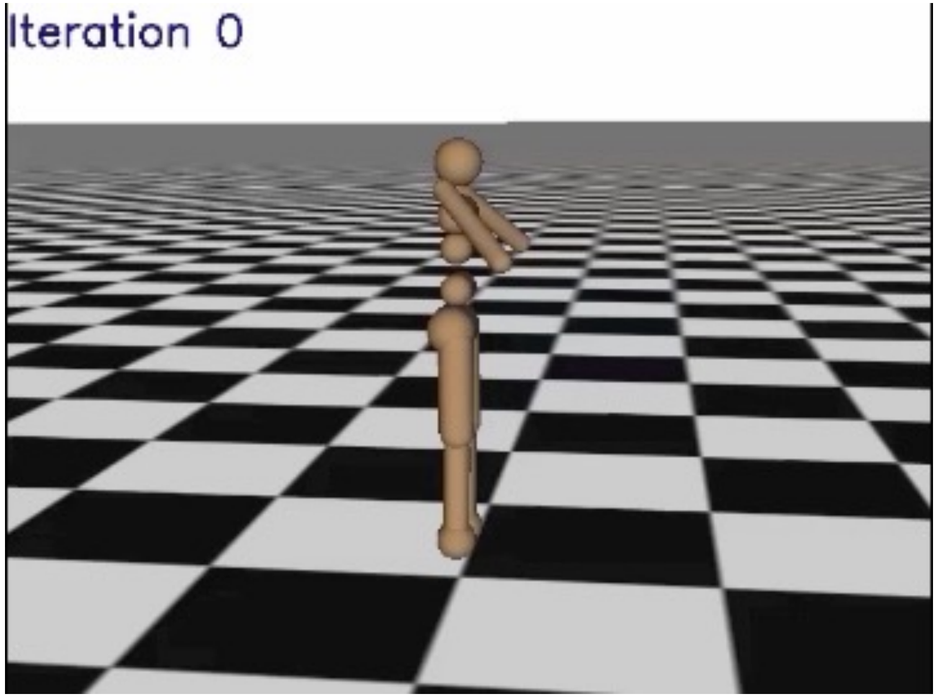
Reinforcement Learning
Spring 2024

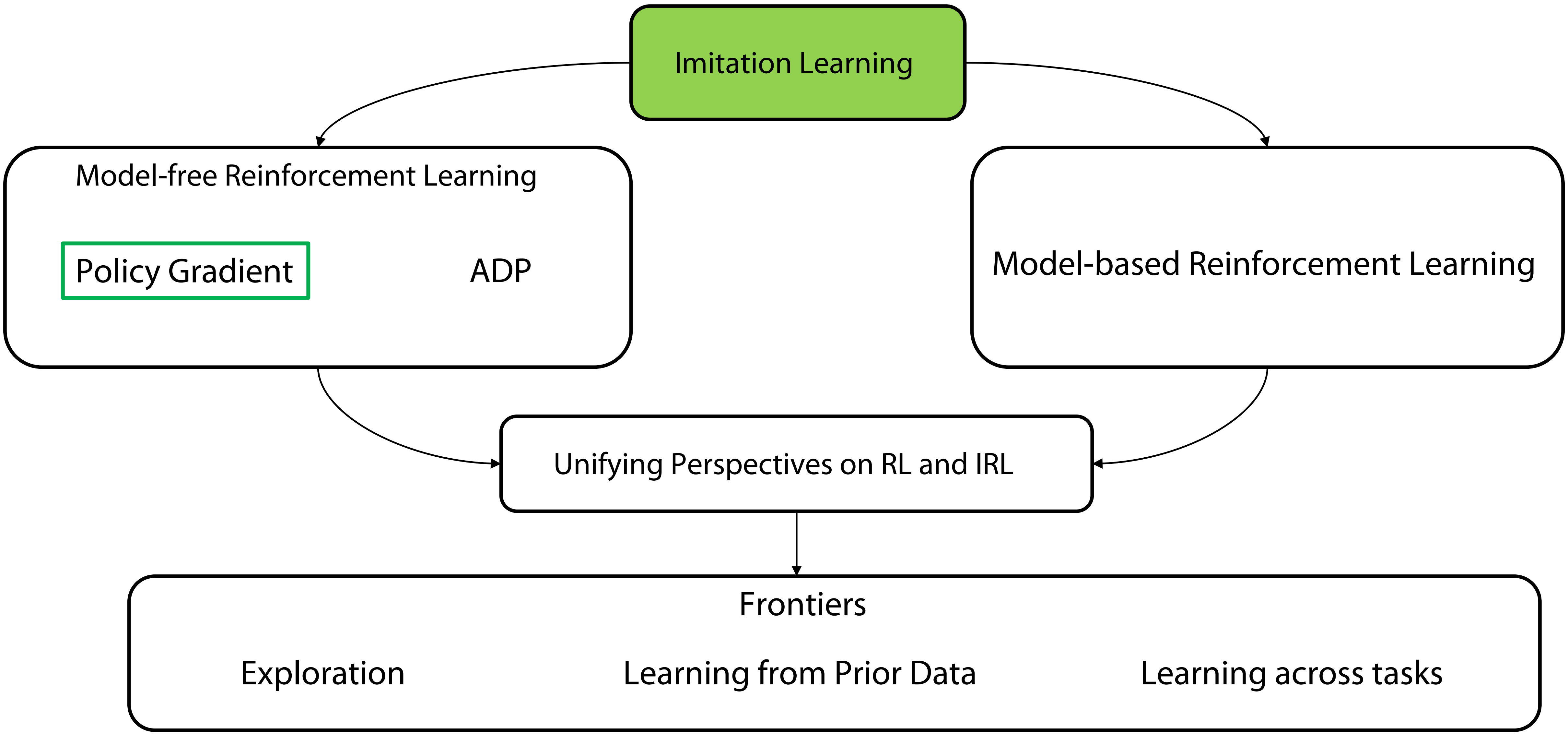
Abhishek Gupta

TAs: Patrick Yin, Qiuyu Chen

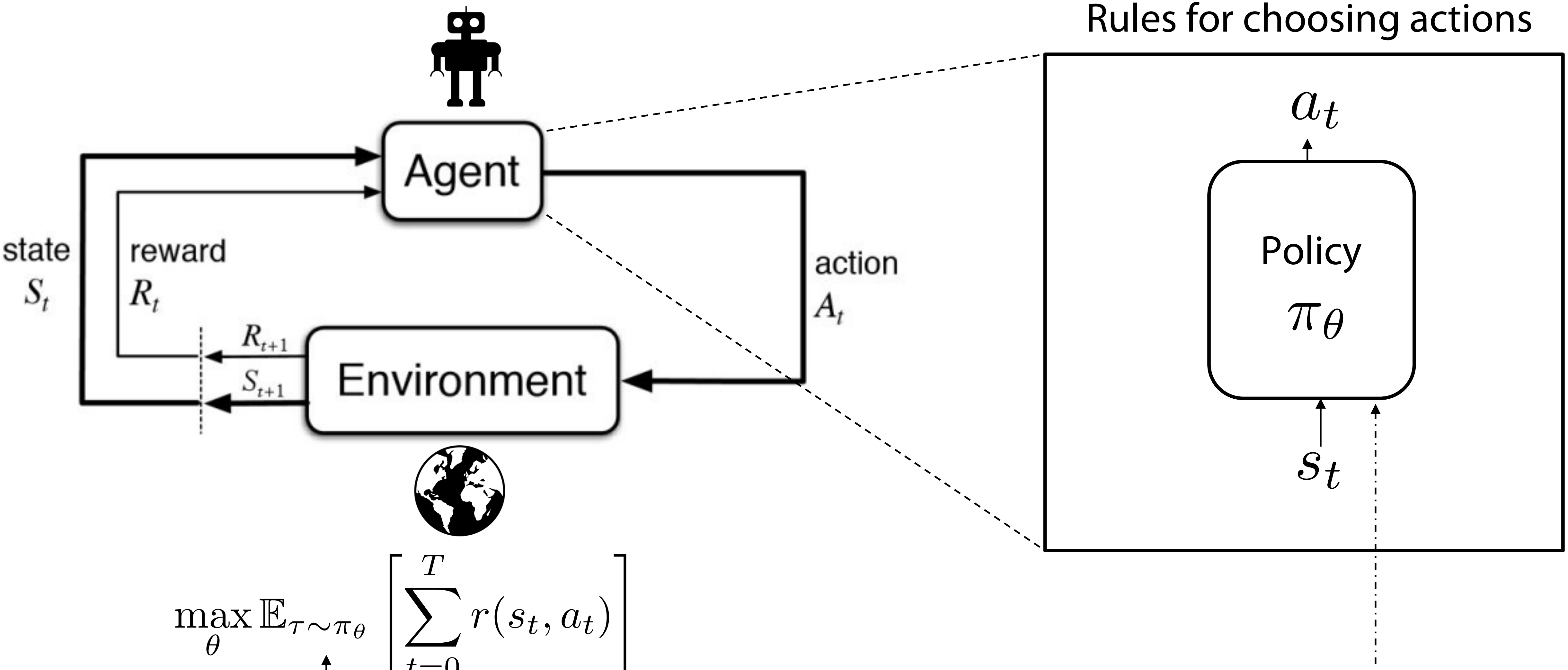


Slides inspired by Sergey Levine, Liyiming Ke

Class Structure



Objective of Reinforcement Learning

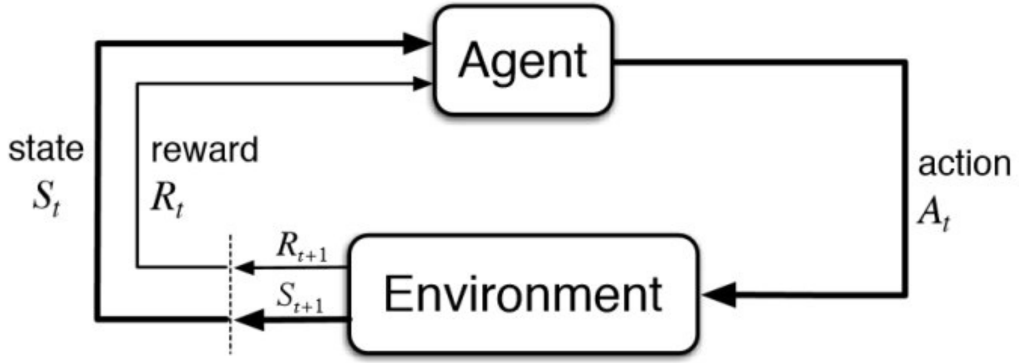


$$\max_\theta \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$

Trajectory sampled using policy

Finite horizon vs infinite horizon objective

$$\max_{\theta} \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$



Finite horizon

$$\mathbb{E}_{\pi_\theta^t} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$

$$\mathbb{E}_{\pi_\theta^t} \left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right]$$

Time-dependent policy
(not stationary)

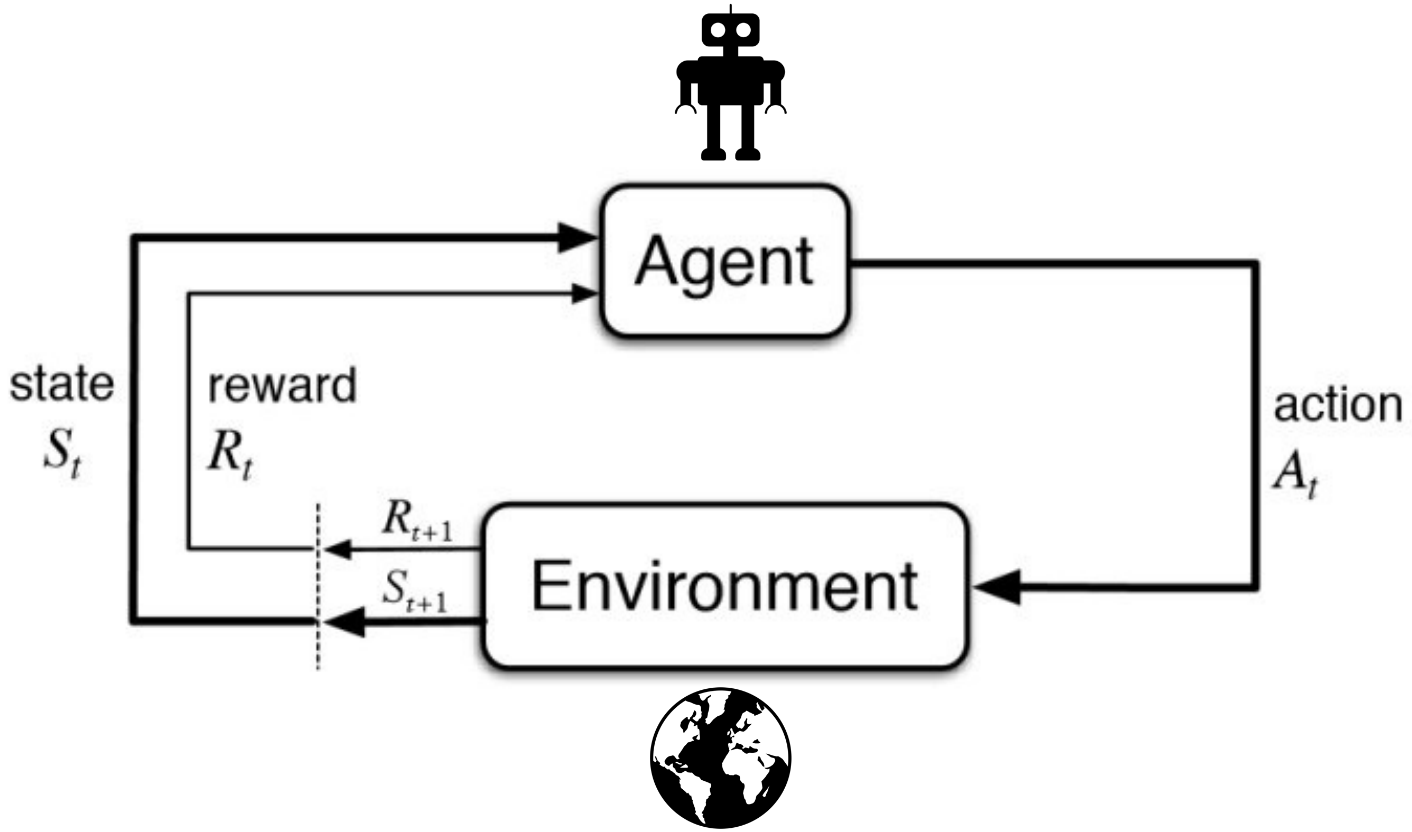
Infinite horizon discounted

$$\mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]$$

Time-independent (stationary) policy
→ Need discount to prevent blow up

Lemma: there always exists a stationary optimal policy

Objective of Reinforcement Learning



$$\max_{\theta} \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$

Trajectory sampled using policy

Assumptions:

1. Rewards are additive
2. Dynamics can be sampled from, but functional form is unknown
3. Rewards are provided as every state is visited, functional form is unknown

Connection to Optimal Control

Closely related: typically problem of finding control given a plant

$$\min_{x,u} \int_0^X L(t, x(t), u(t)).dx$$

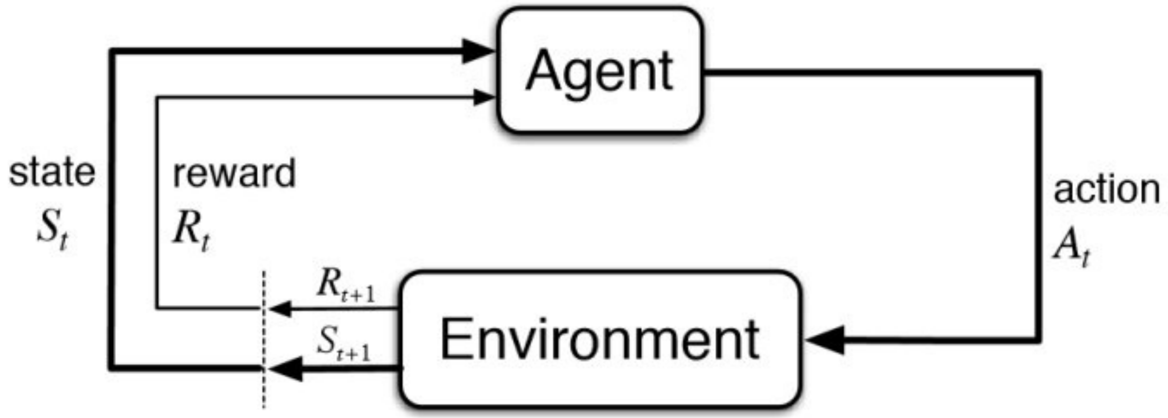
w.r.t

$$x'(t) = f(x(t), u(t))$$

$$\max_\theta \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$

<u>Main difference:</u> model known vs unknown
<u>Minor differences:</u> Cost vs reward, discrete vs continuous time

How should we optimize this objective?



$$\max_\theta \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} r(s_t, a_t)\right]$$

Gradient Ascent

Dynamic Programming

Model-Based Optimization

Each method has it's own +/-

Lecture outline

Deriving the Policy Gradient

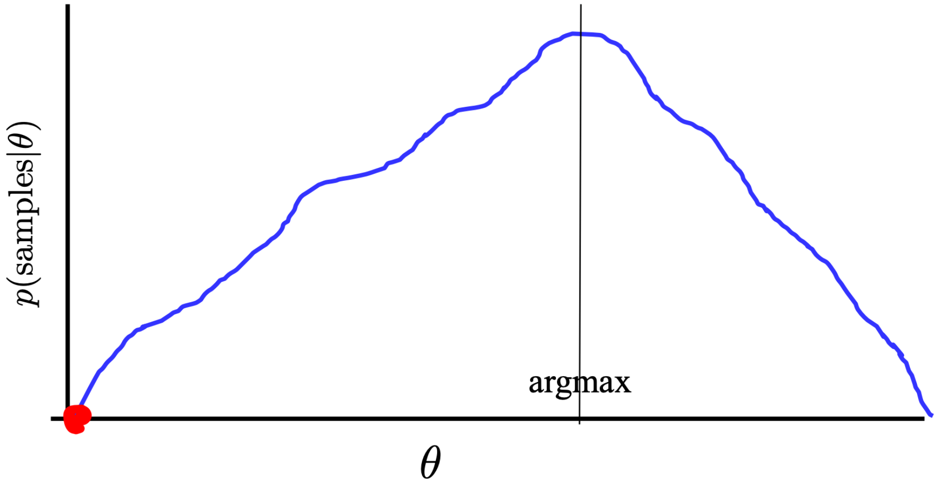
↓

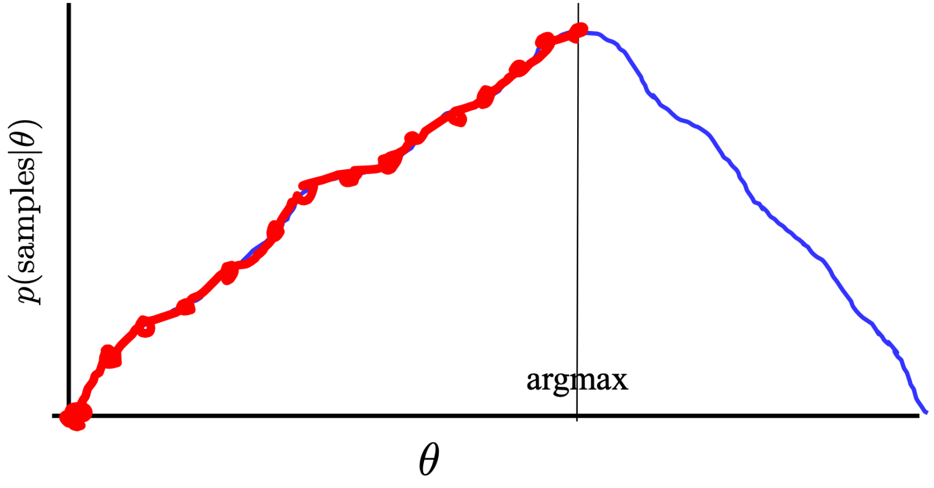
What makes the Policy Gradient Challenging? - Variance

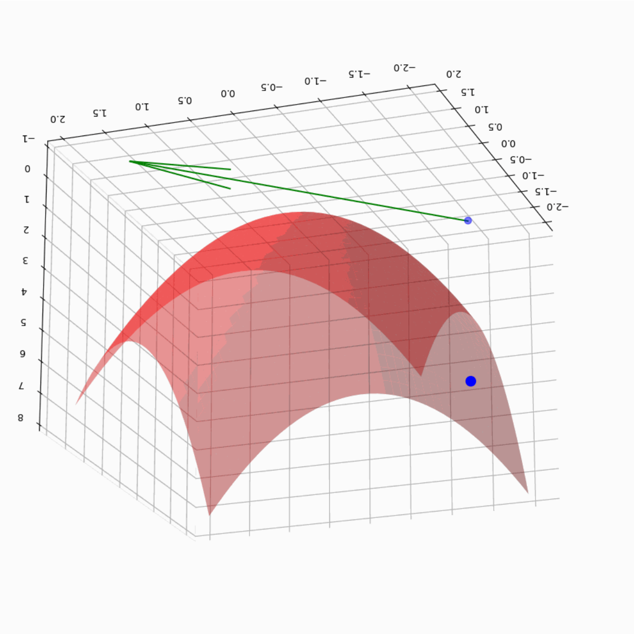
↓

What makes the Policy Gradient Challenging? – Covariant Parameterization

Gradient Ascent







Simple view – move the parameters in the direction of the gradient of the objective

$$\theta_{i+1} = \theta_i + \alpha \nabla_\theta J(\theta)|_{\theta=\theta_i}$$

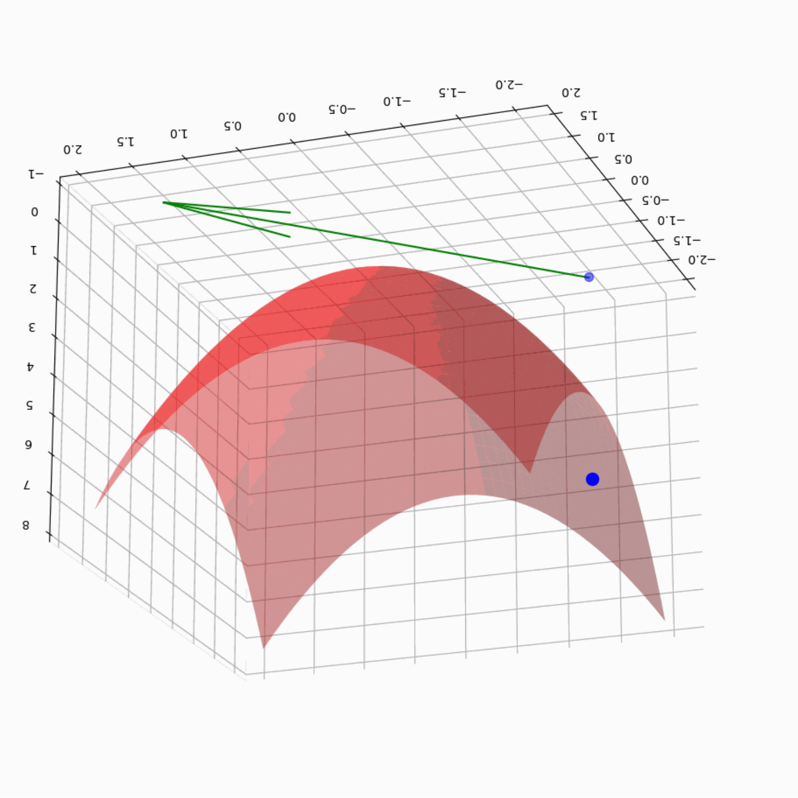
More later: can be derived as steepest ascent in Euclidean norm

Gradient Ascent for Supervised Learning

Recall our imitation learning objective

$$\arg\max_\theta \mathbb{E}_{(s^*,a^*)\sim\mathcal{D}}\left[\log \pi_\theta(a^*|s^*)\right]$$

Let's apply gradient ascent



$$\nabla_\theta \mathbb{E}_{(s^*,a^*)\sim\mathcal{D}}\left[\log \pi_\theta(a^*|s^*)\right]$$

$$\nabla_\theta \int p(s^*, a^*) \log \pi_\theta(a^*|s^*) ds^* da^*$$

$$\int p(s^*, a^*) \nabla_\theta \log \pi_\theta(a^*|s^*) ds^* da^*$$

$$\mathbb{E}_{(s^*,a^*)\sim\mathcal{D}}\left[\nabla_\theta \log \pi_\theta(a^*|s^*)\right]$$

Compute gradient and average

Ok let's do gradient ascent for the RL objective

$$\max_\theta \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$

$$= \int p_\theta(\tau) R(\tau) d\tau$$

<u>REINFORCE gradient descent (RL)</u>

$$\nabla_\theta \mathbb{E}_{x \sim p_\theta(x)} \left[f(x) \right]$$

(Cannot simply compute average of expectation)

<u>Standard gradient descent (supervised learning)</u>

Gradient wrt expectation variable, not of integrand!

$$\nabla_\theta \mathbb{E}_{x \sim g(x)} \left[f_\theta(x) \right]$$

(Whiteboard)

(Gradient passes inside the expectation – compute gradient and average)

Taking the gradient of sum of rewards

$$\max_\theta \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} r(s_t, a_t)\right]$$

Let's take the gradient of this objective

$$J(\theta) = \int p_\theta(\tau) R(\tau) d(\tau)$$

Let's think about this from the trajectory view

$$\nabla_\theta J(\theta) = \nabla_\theta \int p_\theta(\tau) R(\tau) d(\tau)$$

We need to express this in a way that we can evaluate with expectations

$$= \int \nabla_\theta p_\theta(\tau) R(\tau) d(\tau) \quad = \int \frac{p_\theta(\tau)}{p_\theta(\tau)} \nabla_\theta p_\theta(\tau) R(\tau) d(\tau)$$

$$= \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) R(\tau) d(\tau) \ = \mathbb{E}_{p_\theta(\tau)}\left[\nabla_\theta \log p_\theta(\tau) R(\tau)\right]$$

REINFORCE trick

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

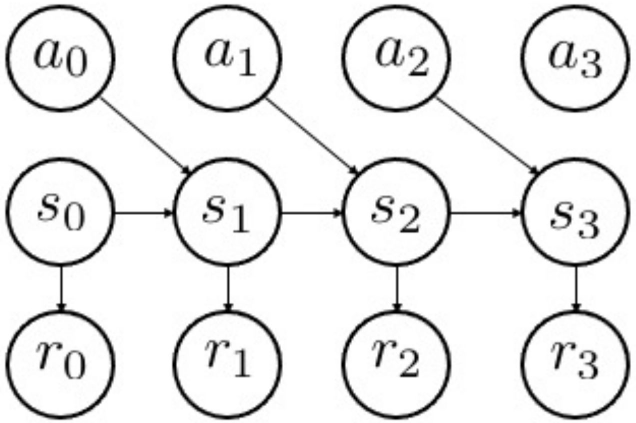
Use chain rule

Taking the gradient of return

Initial State Dynamics Policy

$$p_\theta(\tau) = p(s_0)\Pi_{t=0}^{T-1} p(s_{t+1}|s_t, a_t)\pi(a_t|s_t)$$

(Ancestral sampling)



$$\log p_\theta(\tau) = \log p(s_0) + \sum_{t=0}^{T-1} \log p(s_{t+1}|s_t, a_t) + \log \pi(a_t|s_t)$$

$$\nabla_\theta \log p_\theta(\tau) = \cancel{\nabla_\theta \log p(s_0)} + \sum_{t=0}^{T-1} \cancel{\nabla_\theta \log p(s_{t+1}|s_t, a_t)} + \nabla_\theta \log \pi(a_t|s_t)$$

$$\nabla_\theta \log p_\theta(\tau) = \sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t)$$

Model Free!!

Taking the gradient of return

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\nabla_\theta \log p_\theta(\tau) \sum_{t=0}^{T} r(s_t, a_t) \right]$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{\substack{s_0 \sim p(s_0) \\ s_{t+1} \sim p(s_{t+1}|s_t, a_t) \\ a_t \sim \pi(a_t|s_t)}} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t'=0}^{T} r(s_t, a_t) \right]$$

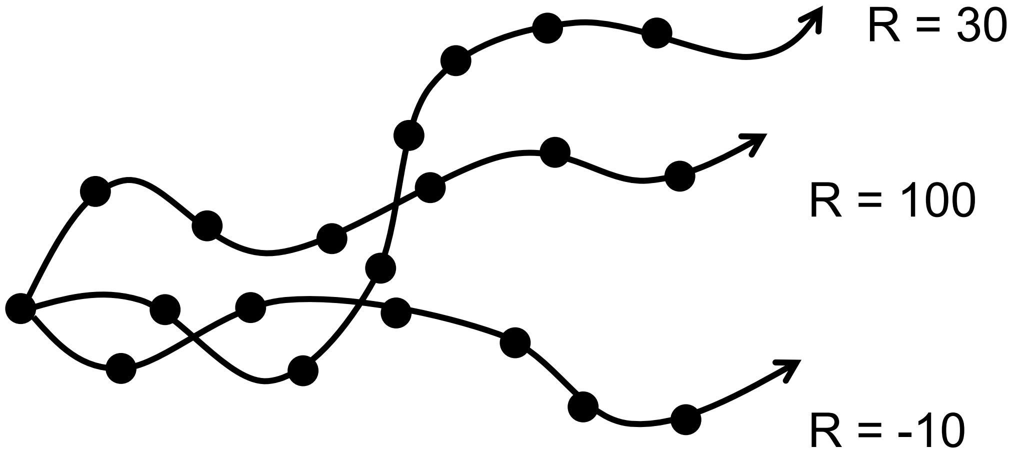
$$\approx \frac{1}{N} \sum_{i=0}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i) \sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$ (approximating using samples)

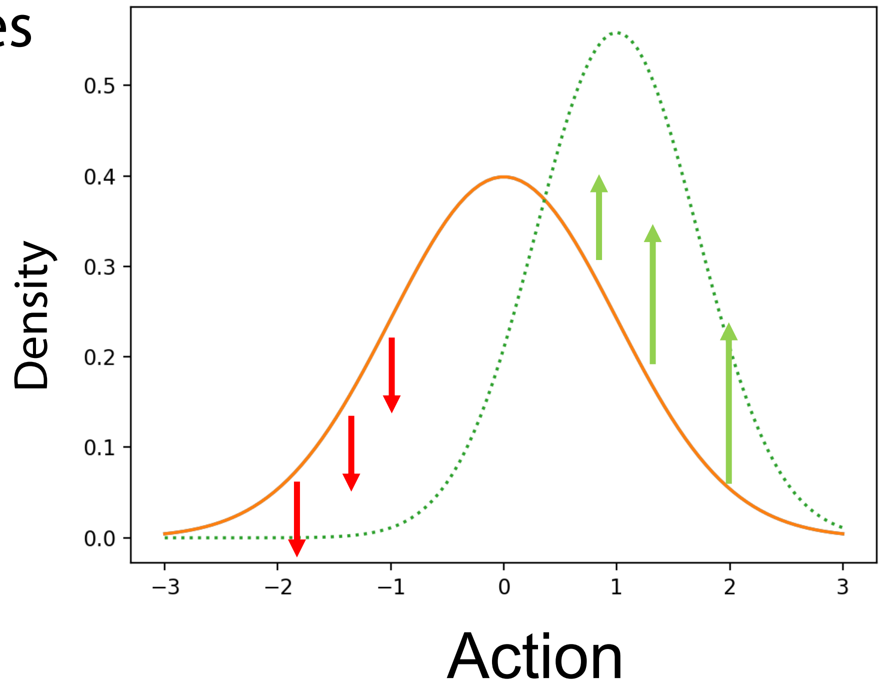
(Monte-Carlo approximation)

What does this mean?

$$\nabla_\theta J(\theta) = \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) d\tau \approx \frac{1}{N} \sum_{i=0}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$

Increase the likelihood of actions in high return trajectories

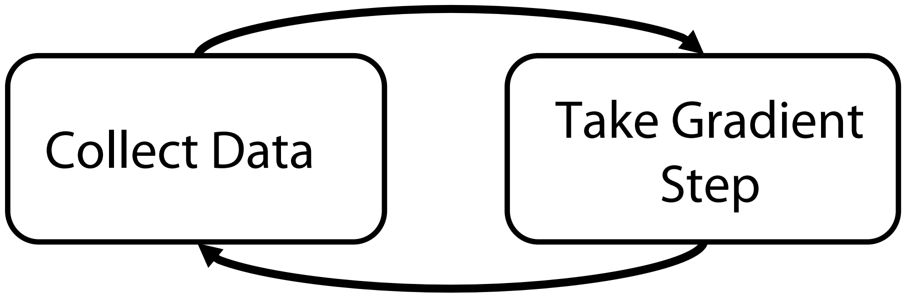




Resulting Algorithm (REINFORCE)

$$\nabla_\theta J(\theta) = \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) d\tau$$

$$\theta_{i+1} = \theta_i + \alpha \nabla_\theta J(\theta)|_{\theta=\theta_i}$$



REINFORCE algorithm:

On-policy

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run it on the robot)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

How is this related to supervised learning?

<u>Reinforcement Learning</u>

$$\nabla_\theta J(\theta) = \int p_\theta(\tau)\nabla_\theta \log p_\theta(\tau) d\tau$$

$$\approx \frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$

<u>Supervised Learning</u>

$$\max_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\log p_\theta(y|x)\right]$$

$$\approx \frac{1}{N}\sum_i \nabla_\theta \log p_\theta(y^i|x^i)$$

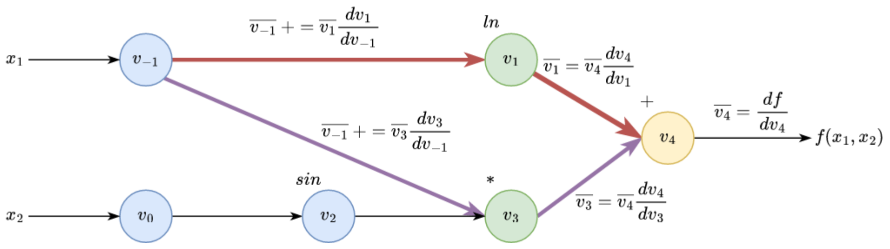
PG = select good data + increase likelihood of selected data

How do we implement this?

REINFORCE algorithm:

1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run it on the robot)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right)\left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

$$\nabla_\theta J(\theta) = \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) d\tau \approx \frac{1}{N} \sum_{i=0}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i) \sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$



PyTorch

Compute gradients with autodiff

Sum up rewards in a trajectory

How do we implement this?

Maximum likelihood:

```
# Given:
# actions - (N*T) x Da tensor of actions
# states - (N*T) x Ds tensor of states
# Build the graph:
logits = policy.predictions(states) # This should return (N*T) x Da tensor of action logits
negative_likelihoods = tf.nn.softmax_cross_entropy_with_logits(labels=actions, logits=logits)
loss = tf.reduce_mean(negative_likelihoods)
gradients = loss.gradients(loss, variables)
```

^Standard maximum likelihood training

How do we implement this?

REINFORCE algorithm:

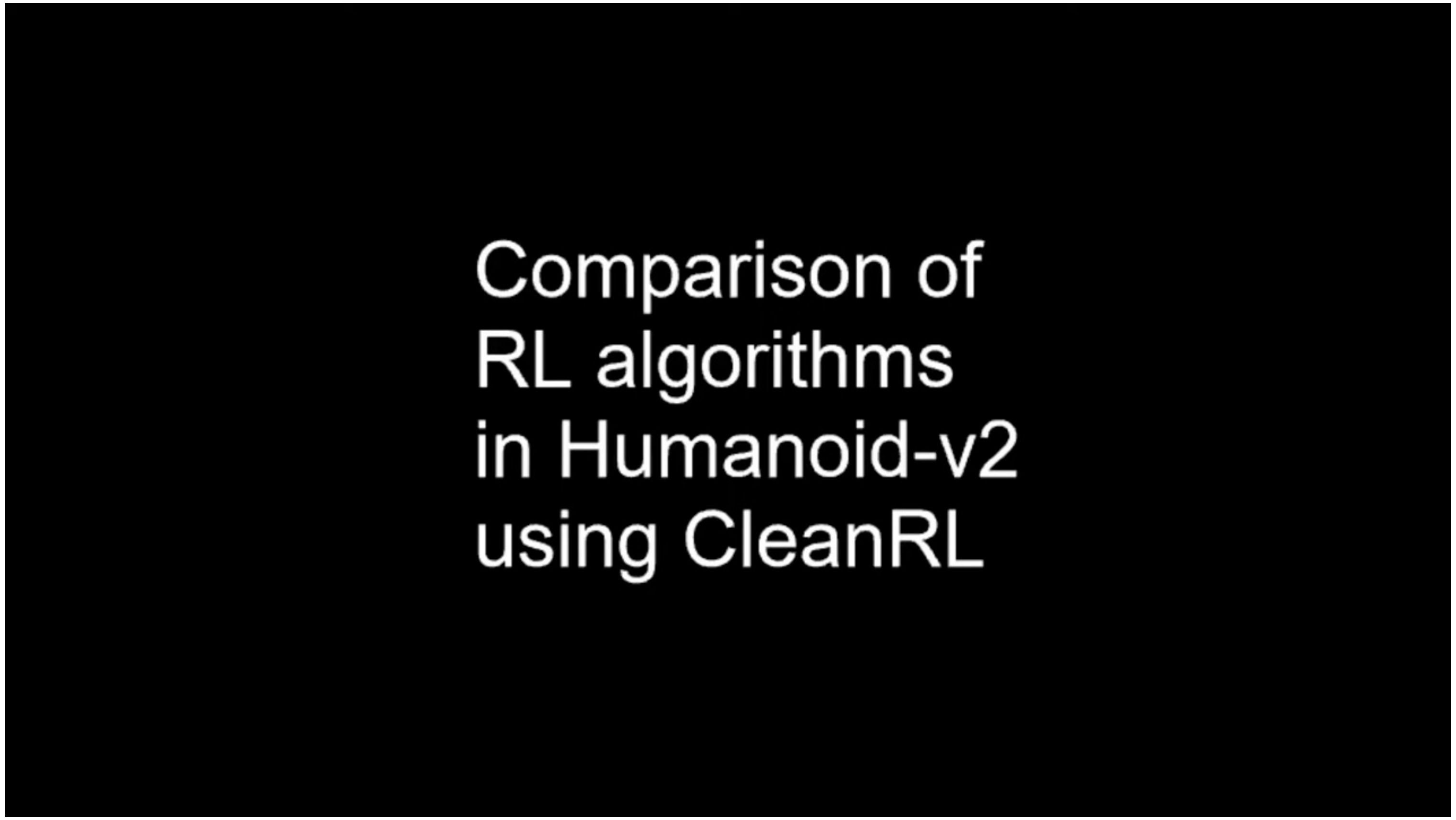
1. sample $\{\tau^i\}$ from $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ (run it on the robot)
2. $\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i|\mathbf{s}_t^i)\right)\left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right)$
3. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

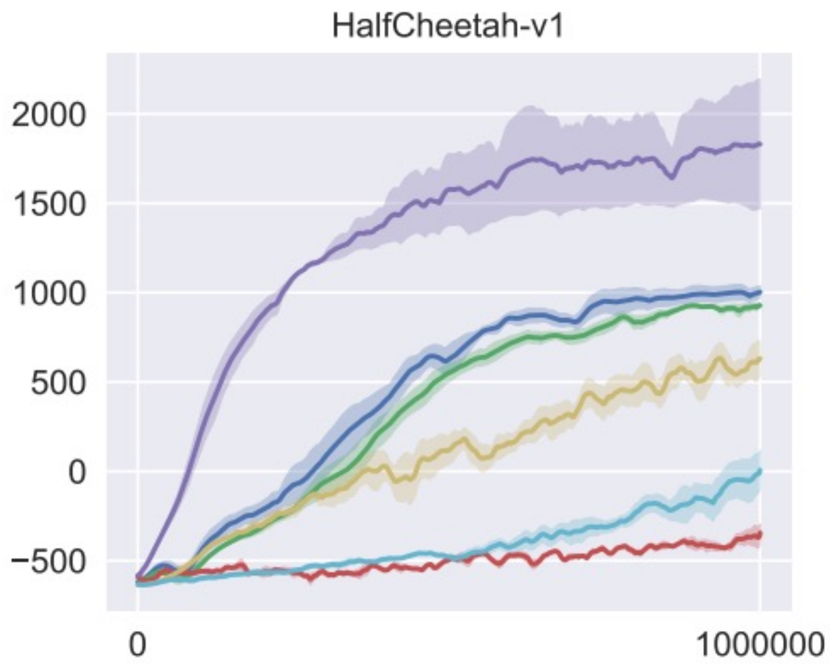
Policy gradient:

```
# Given:
# actions - (N*T) x Da tensor of actions
# states - (N*T) x Ds tensor of states
# q_values – (N*T) x 1 tensor of estimated state-action values  → Sum of rewards
# Build the graph:
logits = policy.predictions(states) # This should return (N*T) x Da tensor of action logits
negative_likelihoods = tf.nn.softmax_cross_entropy_with_logits(labels=actions, logits=logits)
weighted_negative_likelihoods = tf.multiply(negative_likelihoods, q_values)
loss = tf.reduce_mean(weighted_negative_likelihoods)
gradients = loss.gradients(loss, variables)
```

Formalizes the notion of trial and error

Does this work?





Kind of?

Lecture outline

Deriving the Policy Gradient

What makes the Policy Gradient Challenging? - Variance

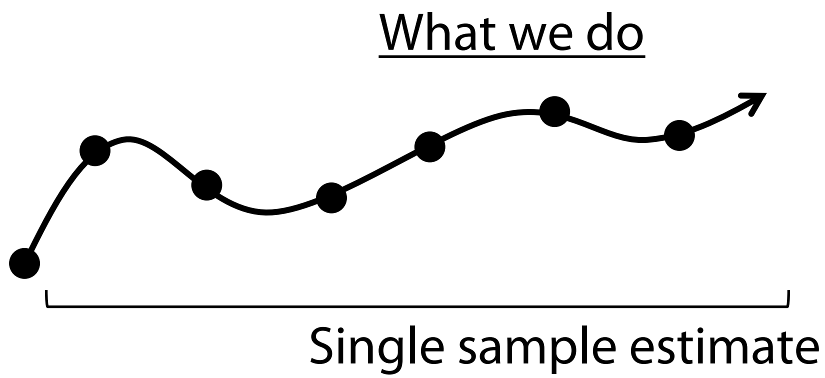
What makes the Policy Gradient Challenging? – Covariant Parameterization

What makes policy gradient challenging?

Hard to tell what matters without many samples

$$\nabla_\theta J(\theta) = \int p_\theta(\tau)\nabla_\theta \log p_\theta(\tau) d\tau$$

$$\approx \frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$



For every (s, a) pair, weight by only the sum of rewards in the current trajectory

Couples together all actions

Susceptible to scale variations

Susceptible to lucky samples

Makes policy gradient unstable, requires huge numbers of samples and huge batch size

What makes policy gradient challenging?

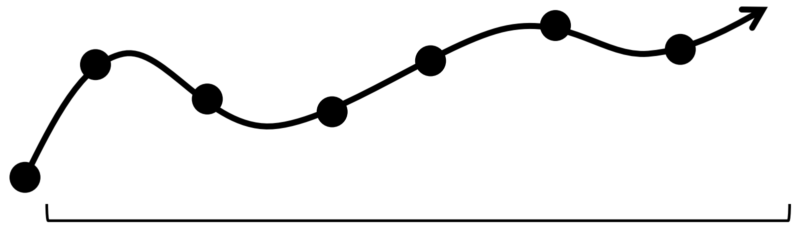
$$\nabla_\theta J(\theta) = \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) d\tau$$

$$\approx \frac{1}{N} \sum_{i=0}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$

High variance estimator!!

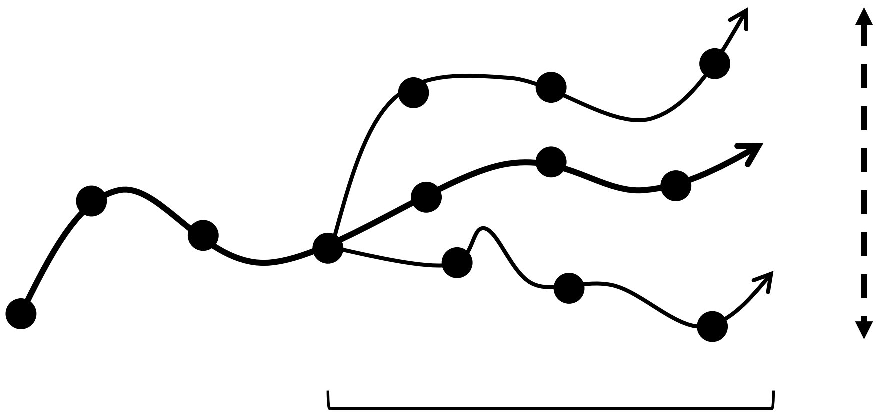
Hard to tell what matters without many samples

What we do



Single sample estimate

What we actually want



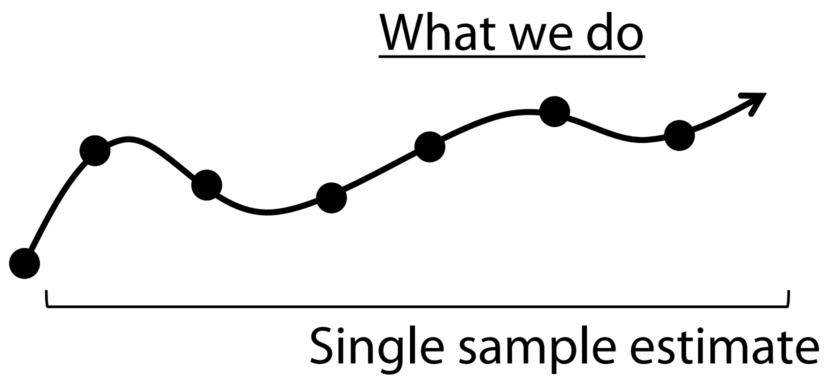
Averaged return estimate

What makes policy gradient challenging?

Hard to tell what matters without many samples

$$\nabla_\theta J(\theta) = \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) d\tau$$

$$\approx \frac{1}{N} \sum_{i=0}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$



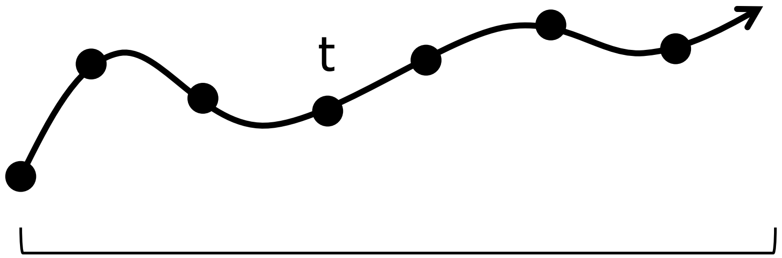
For every (s, a) pair, weight by only the sum of rewards in the current trajectory

Couples together all actions

Variance Reduction with Causality

Idea: Trajectory returns depend on past and future, but we only care about the future, since actions cannot affect the past. Instead, consider **“return-to-go”**

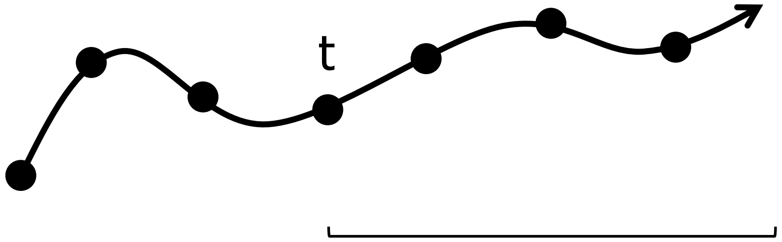
$$\approx \frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\underbrace{\sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)}_{\text{Includes } t' < t}$$



Full trajectory return

Ignore past terms $\Downarrow$

$$\frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\sum_{t'=t}^{T} r(s_t^i, a_t^i)$$



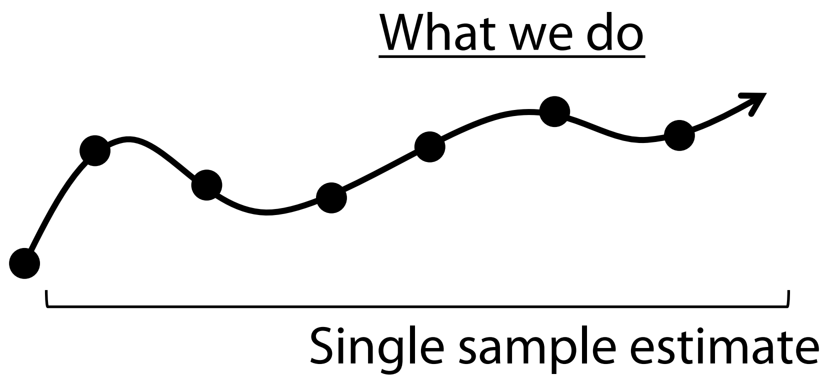
Return to go

What makes policy gradient challenging?

Hard to tell what matters without many samples

$$\nabla_\theta J(\theta) = \int p_\theta(\tau)\nabla_\theta \log p_\theta(\tau) d\tau$$

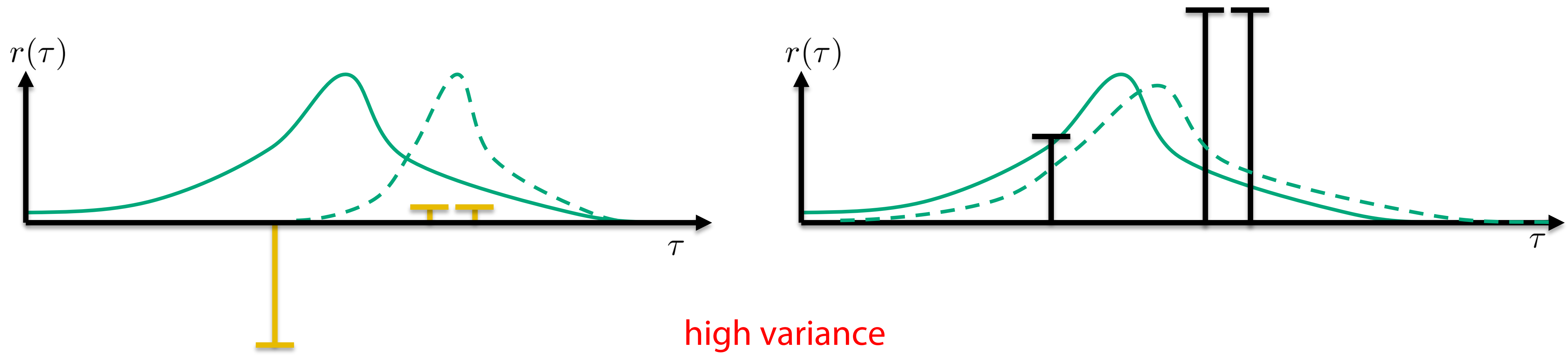
$$\approx \frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$



For every (s, a) pair, weight by only the sum of rewards in the current trajectory

Susceptible to scale variations

Policy gradient is susceptible to scale variations



Arbitrarily uncentered, scaled returns can lead to huge variance:

→ Imagine all rewards were positive, every action would be pushed up, some more than others

→ What if instead, we pushed down some actions and pushed up some others (even if rewards are positive)

Credit: Sergey Levine

Variance Reduction with a Baseline

Idea: We can reduce variance by subtracting a current state dependent function from the policy gradient return

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\left[\sum_{t'=t}^{T} r(s_{t'}^i, a_{t'}^i) - b(s_t)\right]$$

Baseline: Centers the returns, reduces variance

But does this increase bias??

Variance Reduction with a Baseline

$$\int_{\mathcal{S}} \int_{\mathcal{A}} p(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \left[\sum_{t'=t}^{T} r(s_{t'}, a_{t'}) - b(s_t) \right] \, ds_t \, da_t$$

$$\int_{\mathcal{S}} \int_{\mathcal{A}} p(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \left[\sum_{t'=t}^{T} r(s_{t'}, a_{t'}) \right] \, ds_t \, da_t - \boxed{\int_{\mathcal{S}} \int_{\mathcal{A}} p(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t) b(s_t) \, ds_t \, da_t}$$

Let us show this is 0!

Variance Reduction with a Baseline

$$\int \int p(s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \left[b(s_t)\right] ds_t da_t = \int \int p(s_t) \pi_\theta(a_t|s_t) \nabla_\theta \log \pi_\theta(a_t|s_t) \left[b(s_t)\right] ds_t da_t$$

$$= \int p(s_t) b(s_t) \int \pi_\theta(a_t|s_t) \nabla_\theta \log \pi_\theta(a_t|s_t) da_t ds_t$$

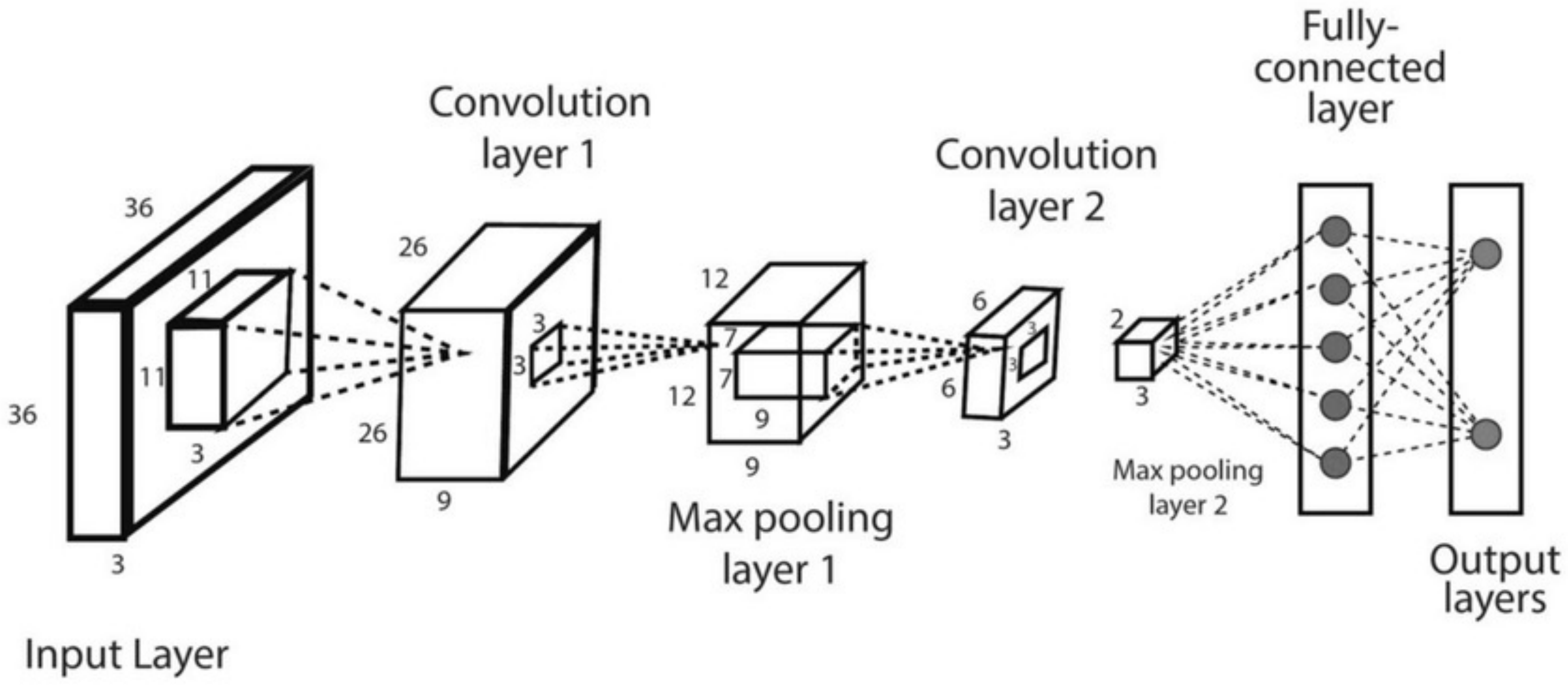
$$= \int p(s_t) b(s_t) \int \nabla_\theta \pi_\theta(a_t|s_t) da_t ds_t$$

$$= \int p(s_t) b(s_t) \nabla_\theta \int \pi_\theta(a_t|s_t) da_t ds_t = \int p(s_t) b(s_t) \nabla_\theta (1) ds_t = 0$$

Unbiased!

Learning Baselines

Baselines are typically learned as deep neural nets from R^s → R^1



$$\arg\min_{\hat{V}} \frac{1}{N} \sum_{j=1}^{N} \|\hat{V}(s_t^j) - \sum_{t=1}^{H} r(s_t^j, a_t^j)\|$$

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \left(\sum_{t'=t}^{T} r(s_{t'}, a_{t'}) - \hat{V}(s_t) \right) \right]$$

Minimize with Monte-Carlo regression at every iteration, club with policy gradient

Why do baselines really reduce variance?

Let's define variance: $\text{Var}[x] = E[x^2] - E[x]^2$ $\nabla_\theta J(\theta) = E_{\tau\sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau)(r(\tau) - b)]$

Whiteboard

$$\text{Var}[x] = E[x^2] - E[x]^2$$

$$\nabla_\theta J(\theta) = E_{\tau\sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau)(r(\tau) - b)]$$

$$\text{Var} = E_{\tau\sim p_\theta(\tau)}[(\nabla_\theta \log p_\theta(\tau)(r(\tau) - b))^2] - \underbrace{E_{\tau\sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau)(r(\tau) - b)]^2}_{\substack{\text{this bit is just } E_{\tau\sim p_\theta(\tau)}[\nabla_\theta \log p_\theta(\tau) r(\tau)] \\ \text{(baselines are unbiased in expectation)}}}$$

$$\frac{d\text{Var}}{db} = \frac{d}{db} E[g(\tau)^2(r(\tau) - b)^2] = \frac{d}{db}\left(\cancel{E[g(\tau)^2 r(\tau)^2]} - 2E[g(\tau)^2 r(\tau) b] + b^2 E[g(\tau)^2]\right)$$

$$= -2E[g(\tau)^2 r(\tau)] + 2bE[g(\tau)^2] = 0$$

$$b = \frac{E[g(\tau)^2 r(\tau)]}{E[g(\tau)^2]} \longleftarrow$$

This is just expected reward, but weighted by gradient magnitudes!

Lecture outline

Deriving the Policy Gradient

What makes the Policy Gradient Challenging? - Variance

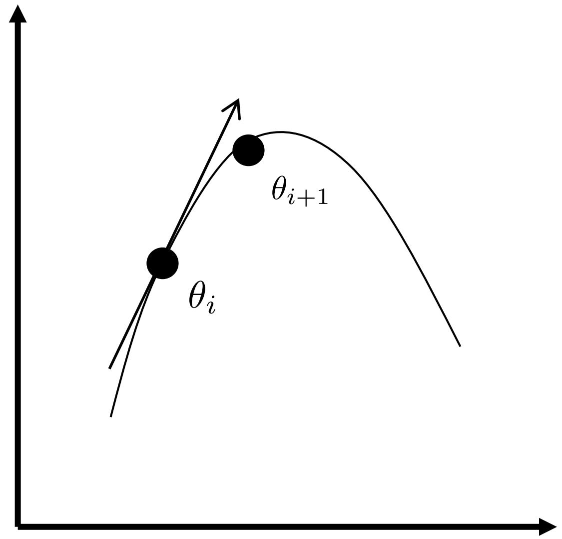
What makes the Policy Gradient Challenging? – Covariant Parameterization

Take a deeper look at REINFORCE

$$\nabla_\theta J(\theta) = \int p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) d\tau \approx \frac{1}{N} \sum_{i=0}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^i | s_t^i) \sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$

Gradient ascent is steepest ascent on linear approximation under the Euclidean metric!

$$\max_\theta \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right]$$
$$= J(\theta)$$



Take a deeper look at REINFORCE

$$\nabla_\theta J(\theta) = \int p_\theta(\tau)\nabla_\theta \log p_\theta(\tau) d\tau \approx \frac{1}{N}\sum_{i=0}^{N}\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i|s_t^i)\sum_{t'=0}^{T} r(s_{t'}^i, a_{t'}^i)$$

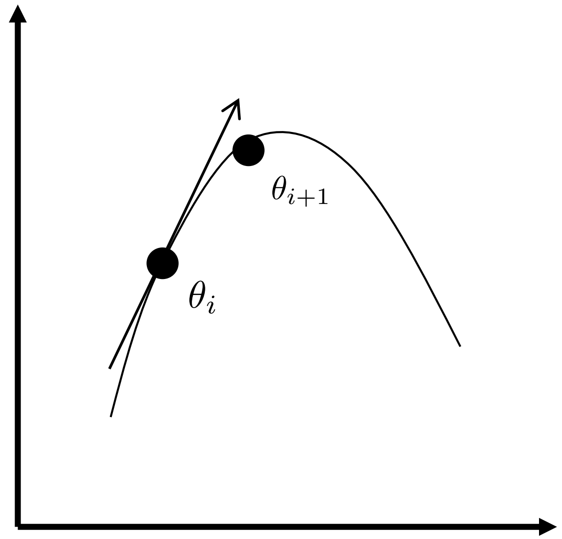
Gradient ascent is steepest ascent on linear approximation under the Euclidean metric!

$$\max \quad J(\theta_i) + \nabla_\theta J(\theta)|_{\theta=\theta_i}(\theta - \theta_i) \qquad \text{Linear approximation}$$

$$(\theta - \theta_i)^T(\theta - \theta_i) \leq \epsilon \qquad \text{Quadratic Constraint}$$

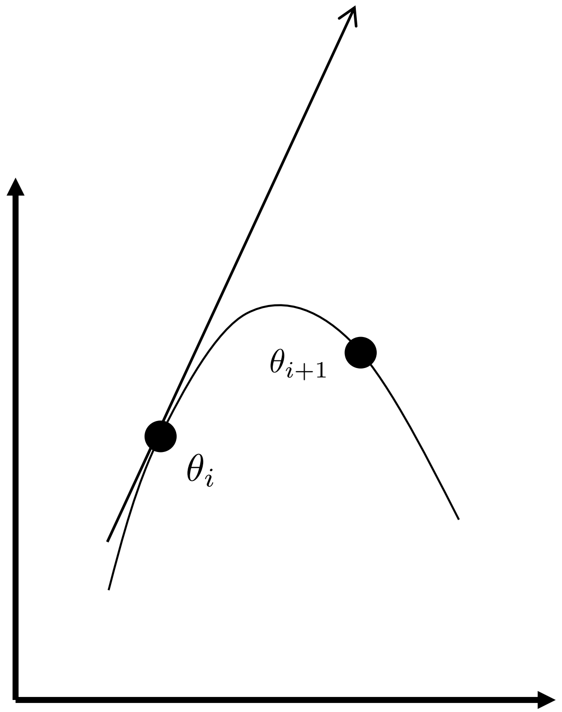
$$\downarrow$$

$$\theta = \theta_i + \alpha \nabla_\theta J(\theta)|_{\theta=\theta_i}$$



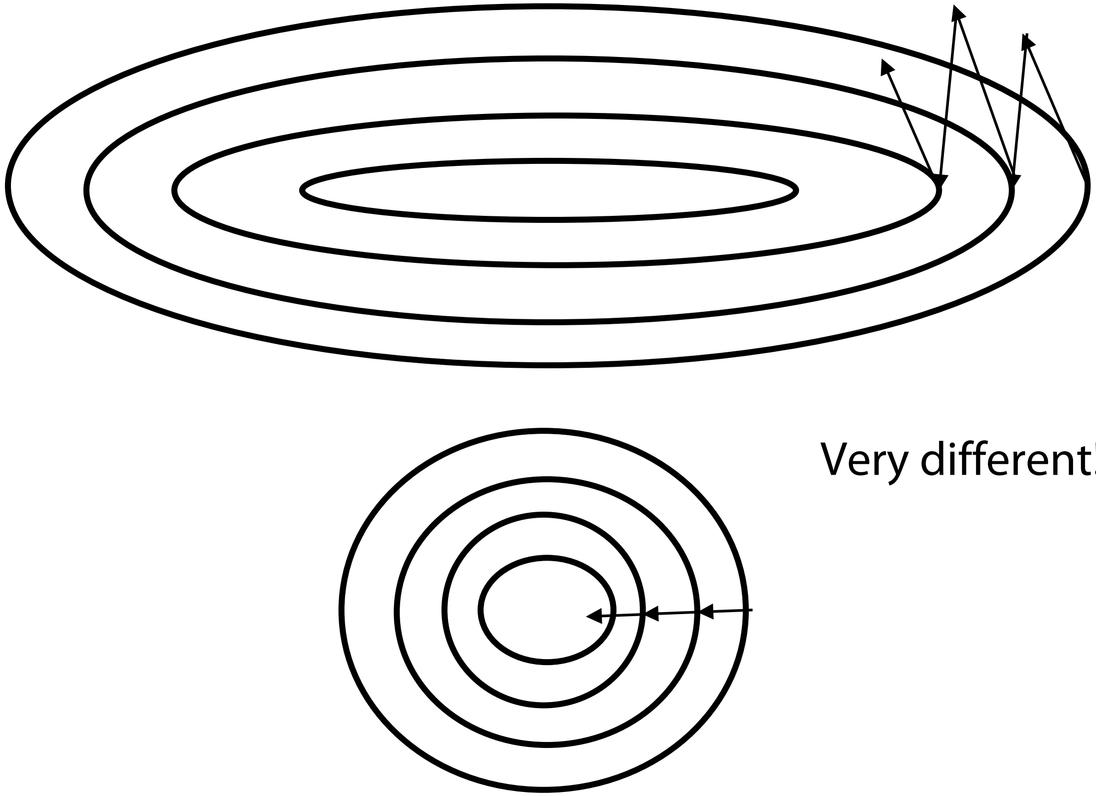
When might this fail?

<u>Large step sizes may cause collapse</u>



Must use very small step sizes, slow!

<u>Sensitive to Policy Parameterization</u>



Very different!

Can struggle for a deep neural network!

Parameterization dependence of PG

<u>Sensitive to Policy Parameterization</u>

$$L(\theta) = \theta_1 + \theta_2$$

$$L(\phi) = \phi_1^{0.5} + \phi_2^{-1}$$

$$\phi_1 = \theta_1^2$$

$$\phi_2 = \theta_2^{-1}$$

$$\nabla_{\theta_1} L = 1$$

$$\nabla_{\theta_2} L = 1$$

$$\nabla_{\phi_1} L = 0.5\phi_1^{-0.5} = 0.5\theta_1^{-1}$$

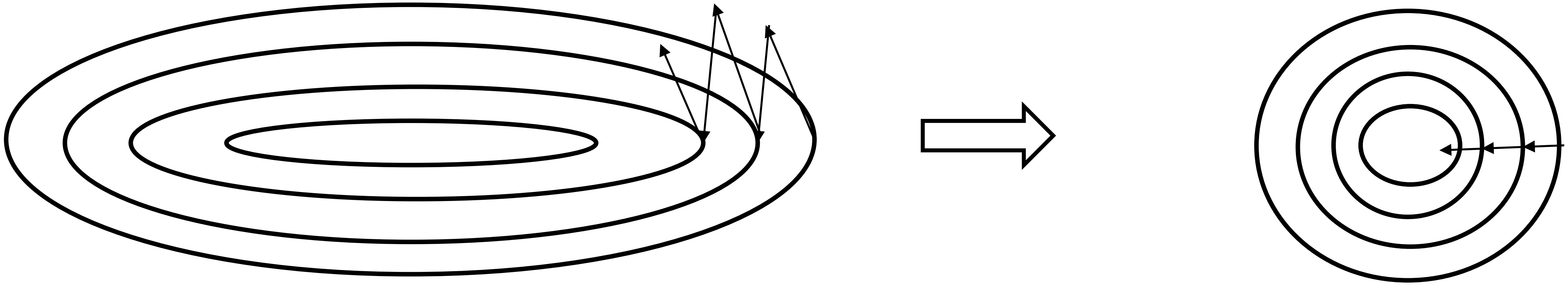
$$\nabla_{\phi_2} L = -\phi_2^{-2} = -\theta_2^2$$

Not covariant!

Modified Constraint on Policy Gradient

$$\max \quad J(\theta_i) + \nabla_\theta J(\theta)|_{\theta=\theta_i}(\theta - \theta_i)$$
$$(\theta - \theta_i)^T(\theta - \theta_i) \leq \epsilon$$

$$\max \quad J(\theta_i) + \nabla_\theta J(\theta)|_{\theta=\theta_i}(\theta - \theta_i)$$
$$(\theta - \theta_i)^T G(\theta - \theta_i) \leq \epsilon$$



$$\theta_{i+1} = \theta_i + \alpha G^{-1} \nabla_\theta J(\theta)|_{\theta=\theta_i}$$

Rescales according to G^{-1}

Adaptive choice of G can avoid sensitivity to policy parameterization!

Covariant Policy Gradient Updates

$$\max \quad J(\theta_i) + \nabla_\theta J(\theta)|_{\theta=\theta_i}(\theta - \theta_i)$$
$$(\theta - \theta_i)^T G(\theta - \theta_i) \leq \epsilon$$

What should G be?

$$\max \quad J(\theta_i) + \nabla_\theta J(\theta)|_{\theta=\theta_i}(\theta - \theta_i)$$
$$D_{\mathrm{KL}}(\pi_\theta || \pi_{\theta_i}) \leq \epsilon$$

Let us use the constraint as KL divergence on the policy (2nd order Taylor expansion)

Measures functional distance, not parameter distance

Second Order Expansion of KL Divergence

$$D_{KL}(\pi_\theta || \pi_{\theta_i}) \approx \int \pi_\theta \log \pi_\theta - \int \pi_\theta \log \pi_{\theta_i}$$

Whiteboard

(\theta_i - \theta)^2 p_\theta \pi_\theta d/dtheta \Log \pi_\theta

Log \pi_\theta

(\theta_i - \theta) \pi_\theta d/dtheta \Log \pi_\theta

Resulting "Natural" Policy Gradient

$$\max \quad J(\theta_i) + \nabla_\theta J(\theta)|_{\theta=\theta_i}(\theta - \theta_i)$$
$$D_{\mathrm{KL}}(\pi_\theta || \pi_{\theta_i}) \leq \epsilon$$

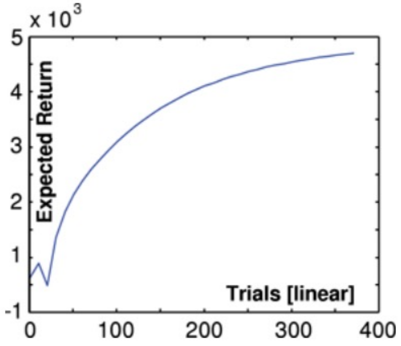
2nd order approximation of KL → Fisher Information Metric

$$F = \mathbb{E}_{\pi_\theta}\left[(\nabla_\theta \log \pi_\theta)(\nabla_\theta \log \pi_\theta)^T\right]$$

$$\max \quad J(\theta_i) + \nabla_\theta J(\theta)|_{\theta=\theta_i}(\theta - \theta_i)$$
$$(\theta - \theta_i)^T F (\theta - \theta_i) \leq \epsilon$$

Resulting update $\quad \theta_{i+1} = \theta_i + \alpha F^{-1} \nabla_\theta J(\theta)|_{\theta=\theta_i} \quad$ Covariant to parameterization

Natural Policy Gradient in Action

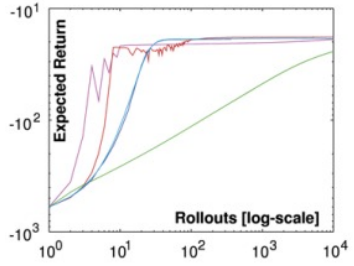


(a) Performance.

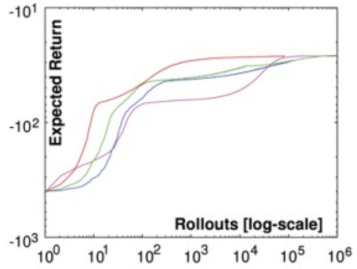
(b) Imitation learning.

(c) Initial reproduction.

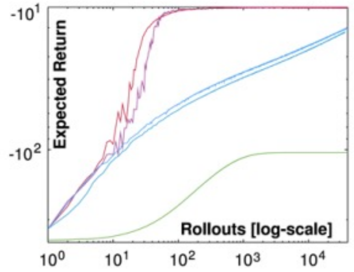
(d) After reinforcement learning.



(b) Minimum motor command with motor primitives



(c) Passing through a point with splines



(d) Passing through a point with motor primitives

Finite Difference Gradient
Vanilla Policy Gradient with constant baseline
Vanilla Policy Gradient with time-variant baseline
Episodic Natural Actor-Critic with single offset basis functions
Episodic Natural Actor-Critic with time-variant offset basis functions

Peters, Schaal '08

Lecture outline

Deriving the Policy Gradient

↓

What makes the Policy Gradient Challenging? - Variance

↓

What makes the Policy Gradient Challenging? – Covariant Parameterization

Fin.