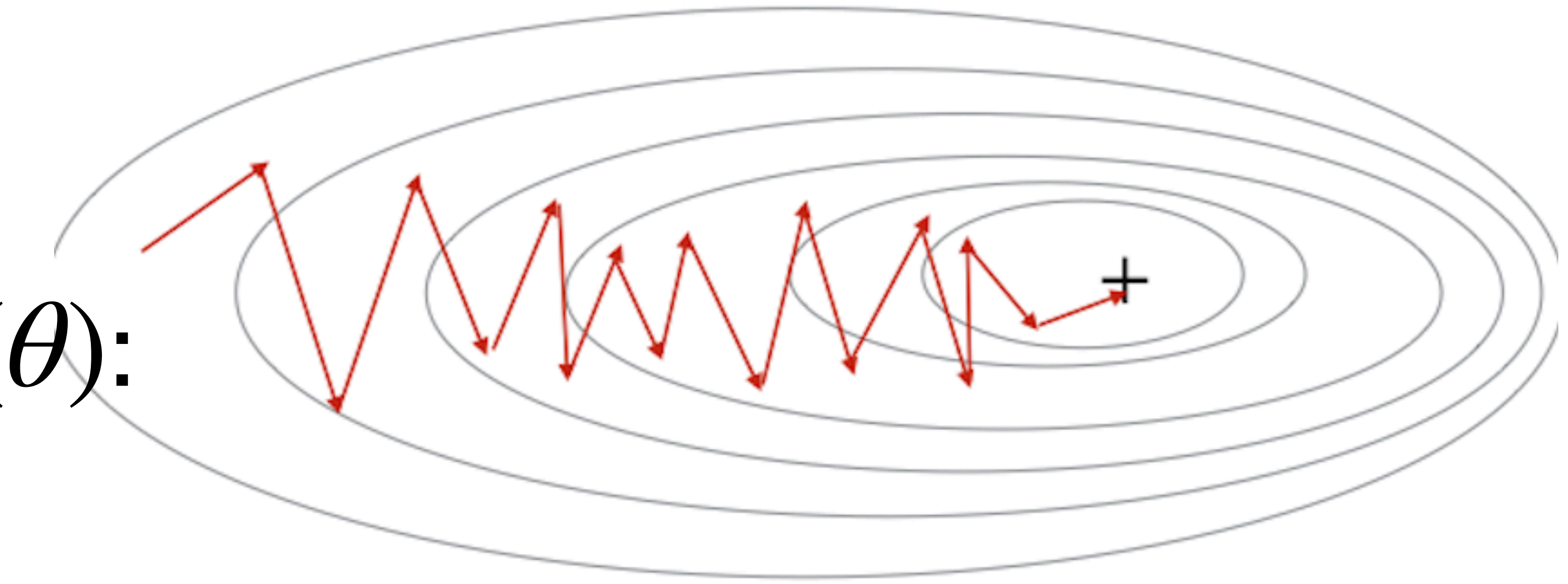
Gradient Descent



- Gradient Descent is an iterative approach, to decrease the objective function as follows:
  - Initialize θ_0, for k = 0, … :

    $$\theta^{k+1} = \theta^k - \eta \nabla J(\theta^k)$$

Stochastic Gradient Descent



- Stochastic Gradient Descent uses (unbiased) estimates of $\nabla J(\theta)$:
  - Initialize θ_0, for k = 0, … :

    $$\theta^{k+1} = \theta^k - \eta^k g^k, \quad \text{where } \mathbb{E}[g^k] = \nabla_\theta J(\theta^k)$$

Example of GD

Example of GD

- Given an objective function

$$J(\theta) : \mathbb{R} \mapsto \mathbb{R}, \quad J(\theta) = \frac{1}{2}(\theta - c)^2,$$

our objective is: $\max_{\theta} J(\theta)$

Example of GD

- Given an objective function

$$J(\theta) : \mathbb{R} \mapsto \mathbb{R}, \quad J(\theta) = \frac{1}{2}(\theta - c)^2,$$

min

our objective is: $\cancel{\max}_{\theta} J(\theta)$

- We have $\nabla J(\theta) = \theta - c$, so GD is:
  - Initialize $\theta^0 = 0$,
  - for t = 0, … :

$$\theta^{k+1} = \theta^k - \eta(\theta^k - c)$$

Example of GD

- Given an objective function

$$J(\theta) : \mathbb{R} \mapsto \mathbb{R}, \quad J(\theta) = \frac{1}{2}(\theta - c)^2,$$

  our objective is: $\max_{\theta} J(\theta)$

- We have $\nabla J(\theta) = \theta - c$, so GD is:
  - Initialize $\theta^0 = 0$,
  - for t = 0, … :

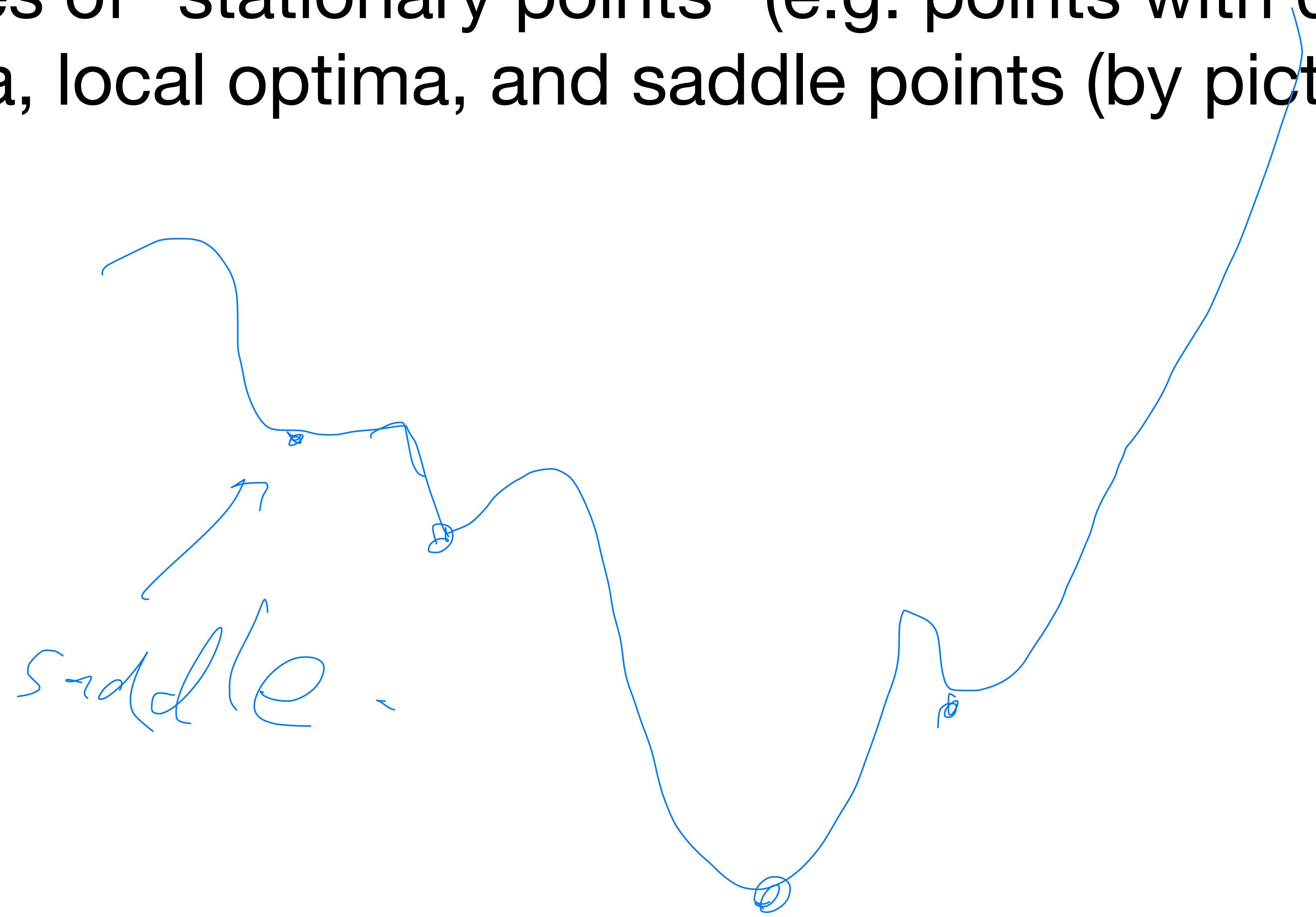
$$\theta^{k+1} = \theta^k - \eta(\theta - c)$$

- Note with $\eta = \cancel{0}$ 1, we find the optima, $\theta^\star = c$, in one step.

Brief overview of GD/SGD:

Brief overview of GD/SGD:

- Different types of “stationary points” (e.g. points with 0 gradients): global optima, local optima, and saddle points (by picture)



Brief overview of GD/SGD:

- Different types of “stationary points” (e.g. points with 0 gradients): global optima, local optima, and saddle points (by picture)

- For convex functions (with certain regularity conditions, such as “smoothness”),
  - GD (with an appropriate constant learning rate) converges to the global optima.
  - SGD (with an appropriately decaying learning rate) converges to the global optima. (lower variance is better for SGD)

Brief overview of GD/SGD:

- Different types of "stationary points" (e.g. points with 0 gradients): global optima, local optima, and saddle points (by picture)

- For convex functions (with certain regularity conditions, such as "smoothness"),
  - GD (with an appropriate constant learning rate) converges to the global optima.
  - SGD (with an appropriately decaying learning rate) converges to the global optima. (lower variance is better for SGD)

- For non-convex functions, we could hope to find a local minima.

Brief overview of GD/SGD:

- Different types of "stationary points" (e.g. points with 0 gradients):
  global optima, local optima, and saddle points (by picture)

- For convex functions (with certain regularity conditions, such as "smoothness"),
  - GD (with an appropriate constant learning rate) converges to the global optima.
  - SGD (with an appropriately decaying learning rate) converges to the global optima.
    (lower variance is better for SGD)

- For non-convex functions, we could hope to find a local minima.

- What we can prove (under some regularity conditions) is a little weaker:
  Both GD (with some constant learning rate) and SGD (with some decaying learning rate) converge to a stationary point, i.e.

  As $k \to \infty, \; \nabla J(\theta^k) \to 0$

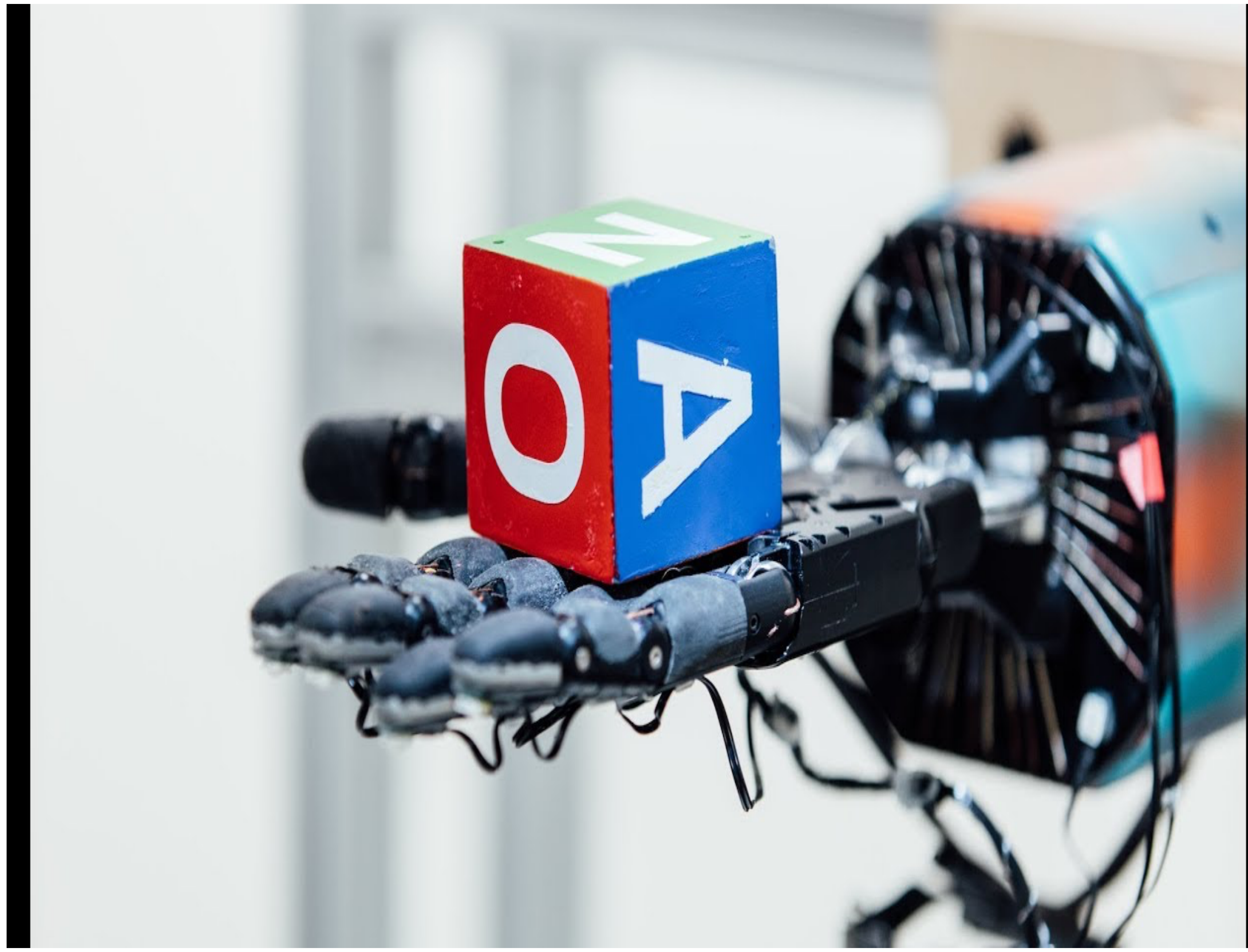
Today

- Recap
- Gradient Descent
- Policy Gradient
    - Likelihood ratio method
    - REINFORCE
    - Estimation

Policy Optimization

[AlphaZero, Silver et.al, 17]

[OpenAI Five, 18]



[OpenAI,19]

The Learning Setting:

We don't know the MDP, but we can obtain trajectories.

The Finite Horizon, Learning Setting. We can obtain trajectories as follows:

- We start at $s_0 \sim \mu$.
- We can act for H steps and observe the trajectory $\tau = \{s_0, a_0, s_1, a_1, \ldots, s_{H-1}, a_{H-1}\}$

Note that with a simulator, we can sample trajectories as specified in the above.

Optimization Objective

- Consider a parameterize class of policies:

$$\{\pi_\theta(a \mid s) \mid \theta \in \mathbb{R}^d\}$$

$L(\theta) = \sum_i (y_i - f_\theta(x_i))^2$

(why do we make it stochastic?)

- Objective $\max_\theta J(\theta)$, where

$$J(\theta) := E_{s_0 \sim \mu_0}\left[V^{\pi_\theta}(s_0)\right] = E_{\tau \sim \rho_{\pi_\theta}}\left[\sum_{h=0}^{H-1} r(s_h, a_h) \middle| \pi_\theta\right]$$

Ascent.

- Policy Gradient ~~Descent~~:

$$\theta^{k+1} = \theta_k + \eta \nabla J(\theta^k)$$

Optimization Objective

- Consider a parameterize class of policies:

$$\{\pi_\theta(a \mid s) \mid \theta \in \mathbb{R}^d\}$$

(why do we make it stochastic?)

- Objective $\max_\theta J(\theta)$, where

$$J(\theta) := E_{s_0 \sim \mu_0}\left[V^{\pi_\theta}(s_0)\right] = E_{\tau \sim \rho_{\pi_\theta}}\left[\sum_{h=0}^{H-1} r(s_h, a_h) \middle| \pi_\theta\right]$$

- Policy Gradient Descent:

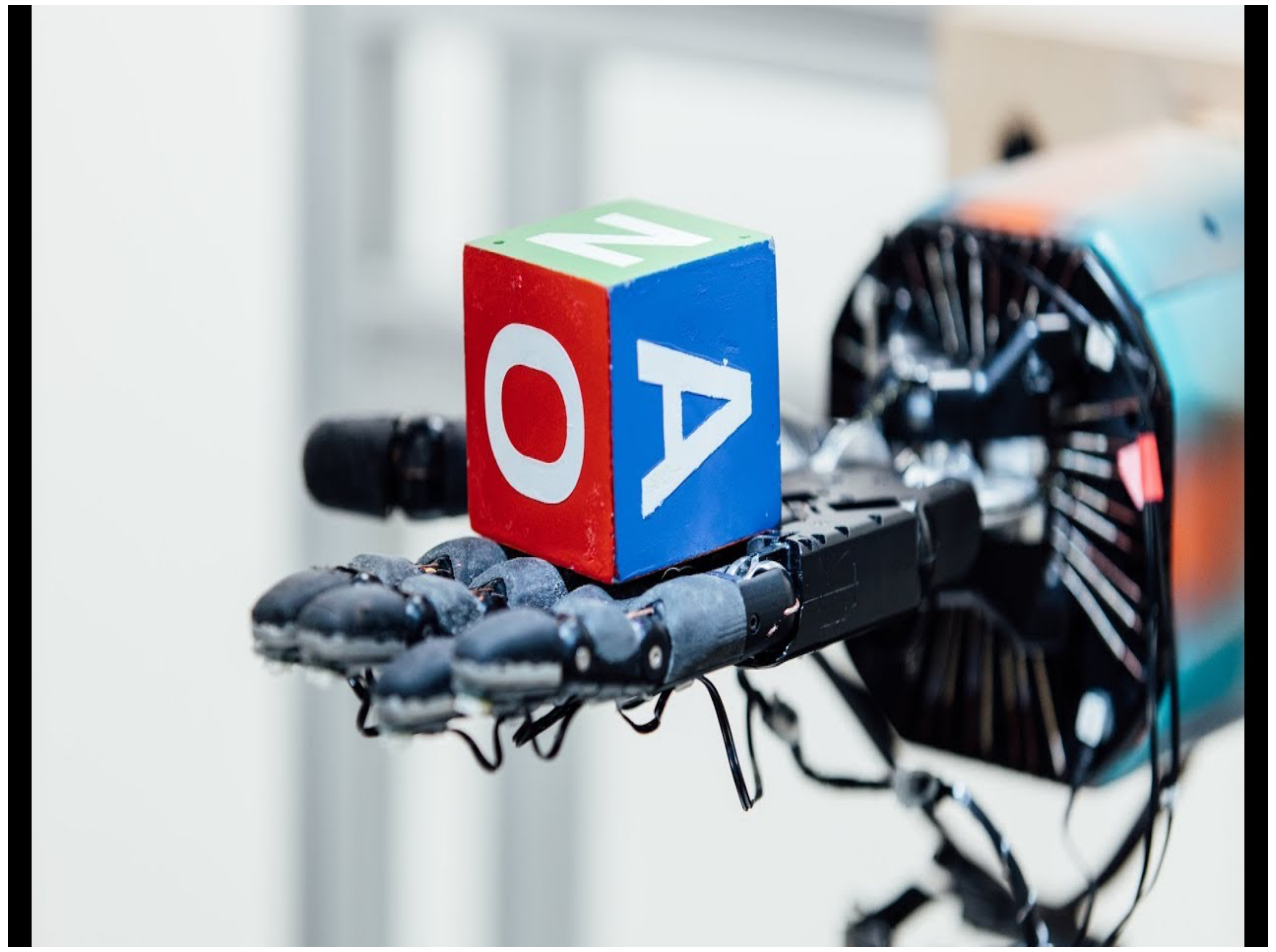
$$\theta^{k+1} = \theta_k + \eta \nabla J(\theta^k)$$

Main question for today's lecture:
how to compute the gradient?

What are parameterized policies?

[AlphaZero, Silver

[OpenAI Five,



[OpenAI,

A state:

- Tabular case: an index in $[|S|] = \{1,\ldots|S|\}$
- Real world: a list/array of the relevant info about the world that makes the process Markovian.
  - e.g. sometimes make a feature vector $\phi(s, a, h) \in \mathbb{R}^d$ which we believe is a "good representation" of the world
  - we sometimes append history info into the current state

Example Policy Parameterizations

Recall that we consider parameterized policy $\pi_\theta(\cdot \mid s) \in \Delta(A), \forall s$

1. Softmax linear Policy

Feature vector $\phi(s, a, h) \in \mathbb{R}^d$, and parameter $\theta \in \mathbb{R}^d$

$$\pi_\theta(a \mid s, h) = \frac{\exp(\theta^\top \phi(s, a, h))}{\sum_{a'} \exp(\theta^\top \phi(s, a', h))}$$

or $\quad \propto \exp(\theta_a \cdot \phi(s, h)) \,/\, Z$

Example Policy Parameterizations

Recall that we consider parameterized policy $\pi_\theta(\cdot \mid s) \in \Delta(A), \forall s$

1. Softmax linear Policy

Feature vector $\phi(s, a, h) \in \mathbb{R}^d$, and parameter $\theta \in \mathbb{R}^d$

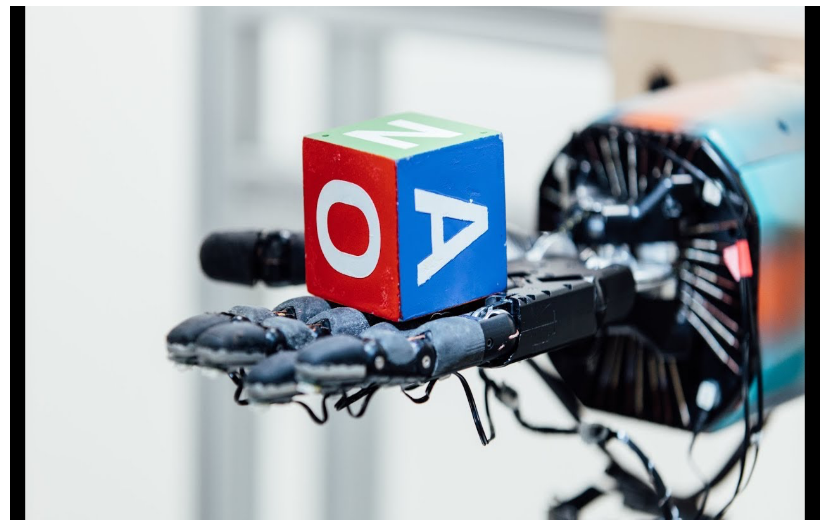
$$\pi_\theta(a \mid s, h) = \frac{\exp(\theta^\top \phi(s, a, h))}{\sum_{a'} \exp(\theta^\top \phi(s, a', h))}$$

2. Neural Policy:

Neural network
$f_\theta : S \times A \times [H] \mapsto \mathbb{R}$

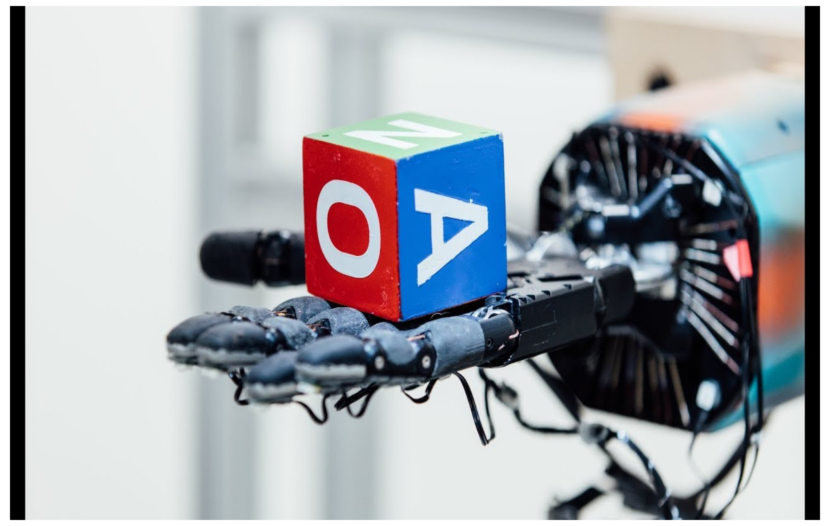
$$\pi_\theta(a \mid s, h) = \frac{\exp(f_\theta(s, a, h))}{\sum_{a'} \exp(f_\theta(s, a', h))}$$

Example Policy Parameterization for "Controls"



Suppose $a \in R^k$, as it might be for a control problem.

Example Policy Parameterization for “Controls”



Suppose $a \in R^k$, as it might be for a control problem.

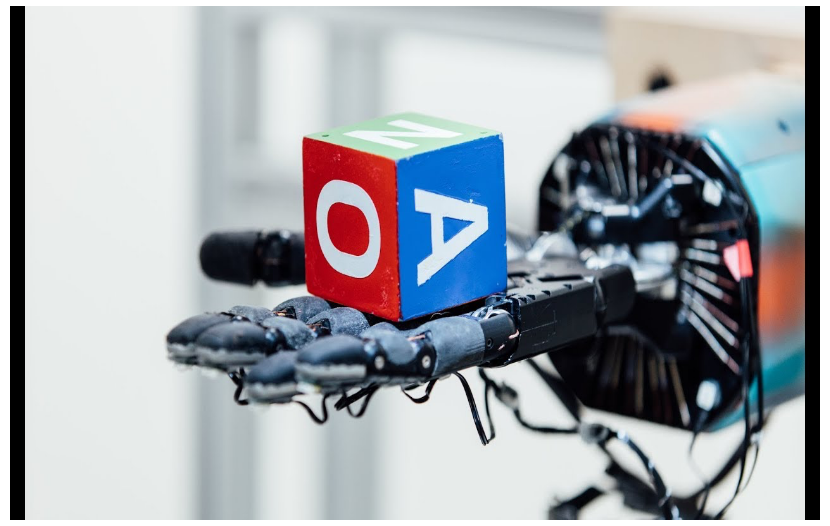
3. Gaussian + Linear Model

- Feature vector: $\phi(s,h) \in \mathbb{R}^d$,
- Parameters: $\theta \in \mathbb{R}^{k \times d}$,
  (and maybe $\sigma \in R^+$)

- Policy: sample action from a (multivariate) Normal with mean $\theta \cdot \phi(s,h)$ and variance $\sigma^2 I$, i.e.

$$\pi_{\theta,\sigma}(\cdot \mid s,h) = \mathcal{N}\left(\theta \cdot \phi(s,h), \sigma^2 I\right)$$

Example Policy Parameterization for "Controls"



Suppose $a \in R^k$, as it might be for a control problem.

3. Gaussian + Linear Model

- Feature vector: $\phi(s, h) \in \mathbb{R}^d$,
- Parameters: $\theta \in \mathbb{R}^{k \times d}$,
  (and maybe $\sigma \in R^+$)

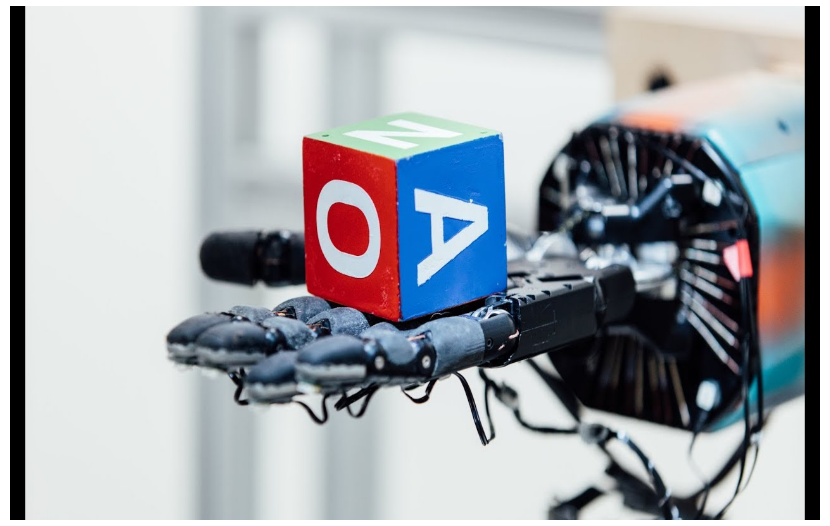
- Policy: sample action from a (multivariate) Normal with mean $\theta \cdot \phi(s, h)$ and variance $\sigma^2 I$, i.e.

$$\pi_{\theta,\sigma}(\cdot \mid s, h) = \mathcal{N}\left(\theta \cdot \phi(s, h), \sigma^2 I\right)$$

- Sampling:

$$a = \theta \cdot \phi(s, h) + \eta, \text{ where } \eta \sim \mathcal{N}(0, \sigma^2 I)$$

Example Policy Parameterization for “Controls”



Suppose $a \in R^k$, as it might be for a control problem.

3. Gaussian + Linear Model

- Feature vector: $\phi(s, h) \in \mathbb{R}^d$,
- Parameters: $\theta \in \mathbb{R}^{k \times d}$,
  (and maybe $\sigma \in R^+$)

- Policy: sample action from a (multivariate) Normal with mean $\theta \cdot \phi(s, h)$ and variance $\sigma^2 I$, i.e.

$$\pi_{\theta,\sigma}(\cdot \mid s, h) = \mathcal{N}\left(\theta \cdot \phi(s, h), \sigma^2 I\right)$$

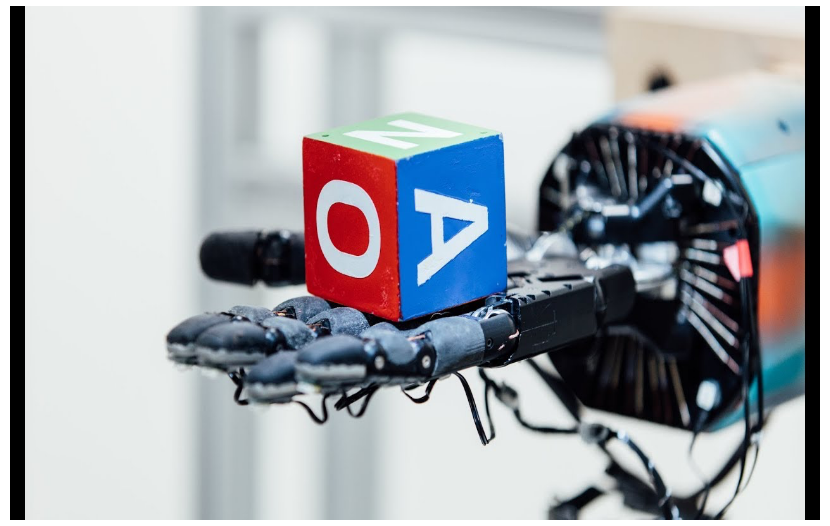
- Sampling:

$$a = \theta \cdot \phi(s, h) + \eta, \text{ where } \eta \sim \mathcal{N}(0, \sigma^2 I)$$

- Implicitly, this is the same functional form as Case 2 (the neural policy case).

$$f_{\theta,\sigma}(s, a) = \frac{\|a - \theta \cdot \phi(s, h)\|^2}{2\sigma^2 k}$$

Neural Policy Parameterization for "Controls"



Suppose $a \in R^k$, as it might be for a control problem.

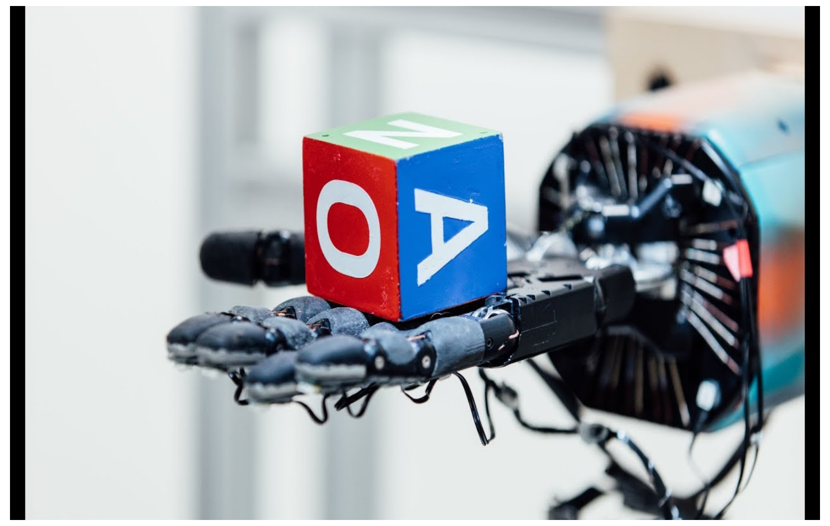
3. Gaussian + Linear Model

- Feature vector: $\phi(s, h) \in \mathbb{R}^d$,
- Parameters: $\theta \in \mathbb{R}^{k \times d}$,
  (and maybe $\sigma \in R^+$)

- Policy: sample action from a (multivariate) Normal with mean $\theta \cdot \phi(s, h)$ and variance $\sigma^2 I$, i.e.

$$\pi_{\theta,\sigma}(\cdot \mid s, h) = \mathcal{N}\left(\theta \cdot \phi(s, h), \sigma^2 I\right)$$

Neural Policy Parameterization for “Controls”



Suppose $a \in R^k$, as it might be for a control problem.

3. Gaussian + Linear Model

- Feature vector: $\phi(s, h) \in \mathbb{R}^d$,
- Parameters: $\theta \in \mathbb{R}^{k \times d}$,
  (and maybe $\sigma \in R^+$)

- Policy: sample action from a (multivariate) Normal with mean $\theta \cdot \phi(s, h)$ and variance $\sigma^2 I$, i.e.
$$\pi_{\theta,\sigma}(\cdot \mid s, h) = \mathcal{N}\left(\theta \cdot \phi(s, h), \sigma^2 I\right)$$

4. Gaussian + Neural Model

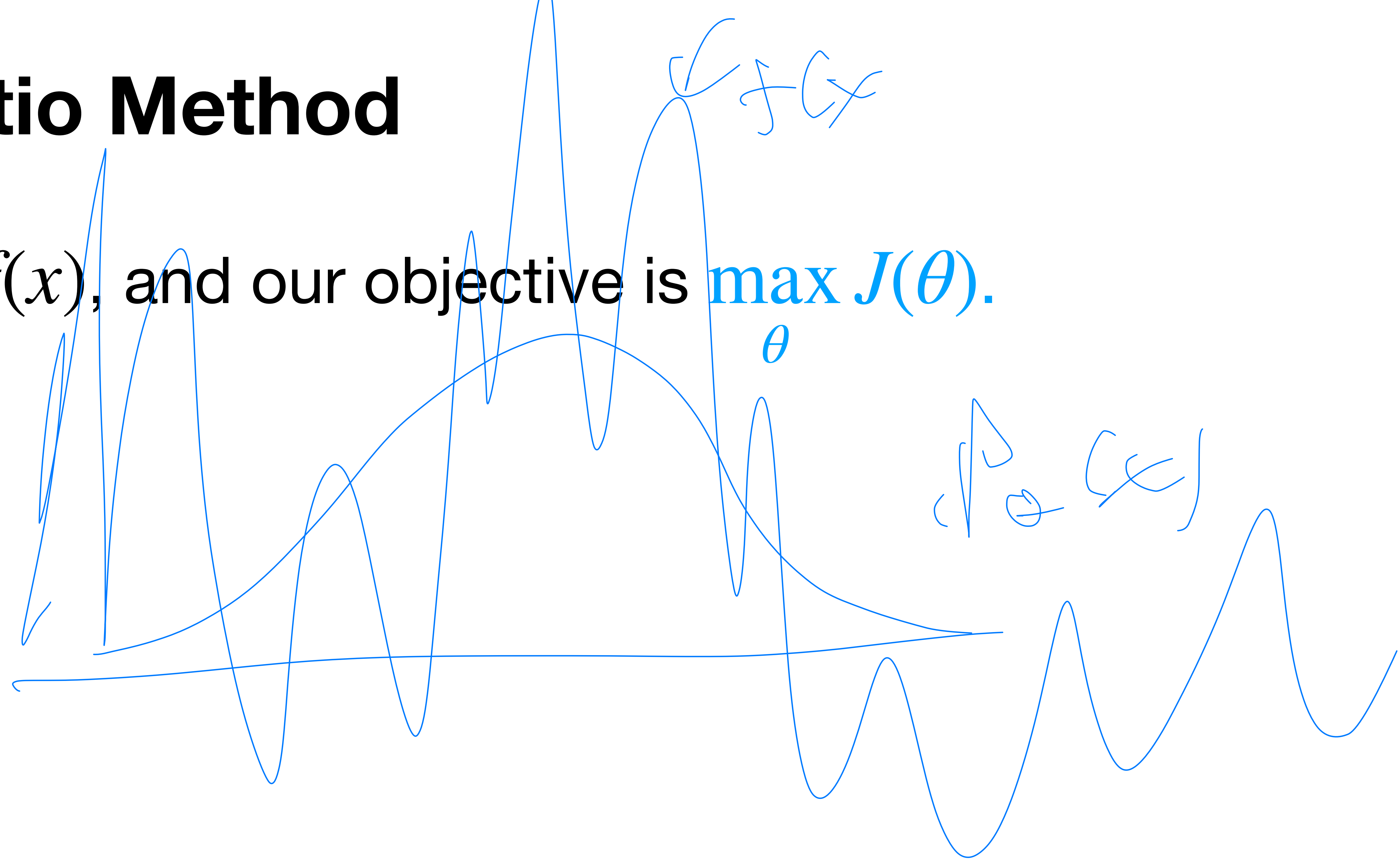
- Neural network $g_\theta : S \times [H] \mapsto \mathbb{R}^k$
- Parameters: $\theta \in R^d$,
  (and maybe $\sigma \in R^+$)

- Policy: a (multivariate) Normal with mean $g_\theta(s)$ and variance $\sigma^2 I$, i.e.
$$\pi_{\theta,\sigma}(\cdot \mid s, h) = \mathcal{N}(g_\theta(s, h), \sigma^2 I)$$

The Likelihood Ratio Method

The Likelihood Ratio Method

- Suppose $J(\theta) = \mathbb{E}_{x \sim P_\theta}\left[f(x)\right] = \sum_x P_\theta(x)f(x)$, and our objective is $\max_\theta J(\theta)$.



The Likelihood Ratio Method

- Suppose $J(\theta) = \mathbb{E}_{x \sim P_\theta}\left[f(x)\right] = \sum_x P_\theta(x) f(x)$, and our objective is $\max_\theta J(\theta)$.
- Computing $\nabla_\theta J(\theta)$ exactly may be difficult to compute (due to the sum over x).

The Likelihood Ratio Method

- Suppose $J(\theta) = \mathbb{E}_{x \sim P_\theta}\left[f(x)\right] = \sum_x P_\theta(x) f(x)$, and our objective is $\max_\theta J(\theta)$.
- Computing $\nabla_\theta J(\theta)$ exactly may be difficult to compute (due to the sum over x).
  - Can we estimate $\nabla_\theta J(\theta)$?

The Likelihood Ratio Method

- Suppose $J(\theta) = \mathbb{E}_{x \sim P_\theta}\left[f(x)\right] = \sum_x P_\theta(x) f(x)$, and our objective is $\max_\theta J(\theta)$.
- Computing $\nabla_\theta J(\theta)$ exactly may be difficult to compute (due to the sum over x).
    - Can we estimate $\nabla_\theta J(\theta)$?
    - Suppose we can: compute $f(x)$, $P_\theta(x)$, and $\nabla P_\theta(x)$ & sample $x \sim P_\theta$

The Likelihood Ratio Method

- Suppose $J(\theta) = \mathbb{E}_{x \sim P_\theta}\left[f(x)\right] = \sum_x P_\theta(x) f(x)$, and our objective is $\max_\theta J(\theta)$.
- Computing $\nabla_\theta J(\theta)$ exactly may be difficult to compute (due to the sum over x).
  - Can we estimate $\nabla_\theta J(\theta)$?
  - Suppose we can: compute $f(x)$, $P_\theta(x)$, and $\nabla P_\theta(x)$ & sample $x \sim P_\theta$
- We have that:

$$\nabla_\theta J(\theta) = \mathbb{E}_{x \sim P_\theta(x)}\left[\nabla_\theta \log P_\theta(x)\, f(x)\right]$$

The Likelihood Ratio Method

- Suppose $J(\theta) = \mathbb{E}_{x \sim P_\theta}\left[f(x)\right] = \sum_x P_\theta(x) f(x)$, and our objective is $\max_\theta J(\theta)$.
- Computing $\nabla_\theta J(\theta)$ exactly may be difficult to compute (due to the sum over x).
  - Can we estimate $\nabla_\theta J(\theta)$?
  - Suppose we can: compute $f(x)$, $P_\theta(x)$, and $\nabla P_\theta(x)$ & sample $x \sim P_\theta$
- We have that:

$$\nabla_\theta J(\theta) = \mathbb{E}_{x \sim P_\theta(x)}\left[\nabla_\theta \log P_\theta(x)\, f(x)\right]$$

Proof:

The Likelihood Ratio Method

- Suppose $J(\theta) = \mathbb{E}_{x \sim P_\theta}\left[f(x)\right] = \sum_x P_\theta(x) f(x)$, and our objective is $\max_\theta J(\theta)$.
- Computing $\nabla_\theta J(\theta)$ exactly may be difficult to compute (due to the sum over x).
  - Can we estimate $\nabla_\theta J(\theta)$?
  - Suppose we can: compute $f(x)$, $P_\theta(x)$, and $\nabla P_\theta(x)$ & sample $x \sim P_\theta$
- We have that:

$$\nabla_\theta J(\theta) = \mathbb{E}_{x \sim P_\theta(x)}\left[\nabla_\theta \log P_\theta(x)\, f(x)\right]$$

Proof:

$$\nabla_\theta J(\theta) = \sum_x \nabla_\theta P_\theta(x) f(x)$$

The Likelihood Ratio Method

- Suppose $J(\theta) = \mathbb{E}_{x\sim P_\theta}\left[f(x)\right] = \sum_x P_\theta(x) f(x)$, and our objective is $\max_\theta J(\theta)$.
- Computing $\nabla_\theta J(\theta)$ exactly may be difficult to compute (due to the sum over x).
  - Can we estimate $\nabla_\theta J(\theta)$?
  - Suppose we can: compute $f(x)$, $P_\theta(x)$, and $\nabla P_\theta(x)$ & sample $x \sim P_\theta$
- We have that:

$$\nabla_\theta J(\theta) = \mathbb{E}_{x\sim P_\theta(x)}\left[\nabla_\theta \log P_\theta(x)\, f(x)\right]$$

Proof:

$$\nabla_\theta J(\theta) = \sum_x \nabla_\theta P_\theta(x) f(x)$$

$$= \sum_x P_\theta(x) \frac{\nabla_\theta P_\theta(x)}{P_\theta(x)} f(x)$$

The Likelihood Ratio Method

- Suppose $J(\theta) = \mathbb{E}_{x\sim P_\theta}\left[f(x)\right] = \sum_x P_\theta(x)f(x)$, and our objective is $\max_\theta J(\theta)$.
- Computing $\nabla_\theta J(\theta)$ exactly may be difficult to compute (due to the sum over x).
  - Can we estimate $\nabla_\theta J(\theta)$?
  - Suppose we can: compute $f(x)$, $P_\theta(x)$, and $\nabla P_\theta(x)$ & sample $x \sim P_\theta$
- We have that:

$$\nabla_\theta J(\theta) = \mathbb{E}_{x\sim P_\theta(x)}\left[\nabla_\theta \log P_\theta(x)\, f(x)\right]$$

Proof:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \sum_x \nabla_\theta P_\theta(x) f(x) \\
&= \sum_x P_\theta(x) \frac{\nabla_\theta P_\theta(x)}{P_\theta(x)} f(x) \\
&= \sum_x P_\theta(x)\, \nabla_\theta \log P_\theta(x) f(x)
\end{aligned}$$

The Likelihood Ratio Method, continued

The Likelihood Ratio Method, continued

- We have:

$$\nabla_\theta J(\theta) = \mathbb{E}_{x \sim P_\theta(x)} \left[\nabla_\theta \log P_\theta(x)\, f(x) \right]$$

The Likelihood Ratio Method, continued

- We have:

$$\nabla_\theta J(\theta) = \mathbb{E}_{x \sim P_\theta(x)} \left[\nabla_\theta \log P_\theta(x)\, f(x) \right]$$

- An unbiased estimate is given by:

$\widehat{\nabla}_\theta J(\theta) = \nabla_\theta \log P_\theta(x) \cdot f(x)$, where $x \sim P_\theta$

The Likelihood Ratio Method, continued

- We have:

$$\nabla_\theta J(\theta) = \mathbb{E}_{x \sim P_\theta(x)} \left[\nabla_\theta \log P_\theta(x) \, f(x) \right]$$

- An unbiased estimate is given by:

$\widehat{\nabla}_\theta J(\theta) = \nabla_\theta \log P_\theta(x) \cdot f(x)$, where $x \sim P_\theta$

- We can lower variance by draw N i.i.d. samples from P_θ and averaging:

$$\widehat{\nabla}_\theta J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log P_\theta(x_i) \, f(x_i)$$

Today

- Recap
- Gradient Descent
- Policy Gradient
  - Likelihood ratio method
  - REINFORCE
  - Estimation

REINFORCE: A Policy Gradient Algorithm

REINFORCE: A Policy Gradient Algorithm

- Let $\rho_\theta(\tau)$ be the probability of a trajectory $\tau = \{s_0, a_0, s_1, a_1, \ldots, s_{H-1}, a_{H-1}\}$, i.e.

$$\rho_\theta(\tau) = \mu(s_0)\pi_\theta(a_0 \mid s_0)P(s_1 \mid s_0, a_0)\ldots P(s_{H-1} \mid s_{H-2}, a_{H-2})\pi_\theta(a_{H-1} \mid s_{H-1})$$

REINFORCE: A Policy Gradient Algorithm

- Let $\rho_\theta(\tau)$ be the probability of a trajectory $\tau = \{s_0, a_0, s_1, a_1, \ldots, s_{H-1}, a_{H-1}\}$, i.e.
$$\rho_\theta(\tau) = \mu(s_0)\pi_\theta(a_0 \mid s_0)P(s_1 \mid s_0, a_0)\ldots P(s_{H-1} \mid s_{H-2}, a_{H-2})\pi_\theta(a_{H-1} \mid s_{H-1})$$

- Let $R(\tau)$ be the cumulative reward on trajectory τ, i.e. $R(\tau) := \sum_{h=0}^{H-1} r(s_h, a_h)$

REINFORCE: A Policy Gradient Algorithm

- Let $\rho_\theta(\tau)$ be the probability of a trajectory $\tau = \{s_0, a_0, s_1, a_1, \ldots, s_{H-1}, a_{H-1}\}$, i.e.

$$\rho_\theta(\tau) = \mu(s_0)\pi_\theta(a_0 \mid s_0)P(s_1 \mid s_0, a_0)\ldots P(s_{H-1} \mid s_{H-2}, a_{H-2})\pi_\theta(a_{H-1} \mid s_{H-1})$$

- Let $R(\tau)$ be the cumulative reward on trajectory τ, i.e. $R(\tau) := \sum_{h=0}^{H-1} r(s_h, a_h)$

- Our objective function is:

$$J(\theta) \ = E_{\tau \sim \rho_\theta}\Big[R(\tau)\Big]$$

REINFORCE: A Policy Gradient Algorithm

- Let $\rho_\theta(\tau)$ be the probability of a trajectory $\tau = \{s_0, a_0, s_1, a_1, \ldots, s_{H-1}, a_{H-1}\}$, i.e.

$$\rho_\theta(\tau) = \mu(s_0)\pi_\theta(a_0 \mid s_0)P(s_1 \mid s_0, a_0)\ldots P(s_{H-1} \mid s_{H-2}, a_{H-2})\pi_\theta(a_{H-1} \mid s_{H-1})$$

- Let $R(\tau)$ be the cumulative reward on trajectory τ, i.e. $R(\tau) := \sum_{h=0}^{H-1} r(s_h, a_h)$

- Our objective function is:

$$J(\theta) \ = E_{\tau\sim\rho_\theta}\Big[R(\tau)\Big]$$

- The REINFORCE Policy Gradient expression:

$$\nabla_\theta J(\theta) := \mathbb{E}_{\tau\sim\rho_\theta}\left[\left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h)\right) R(\tau)\right]$$

Proof

Proof

- From the likelihood ratio method, we have:

$$\nabla_\theta J(\theta) := \mathbb{E}_{\tau \sim \rho_\theta} \left[\nabla_\theta \ln \rho_\theta(\tau) \; R(\tau) \right]$$

Proof

- From the likelihood ratio method, we have:

$$\nabla_\theta J(\theta) := \mathbb{E}_{\tau \sim \rho_\theta} \left[\nabla_\theta \ln \rho_\theta(\tau) \; R(\tau) \right]$$

- We have:

$$\begin{aligned}
\nabla_\theta \ln \rho_\theta(\tau) &= \nabla_\theta \left(\ln \rho(s_0) + \ln \pi_\theta(a_0 \mid s_0) + \ln P(s_1 \mid s_0, a_0) + \ldots \right) \\
&= \nabla_\theta \left(\ln \pi_\theta(a_0 \mid s_0) + \ln \pi_\theta(a_1 \mid s_1) \ldots \right) \\
&= \left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) \right)
\end{aligned}$$

Today

- Recap
- Gradient Descent
- Policy Gradient
    - Likelihood ratio method
    - REINFORCE
    - Estimation

Obtaining an Unbiased Gradient Estimate at θ_0

$$\nabla_\theta J(\theta) := \mathbb{E}_{\tau \sim \rho_\theta}\left[\left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h)\right) R(\tau)\right]$$

Obtaining an Unbiased Gradient Estimate at θ_0

$$\nabla_\theta J(\theta) := \mathbb{E}_{\tau \sim \rho_\theta} \left[\left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) \right) R(\tau) \right]$$

1. Obtain a trajectory $\tau \sim \rho_{\theta_0}$
   (which we can do in our learning setting)

Obtaining an Unbiased Gradient Estimate at θ_0

$$\nabla_\theta J(\theta) := \mathbb{E}_{\tau\sim\rho_\theta}\left[\left(\sum_{h=0}^{H-1}\nabla_\theta \ln \pi_\theta(a_h \,|\, s_h)\right)R(\tau)\right]$$

1. Obtain a trajectory $\tau \sim \rho_{\theta_0}$
   (which we can do in our learning setting)
2. Set:

$$\widetilde{\nabla}_\theta J(\theta_0) := \left(\sum_{h=0}^{H-1}\nabla \ln \pi_{\theta_0}(a_h \,|\, s_h)\right)R(\tau)$$

Obtaining an Unbiased Gradient Estimate at θ_0

$$\nabla_\theta J(\theta) := \mathbb{E}_{\tau \sim \rho_\theta}\left[\left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h | s_h)\right) R(\tau)\right]$$

1. Obtain a trajectory $\tau \sim \rho_{\theta_0}$

   (which we can do in our learning setting)
2. Set:

$$\widetilde{\nabla}_\theta J(\theta_0) := \left(\sum_{h=0}^{H-1} \nabla \ln \pi_{\theta_0}(a_h | s_h)\right) R(\tau)$$

We have: $\mathbb{E}[\widetilde{\nabla}_\theta J(\theta_0)] = \nabla_\theta J(\pi_{\theta_0})$

PG with REINFORCE:

PG with REINFORCE:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$

PG with REINFORCE:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \dots$
2. For k = 0, … :

PG with REINFORCE:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For k = 0, … :
    1. Obtain a trajectory $\tau \sim \rho_{\theta^k}$

       Set $\widetilde{\nabla}_\theta J(\theta^k) = \sum_{h=0}^{H-1} \nabla \ln \pi_{\theta^k}(a_h \mid s_h) R(\tau)$

PG with REINFORCE:

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \dots$
2. For k = 0, … :
   1. Obtain a trajectory $\tau \sim \rho_{\theta^k}$

      Set $\widetilde{\nabla}_\theta J(\theta^k) = \sum_{h=0}^{H-1} \nabla \ln \pi_{\theta^k}(a_h \mid s_h) R(\tau)$

   2. Update: $\theta^{k+1} = \theta^k + \eta^k \widetilde{\nabla}_\theta J(\theta^k)$

The (mini-batch) PG procedure with REINFORCE

(reducing variance using batch sizes of M)

The (mini-batch) PG procedure with REINFORCE

(reducing variance using batch sizes of M)

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$

The (mini-batch) PG procedure with REINFORCE

(reducing variance using batch sizes of M)

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For k = 0, … :

The (mini-batch) PG procedure with REINFORCE

(reducing variance using batch sizes of M)

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For k = 0, … :
    1. Init $g = 0$ and do M times:

       Obtain a trajectory $\tau \sim \rho_{\theta^k}$

       Update: $g \leftarrow g + \sum_{h=0}^{H-1} \nabla \ln \pi_{\theta^k}(a_h | s_h) R(\tau)$

The (mini-batch) PG procedure with REINFORCE

(reducing variance using batch sizes of M)

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For k = 0, … :
    1. Init $g = 0$ and do M times:

       Obtain a trajectory $\tau \sim \rho_{\theta^k}$

       Update: $g \leftarrow g + \sum_{h=0}^{H-1} \nabla \ln \pi_{\theta^k}(a_h | s_h) R(\tau)$
    2. Set $\widetilde{\nabla}_\theta J(\theta^k) := \frac{1}{M} g$

The (mini-batch) PG procedure with REINFORCE

(reducing variance using batch sizes of M)

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For k = 0, … :
   1. Init $g = 0$ and do M times:

      Obtain a trajectory $\tau \sim \rho_{\theta^k}$

      Update: $g \leftarrow g + \sum_{h=0}^{H-1} \nabla \ln \pi_{\theta^k}(a_h | s_h) R(\tau)$
   2. Set $\widetilde{\nabla}_\theta J(\theta^k) := \frac{1}{M} g$
   3. Update: $\theta^{k+1} = \theta^k + \eta^k \widetilde{\nabla}_\theta J(\theta^k)$

The (mini-batch) PG procedure with REINFORCE

(reducing variance using batch sizes of M)

1. Initialize θ_0, parameters: $\eta_1, \eta_2, \ldots$
2. For k = 0, … :
   1. Init $g = 0$ and do M times:

      Obtain a trajectory $\tau \sim \rho_{\theta^k}$

      Update: $g \leftarrow g + \sum_{h=0}^{H-1} \nabla \ln \pi_{\theta^k}(a_h \mid s_h) R(\tau)$
   2. Set $\widetilde{\nabla}_\theta J(\theta^k) := \frac{1}{M} g$
   3. Update: $\theta^{k+1} = \theta^k + \eta^k \widetilde{\nabla}_\theta J(\theta^k)$

We sill have: $\mathbb{E}[\widetilde{\nabla}_\theta J(\theta^k)] = \nabla_\theta J(\theta^k)$

Today

- Recap
- Gradient Descent
- Policy Gradient
    - Likelihood ratio method
    - REINFORCE
    - Estimation
- Other Gradient Expressions

Other PG formulas (that are lower variance for sampling)

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta}\left[\left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h)\right) R(\tau)\right]$$

Other PG formulas
(that are lower variance for sampling)

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta}\left[\left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h)\right) R(\tau)\right]$$

$$= \mathbb{E}_{\tau \sim \rho_\theta}\left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) Q_h^{\pi_\theta}(s_h, a_h)\right]$$

Other PG formulas
(that are lower variance for sampling)

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta}\left[\left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \,|\, s_h)\right) R(\tau)\right]$$

$$= \mathbb{E}_{\tau \sim \rho_\theta}\left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \,|\, s_h) Q_h^{\pi_\theta}(s_h, a_h)\right]$$

$$= \mathbb{E}_{\tau \sim \rho_\theta}\left[\sum_{h=0}^{H-1}\left(\nabla_\theta \ln \pi_\theta(a_h \,|\, s_h) \sum_{t=h}^{H-1} r_t\right)\right]$$

Other PG formulas
(that are lower variance for sampling)

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta}\left[\left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h)\right) R(\tau)\right]$$

$$= \mathbb{E}_{\tau \sim \rho_\theta}\left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) Q_h^{\pi_\theta}(s_h, a_h)\right]$$

$$= \mathbb{E}_{\tau \sim \rho_\theta}\left[\sum_{h=0}^{H-1}\left(\nabla_\theta \ln \pi_\theta(a_h \mid s_h) \sum_{t=h}^{H-1} r_t\right)\right]$$

Intuition: Change action distribution at h only affects rewards later on…

Other PG formulas
(that are lower variance for sampling)

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \rho_\theta} \left[\left(\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) \right) R(\tau) \right]$$

$$= \mathbb{E}_{\tau \sim \rho_\theta} \left[\sum_{h=0}^{H-1} \nabla_\theta \ln \pi_\theta(a_h \mid s_h) Q_h^{\pi_\theta}(s_h, a_h) \right]$$

$$= \mathbb{E}_{\tau \sim \rho_\theta} \left[\sum_{h=0}^{H-1} \left(\nabla_\theta \ln \pi_\theta(a_h \mid s_h) \sum_{t=h}^{H-1} r_t \right) \right]$$

Intuition: Change action distribution at h only affects rewards later on…

HW: You will show these simplified version are also valid PG expressions

Summary:

- PG approach: let's directly try to optimize the objective function of interest!

Attendance:
bit.ly/3RcTC9T

End2End Optimization

Feedback:
bit.ly/3RHtlxy