Policy Search Methods

Pieter Abbeel
UC Berkeley

Jan Peters
TU Darmstadt

Motivation

Motivation

- Learning for high-dimensional robots is difficult:
    - Limit of Value Functions: fill-up state-space
    - Limit of Model Learning: accurate model!

Motivation

- Learning for high-dimensional robots is difficult:
  - Limit of Value Functions: fill-up state-space
  - Limit of Model Learning: accurate model!

- Starting with expert's knowledge helps!
  - Improving upon Demonstrations
  - Using Task-Appropriate Policies is possible

Motivation

- Learning for high-dimensional robots is difficult:
  - Limit of Value Functions: fill-up state-space
  - Limit of Model Learning: accurate model!
- Starting with expert's knowledge helps!
  - Improving upon Demonstrations
  - Using Task-Appropriate Policies is possible
- Exploring on the real system?

Motivation

- Learning for high-dimensional robots is difficult:
  - Limit of Value Functions: fill-up state-space
  - Limit of Model Learning: accurate model!
- Starting with expert's knowledge helps!
  - Improving upon Demonstrations
  - Using Task-Appropriate Policies is possible
- Exploring on the real system?

➡Parametric Policy Search methods can do all that!

Bigger Picture

Reinforcement Learning Data

$$\mathscr{D} = \{(\mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_{i+1}, r_i)\}\}$$

$$\mathbf{x}' \sim \mathscr{P}^{\mathbf{u}}_{\mathbf{x}\mathbf{x}'}$$
$$r \approx \mathscr{R}^{\mathbf{u}}_{\mathbf{x}\mathbf{x}'}$$
$$\downarrow$$
$$V^*(\mathbf{x})$$
$$\downarrow$$
$$\pi^*(\mathbf{u}|\mathbf{x})$$

Optimal Control with Model Learning

$$V^*(\mathbf{x})$$
$$\downarrow$$
$$\pi^*(\mathbf{u}|\mathbf{x})$$

Value Function Methods

$$\pi^*(\mathbf{u}|\mathbf{x})$$

Policy Search

Now

Outline of the Lecture

Outline of the Lecture

1. Introduction with Policy Gradients

Outline of the Lecture

1. Introduction with Policy Gradients

2. Recent Advances in Policy Gradients

Outline of the Lecture

1. Introduction with Policy Gradients

2. Recent Advances in Policy Gradients

3. Probabilistic Policy Search with EM-like Approaches

Outline of the Lecture

1. Introduction with Policy Gradients

2. Recent Advances in Policy Gradients

3. Probabilistic Policy Search with EM-like Approaches

4. Conclusion

Outline of the Lecture

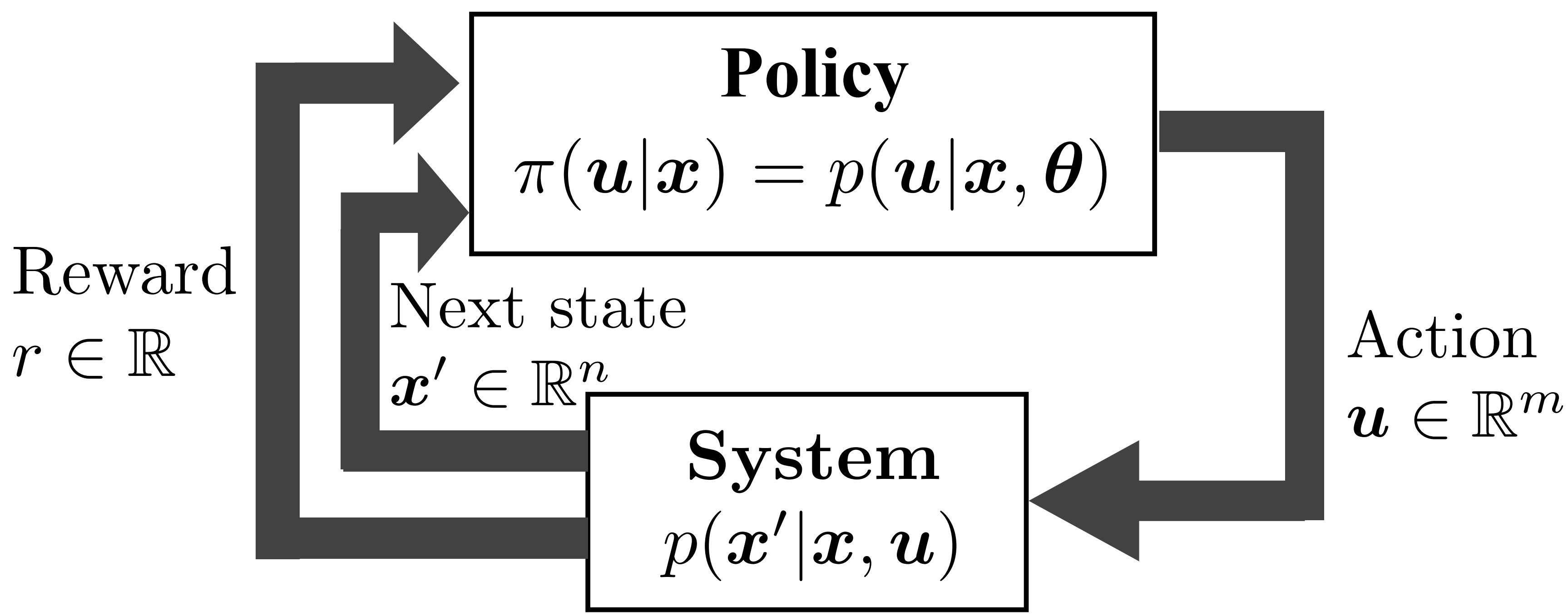
1. Introduction with Policy Gradients

2. Recent Advances in Policy Gradients

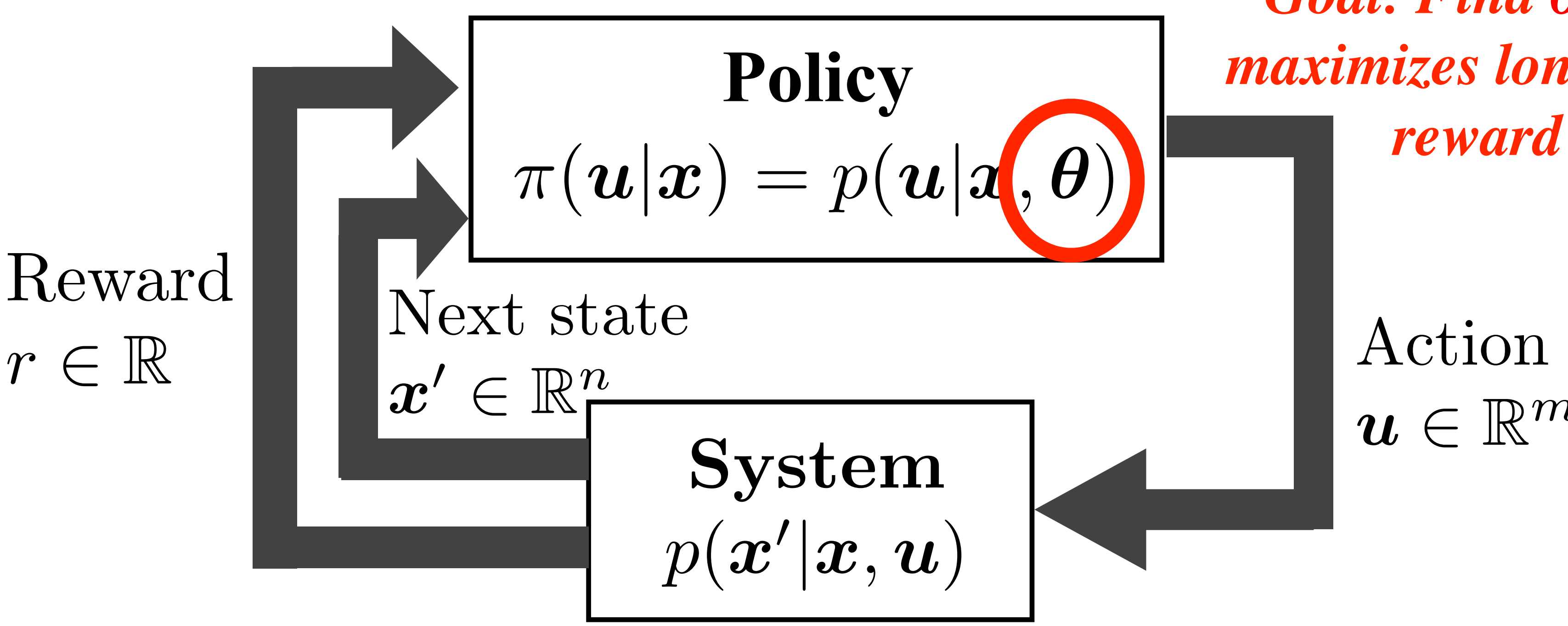
3. Probabilistic Policy Search with EM-like Approaches

4. Conclusion

Basics & Notation



Basics & Notation



Goal: Find $\boldsymbol{\theta}$ that maximizes long term reward

Generic Reinforcement Learning Loop

- Learning requires an iteration through Policy Evaluation and Policy Improvement.

Generic Reinforcement Learning Loop

- Learning requires an iteration through Policy Evaluation and Policy Improvement.

Critic: Policy Evaluation

$$Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) = E\left\{ \sum_{t=0}^{\infty} \gamma^t r_t \,\middle|\, \boldsymbol{x}, \boldsymbol{u} \right\}$$

$$V^{\pi}(\boldsymbol{x}) = E\left\{ \sum_{t=0}^{\infty} \gamma^t r_t \,\middle|\, \boldsymbol{x} \right\}$$

Generic Reinforcement Learning Loop

- Learning requires an iteration through Policy Evaluation and Policy Improvement.

Critic: Policy Evaluation

$$Q^{\pi}(\boldsymbol{x},\boldsymbol{u}) = E\left\{ \sum_{t=0}^{\infty} \gamma^t r_t \,\middle|\, \boldsymbol{x},\boldsymbol{u} \right\}$$

$$V^{\pi}(\boldsymbol{x}) = E\left\{ \sum_{t=0}^{\infty} \gamma^t r_t \,\middle|\, \boldsymbol{x} \right\}$$

Generic Reinforcement Learning Loop

- Learning requires an iteration through Policy Evaluation and Policy Improvement.

Critic: Policy Evaluation

$$Q^\pi(\boldsymbol{x}, \boldsymbol{u}) = E\left\{ \sum_{t=0}^{\infty} \gamma^t r_t \,\middle|\, \boldsymbol{x}, \boldsymbol{u} \right\}$$

$$V^\pi(\boldsymbol{x}) = E\left\{ \sum_{t=0}^{\infty} \gamma^t r_t \,\middle|\, \boldsymbol{x} \right\}$$

→

Actor: Policy Improvement

$$\pi \longrightarrow \pi'$$

Generic Reinforcement Learning Loop

- Learning requires an iteration through Policy Evaluation and Policy Improvement.

Critic: Policy Evaluation

$$Q^{\pi}(\boldsymbol{x},\boldsymbol{u}) = E\left\{ \sum_{t=0}^{\infty} \gamma^t r_t \,\middle|\, \boldsymbol{x},\boldsymbol{u} \right\}$$

$$V^{\pi}(\boldsymbol{x}) = E\left\{ \sum_{t=0}^{\infty} \gamma^t r_t \,\middle|\, \boldsymbol{x} \right\}$$

Actor: Policy Improvement

$$\pi \longrightarrow \pi'$$

Generic Reinforcement Learning Loop

- Learning requires an iteration through Policy Evaluation and Policy Improvement.

Critic: Policy Evaluation

$$Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) = E\left\{ \sum_{t=0}^{\infty} \gamma^t r_t \,\middle|\, \boldsymbol{x}, \boldsymbol{u} \right\}$$

$$V^{\pi}(\boldsymbol{x}) = E\left\{ \sum_{t=0}^{\infty} \gamma^t r_t \,\middle|\, \boldsymbol{x} \right\}$$

Requires Function Approximation

Actor: Policy Improvement

$$\pi \longrightarrow \pi'$$

Greedy vs Incremental

Greedy vs Incremental

Greedy Updates:

$$\boldsymbol{\theta}_{\pi'} = \operatorname{argmax}_{\tilde{\boldsymbol{\theta}}} E_{\pi_{\tilde{\theta}}} \left\{ Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) \right\}$$

Greedy vs Incremental

Greedy Updates:

$$\boldsymbol{\theta}_{\pi'} = \mathrm{argmax}_{\tilde{\boldsymbol{\theta}}} E_{\pi_{\tilde{\theta}}} \left\{ Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) \right\}$$

Policy Gradient Updates:

$$\boldsymbol{\theta}_{\pi'} = \boldsymbol{\theta}_{\pi} + \alpha \left. \frac{dJ(\boldsymbol{\theta})}{d\boldsymbol{\theta}} \right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{\pi}}$$

!

Greedy vs Incremental

Greedy Updates:

$$\boldsymbol{\theta}_{\pi'} = \operatorname{argmax}_{\tilde{\boldsymbol{\theta}}} E_{\pi_{\tilde{\theta}}} \left\{ Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) \right\}$$

V^{π}

Small change

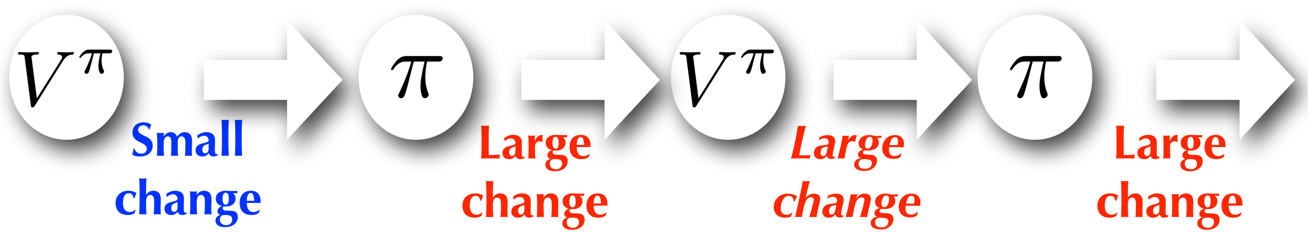
Policy Gradient Updates:

$$\boldsymbol{\theta}_{\pi'} = \boldsymbol{\theta}_{\pi} + \alpha \left. \frac{dJ(\boldsymbol{\theta})}{d\boldsymbol{\theta}} \right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{\pi}}$$

Greedy vs Incremental

Greedy Updates:

$$\boldsymbol{\theta}_{\pi'} = \operatorname{argmax}_{\tilde{\boldsymbol{\theta}}} E_{\pi_{\tilde{\theta}}} \{ Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) \}$$



Policy Gradient Updates:

$$\boldsymbol{\theta}_{\pi'} = \boldsymbol{\theta}_{\pi} + \alpha \left. \frac{dJ(\boldsymbol{\theta})}{d\boldsymbol{\theta}} \right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{\pi}}$$

Greedy vs Incremental

Greedy Updates:

$$\boldsymbol{\theta}_{\pi'} = \operatorname{argmax}_{\tilde{\boldsymbol{\theta}}} E_{\pi_{\tilde{\theta}}} \left\{ Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) \right\}$$

$$V^{\pi} \xrightarrow{\text{Small change}} \pi \xrightarrow{\text{Large change}} V^{\pi} \xrightarrow{\text{Large change}} \pi \xrightarrow{\text{Large change}}$$

potentially unstable learning process with large policy jumps

Policy Gradient Updates:

$$\boldsymbol{\theta}_{\pi'} = \boldsymbol{\theta}_{\pi} + \alpha \left. \frac{dJ(\boldsymbol{\theta})}{d\boldsymbol{\theta}} \right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{\pi}}$$

!

Greedy vs Incremental

Greedy Updates:

$$\boldsymbol{\theta}_{\pi'} = \operatorname{argmax}_{\tilde{\boldsymbol{\theta}}} E_{\pi_{\tilde{\theta}}} \left\{ Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) \right\}$$

V^{π} → π → V^{π} → π →

Small change | Large change | *Large change* | Large change

potentially unstable learning process with large policy jumps

Policy Gradient Updates:

$$\boldsymbol{\theta}_{\pi'} = \boldsymbol{\theta}_{\pi} + \alpha \left. \frac{dJ(\boldsymbol{\theta})}{d\boldsymbol{\theta}} \right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{\pi}}$$

V^{π} →

Small change

!

Greedy vs Incremental

Greedy Updates:

$$\boldsymbol{\theta}_{\pi'} = \operatorname{argmax}_{\tilde{\boldsymbol{\theta}}} E_{\pi_{\tilde{\theta}}} \left\{ Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) \right\}$$

$$V^{\pi} \xrightarrow{\text{Small change}} \pi \xrightarrow{\text{Large change}} V^{\pi} \xrightarrow{\textit{Large change}} \pi \xrightarrow{\text{Large change}}$$

potentially unstable learning process with large policy jumps

Policy Gradient Updates:

$$\boldsymbol{\theta}_{\pi'} = \boldsymbol{\theta}_{\pi} + \alpha \left. \frac{dJ(\boldsymbol{\theta})}{d\boldsymbol{\theta}} \right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{\pi}}$$

$$V^{\pi} \xrightarrow{\text{Small change}} \pi \xrightarrow{\text{Small change}} V^{\pi} \xrightarrow{\text{Small change}} \pi \xrightarrow{\text{Small change}}$$

!

Greedy vs Incremental

Greedy Updates:

$$\boldsymbol{\theta}_{\pi'} = \operatorname{argmax}_{\tilde{\boldsymbol{\theta}}} E_{\pi_{\tilde{\theta}}} \left\{ Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) \right\}$$

$$V^{\pi} \xrightarrow[\text{Small change}]{} \pi \xrightarrow[\text{Large change}]{} V^{\pi} \xrightarrow[\text{Large change}]{} \pi \xrightarrow[\text{Large change}]{}$$

potentially unstable learning process with large policy jumps

Policy Gradient Updates:

$$\boldsymbol{\theta}_{\pi'} = \boldsymbol{\theta}_{\pi} + \alpha \left. \frac{dJ(\boldsymbol{\theta})}{d\boldsymbol{\theta}} \right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{\pi}}$$

$$V^{\pi} \xrightarrow[\text{Small change}]{} \pi \xrightarrow[\text{Small change}]{} V^{\pi} \xrightarrow[\text{Small change}]{} \pi \xrightarrow[\text{Small change}]{}$$

stable learning process with smooth policy improvement

!

Objective Function

- Goal: Optimize the expected return

Objective Function

- Goal: Optimize the expected return

$$J(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) r(\boldsymbol{x}, \boldsymbol{u}) d\boldsymbol{u} d\boldsymbol{x},$$

Objective Function

- Goal: Optimize the expected return

$$J(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) r(\boldsymbol{x}, \boldsymbol{u}) d\boldsymbol{u} d\boldsymbol{x},$$

State distribution

Objective Function

- Goal: Optimize the expected return

$$J(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) r(\boldsymbol{x}, \boldsymbol{u}) d\boldsymbol{u} d\boldsymbol{x},$$

State distribution

Policy
(we can choose it)

Objective Function

- Goal: Optimize the expected return

$$J(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) r(\boldsymbol{x}, \boldsymbol{u}) d\boldsymbol{u} d\boldsymbol{x},$$

State distribution

Policy
(we can choose it)

Reward

Objective Function

- Goal: Optimize the expected return

$$J(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) r(\boldsymbol{x}, \boldsymbol{u}) d\boldsymbol{u} d\boldsymbol{x},$$

State distribution

Policy
(we can choose it)

Reward

$$= E\left\{\sum_{t=0}^{\infty} \gamma^t r_t\right\}$$

Gradient-based Policy Iteration

Actor: Policy Evaluation

Estimate Gradient

$$\mathbf{g}_t = \boldsymbol{\nabla} J(\boldsymbol{\theta})$$

Gradient-based Policy Iteration

Actor: Policy Evaluation

Estimate Gradient

$$\mathbf{g}_t = \boldsymbol{\nabla} J(\boldsymbol{\theta})$$

Critic: Policy Improvement

Update Parameters

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t \mathbf{g}_t$$

Policy Gradient Methods

!

Many related approaches exist in the literature, e.g., Mean-Value Differentiation, Model-based approaches, DDP, Frequency-based approaches, etc.

Policy Gradient Methods

- Finite Difference Methods (FD)
- Likelihood Ratio Methods
  - 'Vanilla' Policy Gradients (VPG)
  - 'Natural' Policy Gradients (NPG)

Many related approaches exist in the literature, e.g., Mean-Value Differentiation, Model-based approaches, DDP, Frequency-based approaches, etc.

!

Black-Box Approaches

A large class of algorithms includes Kiefer-Wolfowitz procedure, Robbins-Monroe, Simultaneous Perturbation Stochastic Approximation SPSA, ...

Black-Box Approaches

1. Perturb the parameters of your policy:

$$\theta + \delta\theta$$

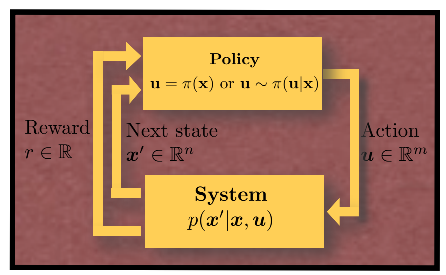
A large class of algorithms includes Kiefer-Wolfowitz procedure, Robbins-Monroe, Simultaneous Perturbation Stochastic Approximation SPSA, ...

!

Black-Box Approaches

1. Perturb the parameters of your policy:

$$\theta + \delta\theta$$



$$J(\theta + \delta\theta) - J(\theta)$$

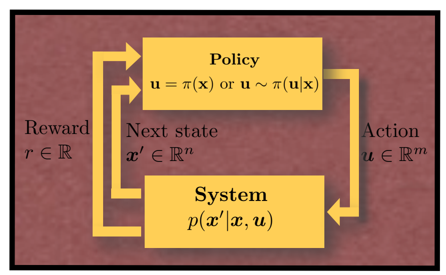
A large class of algorithms includes Kiefer-Wolfowitz procedure, Robbins-Monroe, Simultaneous Perturbation Stochastic Approximation SPSA, ...

!

Black-Box Approaches

1. Perturb the parameters of your policy:

$$\theta + \delta\theta$$



$$J(\theta + \delta\theta) - J(\theta)$$

2. Gradient estimation by regression:

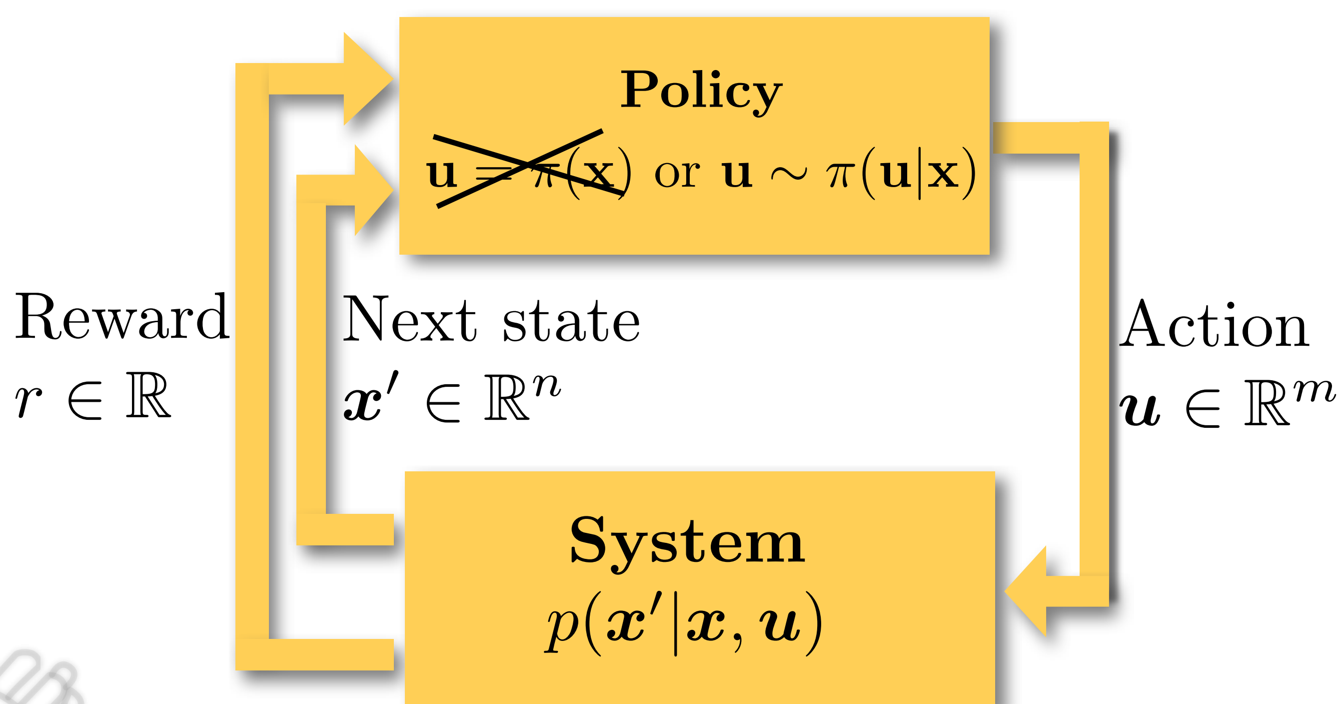
$$\mathbf{g}_{\mathrm{FD}} = \left(\boldsymbol{\Delta\Theta}^T \boldsymbol{\Delta\Theta}\right)^{-1} \boldsymbol{\Delta\Theta}^T \boldsymbol{\Delta} J.$$

A large class of algorithms includes Kiefer-Wolfowitz procedure, Robbins-Monroe, Simultaneous Perturbation Stochastic Approximation SPSA, ...

!

Whitebox Approaches

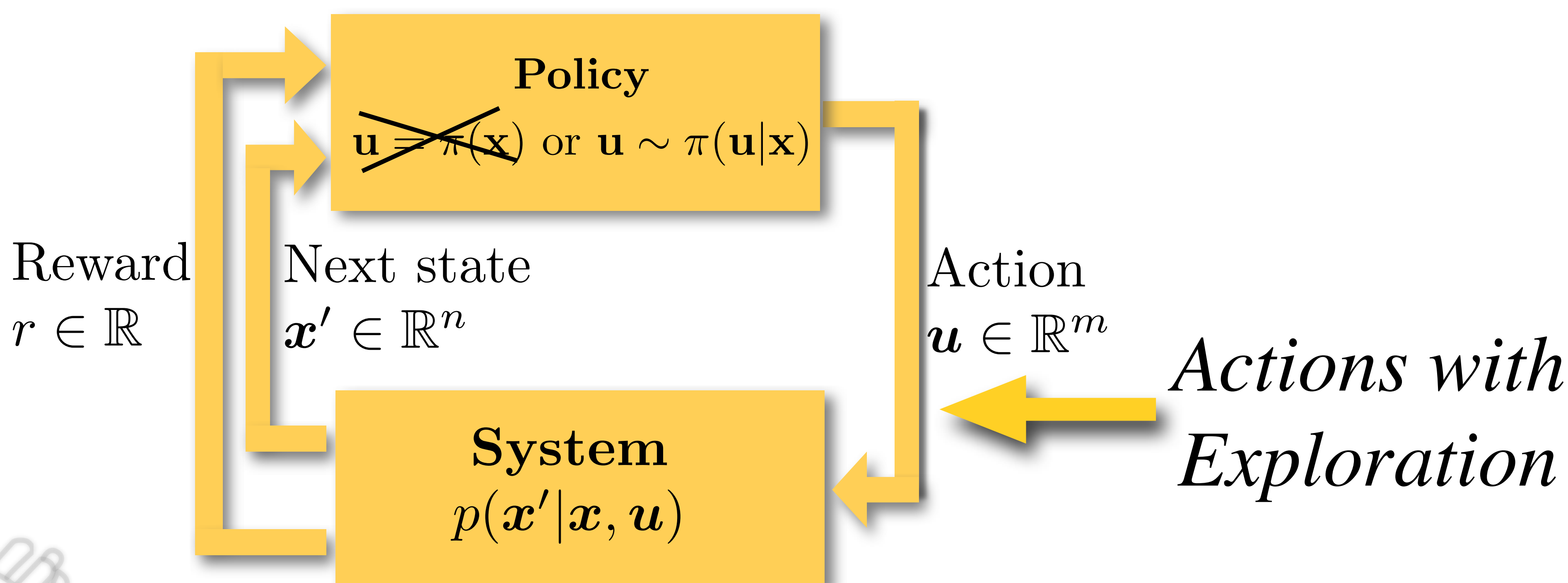
Whitebox Approach: Use a explorative, stochastic policy and make use of the knowledge of your policy.



Many related approaches in the RL literature starting from Werbos (1971), Hasdorff (1976), Williams (1988), ...

Whitebox Approaches

Whitebox Approach: Use a explorative, stochastic policy and make use of the knowledge of your policy.



Many related approaches in the RL literature starting from Werbos (1971), Hasdorff (1976), Williams (1988), ...

!

Likelihood Ratio Gradient

For a cost function

$$J(\theta) = \int_{\mathbb{T}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}|\pi) R(\boldsymbol{\tau}) d\boldsymbol{\tau}$$

we have the gradient

$$\nabla J(\theta) = \nabla \int_{\mathbb{T}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}|\pi) R(\boldsymbol{\tau}) d\boldsymbol{\tau} = \int_{\mathbb{T}} \nabla p_{\boldsymbol{\theta}}(\boldsymbol{\tau}|\pi) R(\boldsymbol{\tau}) d\boldsymbol{\tau}$$

Using the trick

$$\nabla p_{\boldsymbol{\theta}}(\boldsymbol{\tau}|\pi) = p_{\boldsymbol{\theta}}(\boldsymbol{\tau}|\pi) \nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau}|\pi)$$

we obtain

$$\begin{aligned}
\nabla J(\theta) &= \int_{\mathbb{T}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}|\pi) \nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau}|\pi) R(\boldsymbol{\tau}) d\boldsymbol{\tau} \\
&= E\{\nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau}|\pi) R(\boldsymbol{\tau})\} \\
&\approx \frac{1}{K} \sum_{k=1}^{K} \nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{\tau}_k|\pi) R(\boldsymbol{\tau}_k)
\end{aligned}$$

Needs only samples!

!

Likelihood Ratio Gradient

Why is this cool?

Because: The definition of a path probability

$$p(\boldsymbol{\tau}) = p(\mathbf{x}_1)\prod_{t=1}^{T} p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)\pi(\mathbf{u}_t|\mathbf{x}_t)$$

implies

$$\log p(\boldsymbol{\tau}) = \sum_{t=1}^{T} \log \pi(\mathbf{u}_t|\mathbf{x}_t) + \mathrm{const}$$

Hence, we can get the derivative of the distribution without a model of the system:

$$\nabla \log p(\boldsymbol{\tau}) = \sum_{t=1}^{T} \nabla \log \pi(\mathbf{u}_t|\mathbf{x}_t)$$

Likelihood Ratio Gradient

As a result:

$$
\begin{aligned}
\nabla J(\theta) &= E\left\{\sum\nolimits_{t=1}^{T} \nabla \log \pi(\mathbf{u}_t|\mathbf{x}_t) R(\boldsymbol{\tau})\right\} \\
&= E\left\{\sum\nolimits_{t=1}^{T} \nabla \log \pi(\mathbf{u}_t|\mathbf{x}_t) \sum\nolimits_{h=t}^{T} r(\mathbf{x}_t, \mathbf{u}_t)\right\} \\
&= E\left\{\sum\nolimits_{t=1}^{T} \nabla \log \pi(\mathbf{u}_t|\mathbf{x}_t) Q^{\pi}(\mathbf{x}_t, \mathbf{u}_t)\right\}
\end{aligned}
$$

Outline of the Lecture

1. Introduction with Policy Gradients

2. Recent Advances in Policy Gradients

3. Probabilistic Policy Search with EM-like Approaches

4. Conclusion

Likelihood Ratio Approach: Policy Gradient Theorem

Originally discovered: Aleksandrov, 1968; Glynn, 1986
Examples: episodic REINFORCE, SRV, GPOMDP

Likelihood Ratio Approach: Policy Gradient Theorem

According to the policy gradient theorem, the gradient can be computed as

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \nabla_{\boldsymbol{\theta}} \pi(\boldsymbol{u}|\boldsymbol{x})(Q^{\pi}(\boldsymbol{x},\boldsymbol{u}) - b^{\pi}(\boldsymbol{x})) d\boldsymbol{u} d\boldsymbol{x}.$$

Originally discovered: Aleksandrov, 1968; Glynn, 1986
Examples: episodic REINFORCE, SRV, GPOMDP

Likelihood Ratio Approach: Policy Gradient Theorem

According to the policy gradient theorem, the gradient can be computed as

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \nabla_{\boldsymbol{\theta}} \pi(\boldsymbol{u}|\boldsymbol{x})(Q^{\pi}(\boldsymbol{x},\boldsymbol{u}) - b^{\pi}(\boldsymbol{x})) d\boldsymbol{u} d\boldsymbol{x}.$$

Gradient of the expected return

Derivative only of the policy

State-action value function

Arbitrary baseline function

Originally discovered: Aleksandrov, 1968; Glynn, 1986
Examples: episodic REINFORCE, SRV, GPOMDP

Likelihood Ratio Approach: Policy Gradient Theorem

According to the policy gradient theorem, the gradient can be computed as

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \nabla_{\boldsymbol{\theta}} \pi(\boldsymbol{u}|\boldsymbol{x}) (Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) - b^{\pi}(\boldsymbol{x})) d\boldsymbol{u} d\boldsymbol{x}.$$

Gradient of the expected return

Derivative only of the policy

State-action value function

Arbitrary baseline function

Problems: High Variance, dependence on the baseline, slow convergence!

Originally discovered: Aleksandrov, 1968; Glynn, 1986
Examples: episodic REINFORCE, SRV, GPOMDP

Compatible Function Approximation

(Sutton et al., 2000; Konda & Tsitsiklis, 2000)

Compatible Function Approximation

The state-action value function can be replaced by

$$Q^{\pi}(\boldsymbol{x},\boldsymbol{u}) \equiv f^{\pi}_{\boldsymbol{w}}(\boldsymbol{x},\boldsymbol{u}) = \frac{d\log\pi(\boldsymbol{u}|\boldsymbol{x})}{d\boldsymbol{\theta}}^{T}\boldsymbol{w}$$

without biasing the gradient.

(Sutton et al., 2000; Konda & Tsitsiklis, 2000)

Compatible Function Approximation

The state-action value function can be replaced by

$$Q^{\pi}(\boldsymbol{x},\boldsymbol{u}) \equiv f_{\boldsymbol{w}}^{\pi}(\boldsymbol{x},\boldsymbol{u}) = \frac{d\log\pi(\boldsymbol{u}|\boldsymbol{x})}{d\boldsymbol{\theta}}^{T}\boldsymbol{w}$$

State-action value function

Compatible function approximation

Log-policy derivative

Parameters of the function approximator

without biasing the gradient.

(Sutton et al., 2000; Konda & Tsitsiklis, 2000)

Compatible Function Approximation

The state-action value function can be replaced by

$$Q^{\pi}(\boldsymbol{x},\boldsymbol{u}) \equiv f^{\pi}_{\boldsymbol{w}}(\boldsymbol{x},\boldsymbol{u}) = \frac{d\log\pi(\boldsymbol{u}|\boldsymbol{x})}{d\boldsymbol{\theta}}^{T}\boldsymbol{w}$$

State-action value function

Compatible function approximation

Log-policy derivative

Parameters of the function approximator

without biasing the gradient.

Thus, the gradient becomes

$$\boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \boldsymbol{\nabla}_{\boldsymbol{\theta}} \pi(\boldsymbol{u}|\boldsymbol{x}) (f^{\pi}_{\boldsymbol{w}}(\boldsymbol{x},\boldsymbol{u}) - b^{\pi}(\boldsymbol{x})) d\boldsymbol{u} d\boldsymbol{x}.$$

(Sutton et al., 2000; Konda & Tsitsiklis, 2000)

All-Action Gradient

(Peters et al. 2003, 2005)

All-Action Gradient

By integrating over all possible actions in a state, the baseline can be integrated out, and the gradient becomes

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \nabla_{\boldsymbol{\theta}} \pi(\boldsymbol{u}|\boldsymbol{x})(f_w^{\pi}(\boldsymbol{x},\boldsymbol{u}) - b(\boldsymbol{x}))d\boldsymbol{u}d\boldsymbol{x}, \\ &= \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) \nabla_{\boldsymbol{\theta}} \log\pi(\boldsymbol{u}|\boldsymbol{x}) \nabla_{\boldsymbol{\theta}} \log\pi(\boldsymbol{u}|\boldsymbol{x})^T \boldsymbol{w} d\boldsymbol{u}d\boldsymbol{x}, \\ &= \boldsymbol{F}(\boldsymbol{\theta})\boldsymbol{w}.\end{aligned}$$

(Peters et al. 2003, 2005)

All-Action Gradient

By integrating over all possible actions in a state, the baseline can be integrated out, and the gradient becomes

$$\begin{aligned}
\boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \boldsymbol{\nabla}_{\boldsymbol{\theta}} \pi(\boldsymbol{u}|\boldsymbol{x})(f_{w}^{\pi}(\boldsymbol{x},\boldsymbol{u}) - b(\boldsymbol{x}))d\boldsymbol{u}d\boldsymbol{x}, \\
&= \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x})^{T} \boldsymbol{w} d\boldsymbol{u}d\boldsymbol{x}, \\
&= \boldsymbol{F}(\boldsymbol{\theta})\boldsymbol{w}.
\end{aligned}$$

(Peters et al. 2003, 2005)

All-Action Gradient

By integrating over all possible actions in a state, the baseline can be integrated out, and the gradient becomes

$$
\begin{aligned}
\boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \boldsymbol{\nabla}_{\boldsymbol{\theta}} \pi(\boldsymbol{u}|\boldsymbol{x}) (f_w^{\pi}(\boldsymbol{x}, \boldsymbol{u}) - b(\boldsymbol{x})) d\boldsymbol{u} d\boldsymbol{x}, \\
&= \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x})^T \boldsymbol{w} d\boldsymbol{u} d\boldsymbol{x}, \\
&= \boldsymbol{F}(\boldsymbol{\theta}) \boldsymbol{w}.
\end{aligned}
$$

All Action Matrix

Parameters

(Peters et al. 2003, 2005)

Natural Gradients

(Amari, 1998)

Natural Gradients

A more efficient gradient in learning problems is the natural gradient (Amari, 1998)

$$\tilde{\boldsymbol{\nabla}}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = G^{-1}(\boldsymbol{\theta}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

(Amari, 1998)

Natural Gradients

A more efficient gradient in learning problems is the natural gradient (Amari, 1998)

Natural gradient

$$\tilde{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = G^{-1}(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

Inverse of the Fisher Information Matrix

'Vanilla` gradient

(Amari, 1998)

Natural Gradients

A more efficient gradient in learning problems is the natural gradient (Amari, 1998)

Natural gradient

$$\tilde{\boldsymbol{\nabla}}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = G^{-1}(\boldsymbol{\theta}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

Inverse of the Fisher Information Matrix

'Vanilla` gradient

where the policy gradient $\boldsymbol{\nabla} J(\boldsymbol{\theta})$ is given by the policy gradient theorem.

(Amari, 1998)

Natural Gradients

A more efficient gradient in learning problems is the natural gradient (Amari, 1998)

Natural gradient

$$\tilde{\boldsymbol{\nabla}}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = G^{-1}(\boldsymbol{\theta}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

Inverse of the Fisher Information Matrix

'Vanilla` gradient

where the policy gradient $\boldsymbol{\nabla} J(\boldsymbol{\theta})$ is given by the policy gradient theorem.

But how can we obtain the Fisher information matrix $G(\boldsymbol{\theta})$??

(Amari, 1998)

Fisher Information

So how does the All-Action Matrix

(Peters et al., 2003; 2005; Bagnell et al., 2003)

Fisher Information

So how does the All-Action Matrix

$$\boldsymbol{F}(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) d\boldsymbol{u} d\boldsymbol{x}.$$

(Peters et al., 2003; 2005; Bagnell et al., 2003)

Fisher Information

So how does the All-Action Matrix

$$\boldsymbol{F}(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) d\boldsymbol{u} d\boldsymbol{x}.$$

relate to the Fisher Information Matrix

(Peters et al., 2003; 2005; Bagnell et al., 2003)

Fisher Information

So how does the All-Action Matrix

$$\boldsymbol{F}(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) d\boldsymbol{u} d\boldsymbol{x}.$$

relate to the Fisher Information Matrix

$$\boldsymbol{G}(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \left(d^{\pi}(\boldsymbol{x}) \pi(\boldsymbol{u}|\boldsymbol{x})\right) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \left(d^{\pi}(\boldsymbol{x}) \pi(\boldsymbol{u}|\boldsymbol{x})\right) d\boldsymbol{u} d\boldsymbol{x}.$$

(Peters et al., 2003; 2005; Bagnell et al., 2003)

Fisher Information

So how does the All-Action Matrix

$$\boldsymbol{F}(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) d\boldsymbol{u} d\boldsymbol{x}.$$

relate to the Fisher Information Matrix

$$\boldsymbol{G}(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \left(d^{\pi}(\boldsymbol{x}) \pi(\boldsymbol{u}|\boldsymbol{x})\right) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \left(d^{\pi}(\boldsymbol{x}) \pi(\boldsymbol{u}|\boldsymbol{x})\right) d\boldsymbol{u} d\boldsymbol{x}.$$

While Kakade (2002) suggested that **F** is an ‘average of point Fisher information matrices’, we could prove that

(Peters et al., 2003; 2005; Bagnell et al., 2003)

Fisher Information

So how does the All-Action Matrix

$$\boldsymbol{F}(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}|\boldsymbol{x}) d\boldsymbol{u} d\boldsymbol{x}.$$

relate to the Fisher Information Matrix

$$\boldsymbol{G}(\boldsymbol{\theta}) = \int_{\mathbb{X}} d^{\pi}(\boldsymbol{x}) \int_{\mathbb{U}} \pi(\boldsymbol{u}|\boldsymbol{x}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \left(d^{\pi}(\boldsymbol{x}) \pi(\boldsymbol{u}|\boldsymbol{x})\right) \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \left(d^{\pi}(\boldsymbol{x}) \pi(\boldsymbol{u}|\boldsymbol{x})\right) d\boldsymbol{u} d\boldsymbol{x}.$$

While Kakade (2002) suggested that **F** is an ‘average of point Fisher information matrices’, we could prove that

$$\boxed{\mathbf{F} = \mathbf{G}.}$$

(Peters et al., 2003; 2005; Bagnell et al., 2003)

Natural Policy Gradients

(Kakade, 2002; Peters et al. 2003, 2005; Bagnell & Schneider, 2003)

Natural Policy Gradients

Thus, the gradient simplifies to

$$\boxed{\tilde{\boldsymbol{\nabla}}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \boldsymbol{G}^{-1}(\boldsymbol{\theta}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \boldsymbol{G}^{-1}(\boldsymbol{\theta}) \boldsymbol{F}(\boldsymbol{\theta}) \boldsymbol{w} = \boldsymbol{w},}$$

(Kakade, 2002; Peters et al. 2003, 2005; Bagnell & Schneider, 2003)

Natural Policy Gradients

Thus, the gradient simplifies to

$$\boxed{\tilde{\boldsymbol{\nabla}}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \boldsymbol{G}^{-1}(\boldsymbol{\theta}) \boldsymbol{\nabla}_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \boldsymbol{G}^{-1}(\boldsymbol{\theta}) \boldsymbol{F}(\boldsymbol{\theta}) \boldsymbol{w} = \boldsymbol{w},}$$

and the policy parameter update becomes

$$\boxed{\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t \boldsymbol{w}_t.}$$

Important: The gradient estimation simplifies to determining the parameters of the compatible function approximation.

(Kakade, 2002; Peters et al. 2003, 2005; Bagnell & Schneider, 2003)

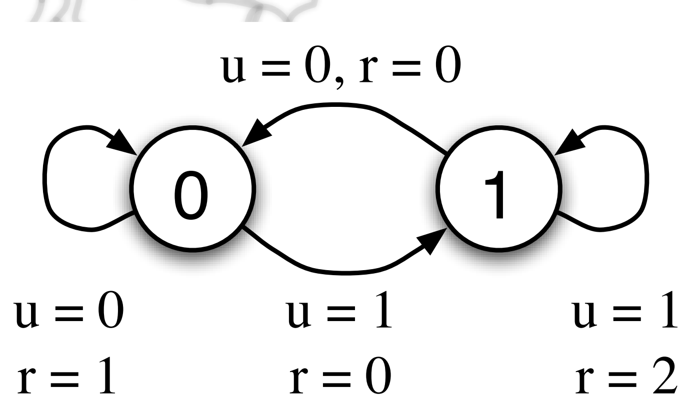
Are they useful?

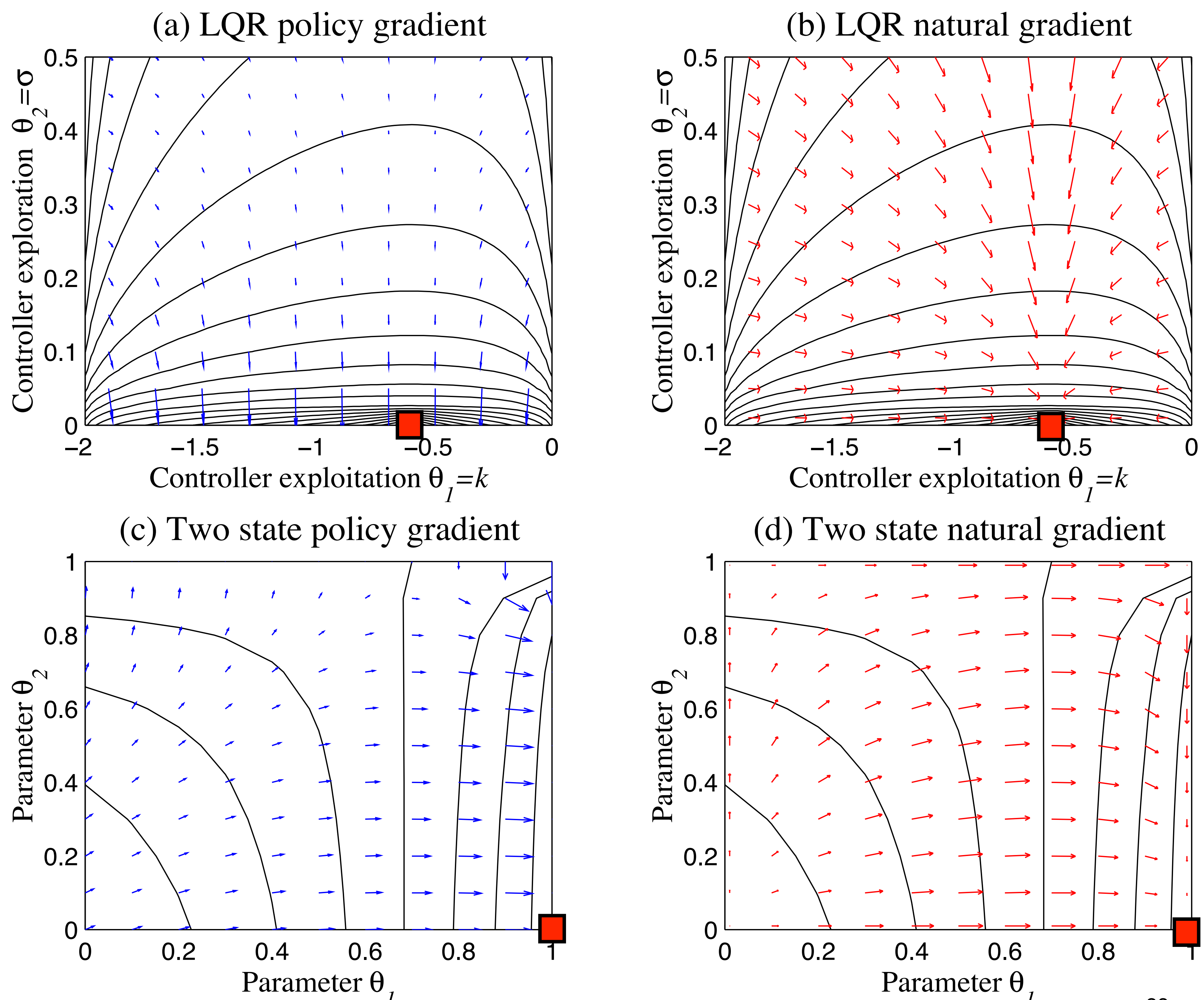
Are they useful?

Linear Quadratic Regulation

$$x_{t+1} = Ax_t + Bu_t$$
$$u_t \sim \pi(u|x_t) = \mathcal{N}(u|kx_t, \sigma)$$
$$r_t = -x_t^T Q x_t - u_t^T R u_t$$

Two-State Problem





(Peters et al. 2003, 2005)

Can the Compatible FA be learned?

(Peters et al. 2003, 2005)

Can the Compatible FA be learned?

The compatible function approximation is mean-zero! Thus, it can only represent the Advantage Function:

$$f^{\pi}_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{u}) = Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) - V^{\pi}(\boldsymbol{x}) = A^{\pi}(\boldsymbol{x}, \boldsymbol{u}).$$

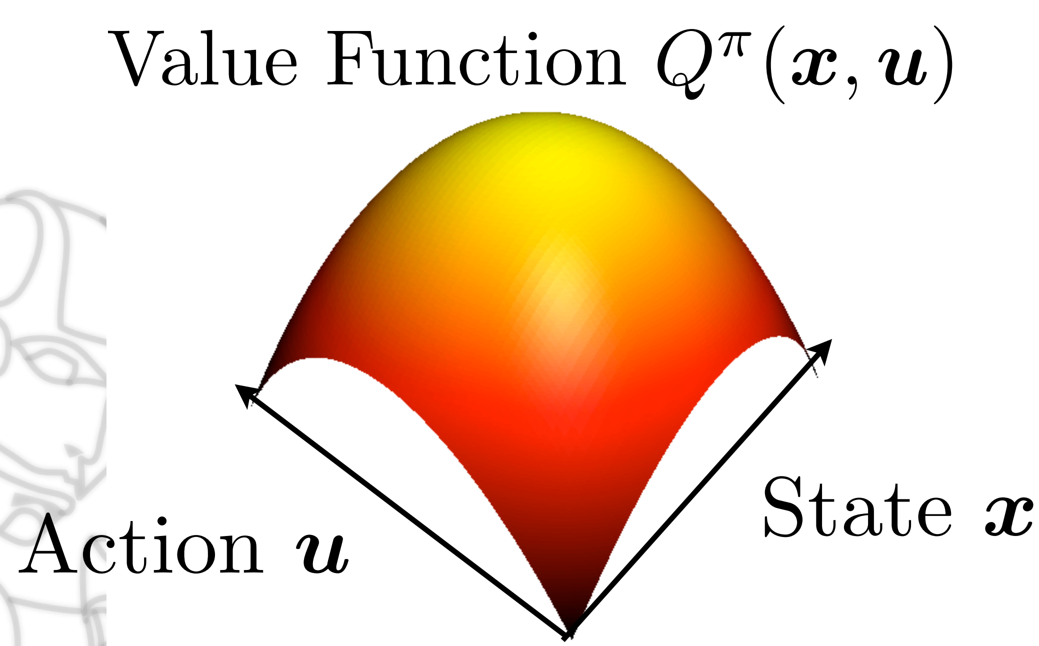
(Peters et al. 2003, 2005)

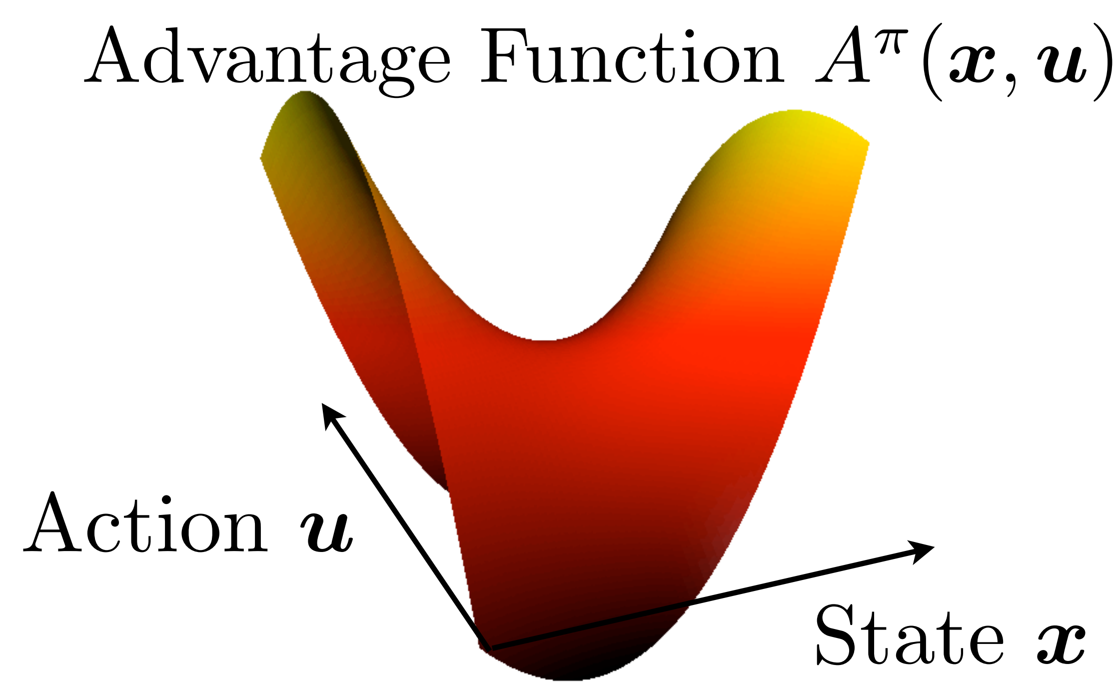
Can the Compatible FA be learned?

The compatible function approximation is mean-zero! Thus, it can only represent the Advantage Function:

$$f_{\boldsymbol{w}}^{\pi}(\boldsymbol{x},\boldsymbol{u}) = Q^{\pi}(\boldsymbol{x},\boldsymbol{u}) - V^{\pi}(\boldsymbol{x}) = A^{\pi}(\boldsymbol{x},\boldsymbol{u}).$$

The advantage function is very different from the value functions





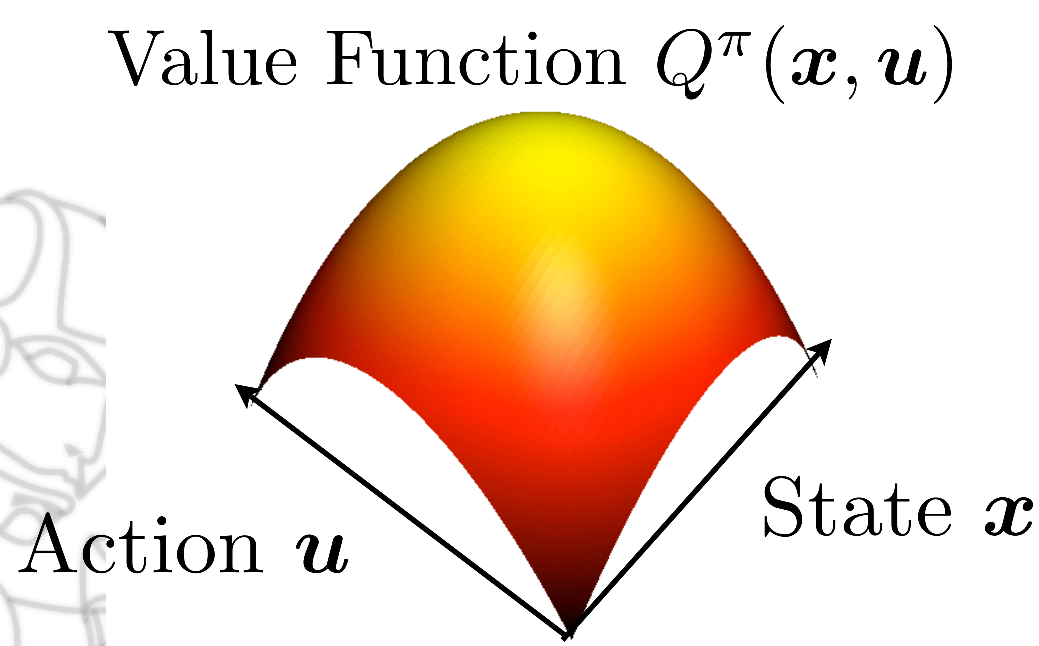
(Peters et al. 2003, 2005)

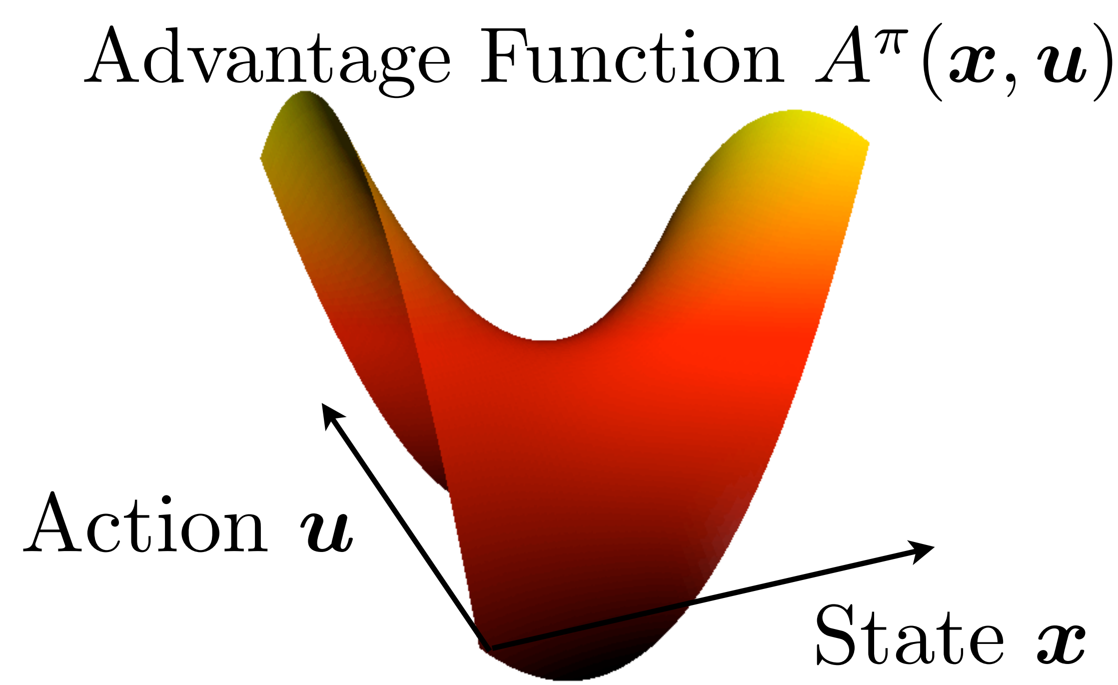
Can the Compatible FA be learned?

The compatible function approximation is mean-zero! Thus, it can only represent the Advantage Function:

$$f_{\boldsymbol{w}}^{\pi}(\boldsymbol{x}, \boldsymbol{u}) = Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) - V^{\pi}(\boldsymbol{x}) = A^{\pi}(\boldsymbol{x}, \boldsymbol{u}).$$

The advantage function is very different from the value functions





Traditional value function learning methods such as Temporal Difference learning cannot be applied.

(Peters et al. 2003, 2005)

The Compatible FA can be learned!

The Compatible FA can be learned!

We cannot apply traditional methods directly on

The Compatible FA can be learned!

We cannot apply traditional methods directly on

$$f^{\pi}_{\boldsymbol{w}}(\boldsymbol{x},\boldsymbol{u}) = Q^{\pi}(\boldsymbol{x},\boldsymbol{u}) - V^{\pi}(\boldsymbol{x}) = A^{\pi}(\boldsymbol{x},\boldsymbol{u}).$$

But when we add further function approximation

The Compatible FA can be learned!

We cannot apply traditional methods directly on

$$f^{\pi}_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{u}) = Q^{\pi}(\boldsymbol{x}, \boldsymbol{u}) - V^{\pi}(\boldsymbol{x}) = A^{\pi}(\boldsymbol{x}, \boldsymbol{u}).$$

But when we add further function approximation

$$V^{\pi}(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{v}$$

into the Bellman equation

The Compatible FA can be learned!

We cannot apply traditional methods directly on

$$f^{\pi}_{\boldsymbol{w}}(\boldsymbol{x},\boldsymbol{u}) = Q^{\pi}(\boldsymbol{x},\boldsymbol{u}) - V^{\pi}(\boldsymbol{x}) = A^{\pi}(\boldsymbol{x},\boldsymbol{u}).$$

But when we add further function approximation

$$V^{\pi}(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T\boldsymbol{v}$$

into the Bellman equation

$$V^{\pi}(\boldsymbol{x}_t) + \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)^T\boldsymbol{w} = r(\boldsymbol{x}_t,\boldsymbol{u}_t) + \gamma V^{\pi}(\boldsymbol{x}_{t+1}) + \epsilon_t$$

we get a **linear regression problem** which can be solved with appropriate regression techniques, e.g., Boyan’s (1996) LSTD(λ) algorithm.

The Compatible FA can be learned!

We cannot apply traditional methods directly on

$$f^{\pi}_{\boldsymbol{w}}(\boldsymbol{x},\boldsymbol{u}) = Q^{\pi}(\boldsymbol{x},\boldsymbol{u}) - V^{\pi}(\boldsymbol{x}) = A^{\pi}(\boldsymbol{x},\boldsymbol{u}).$$

But when we add further function approximation

$$V^{\pi}(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T\boldsymbol{v}$$

into the Bellman equation

$$V^{\pi}(\boldsymbol{x}_t) + \boldsymbol{\nabla}_{\boldsymbol{\theta}} \log \pi(\boldsymbol{u}_t|\boldsymbol{x}_t)^T\boldsymbol{w} = r(\boldsymbol{x}_t,\boldsymbol{u}_t) + \gamma V^{\pi}(\boldsymbol{x}_{t+1}) + \epsilon_t$$

we get a **linear regression problem** which can be solved with appropriate regression techniques, e.g., Boyan's (1996) LSTD(λ) algorithm.

➡ **Allows the derivation of many well-known old reinforcement learning algorithms, e.g., Sutton et al. (1983) Actor-Critic and Bradtke & Barto's (1993) LQR-Q-Learning.**

What about this additional FA?

(Peters et al. 2003, 2005)

What about this additional FA?

...but in many cases, we don't have a good additional function approximations!

(Peters et al. 2003, 2005)

What about this additional FA?

...but in many cases, we don't have a good additional function approximations!

For one rollout, if we sum up the Bellman Equations

(Peters et al. 2003, 2005)

What about this additional FA?

...but in many cases, we don't have a good additional function approximations!

For one rollout, if we sum up the Bellman Equations

and eliminate the values of the intermediary states, we obtain

(Peters et al. 2003, 2005)

What about this additional FA?

...but in many cases, we don't have a good additional function approximations!

For one rollout, if we sum up the Bellman Equations

$$
\begin{aligned}
V^\pi(\boldsymbol{x}_0) + \nabla \log \pi(\boldsymbol{u}_0|\boldsymbol{x}_0) &= r(\boldsymbol{x}_0, \boldsymbol{u}_0) + \gamma V^\pi(\boldsymbol{x}_1) \\
V^\pi(\boldsymbol{x}_1) + \nabla \log \pi(\boldsymbol{u}_1|\boldsymbol{x}_1) &= r(\boldsymbol{x}_1, \boldsymbol{u}_1) + \gamma V^\pi(\boldsymbol{x}_0) \\
&\vdots \\
V^\pi(\boldsymbol{x}_T) + \nabla \log \pi(\boldsymbol{u}_T|\boldsymbol{x}_T) &= r(\boldsymbol{x}_T, \boldsymbol{u}_T) + \gamma V^\pi(\boldsymbol{x}_{T+1})
\end{aligned}
$$

and eliminate the values of the intermediary states, we obtain

$$
\underbrace{V^\pi(\mathbf{x}_0)}_{J} + \underbrace{\left(\sum_{t=0}^{T} \nabla_\theta \log \pi(\mathbf{u}_t|\mathbf{x}_t) \right)^T}_{\varphi_i} \mathbf{w} = \underbrace{\sum_{t=0}^{T} \gamma^t r(\mathbf{x}_t, \mathbf{u}_t)}_{R_i} + \gamma^{T+1} \underbrace{V^\pi(\mathbf{x}_{T+1})}_{0}
$$

ONE offset parameter suffices as additional function approximation!

(Peters et al. 2003, 2005)

Episodic Natural Actor-Critic

Episodic Natural Actor-Critic

Critic: Episodic Evaluation

$$\mathbf{\Phi} = \begin{bmatrix} \varphi_1, & \varphi_2, & \ldots, & \varphi_N \\ 1, & 1, & \ldots, & 1 \end{bmatrix}^T$$

$$\mathbf{R} = \left[R_1, R_2^T, \ldots, R_N^T\right]^T$$

$$\begin{bmatrix} \boldsymbol{w} \\ J \end{bmatrix} = \left(\mathbf{\Phi}^T \mathbf{\Phi}\right)^{-1} \mathbf{\Phi}^T \boldsymbol{R}$$

Episodic Natural Actor-Critic

Critic: Episodic Evaluation

Sufficient Statistics

$$\mathbf{\Phi} = \begin{bmatrix} \varphi_1, & \varphi_2, & \ldots, & \varphi_N \\ 1, & 1, & \ldots, & 1 \end{bmatrix}^T$$

$$\mathbf{R} = \left[R_1, R_2^T, \ldots, R_N^T\right]^T$$

$$\begin{bmatrix} \boldsymbol{w} \\ J \end{bmatrix} = \left(\mathbf{\Phi}^T \mathbf{\Phi}\right)^{-1} \mathbf{\Phi}^T \boldsymbol{R}$$

Episodic Natural Actor-Critic

Critic: Episodic Evaluation

Sufficient Statistics

$$\mathbf{\Phi} = \begin{bmatrix} \varphi_1, & \varphi_2, & \ldots, & \varphi_N \\ 1, & 1, & \ldots, & 1 \end{bmatrix}^T$$

$$\mathbf{R} = \left[R_1, R_2^T, \ldots, R_N^T\right]^T$$

Linear Regression

$$\begin{bmatrix} \boldsymbol{w} \\ J \end{bmatrix} = \left(\mathbf{\Phi}^T \mathbf{\Phi}\right)^{-1} \mathbf{\Phi}^T \boldsymbol{R}$$

Episodic Natural Actor-Critic

Critic: Episodic Evaluation

Sufficient Statistics

$$\mathbf{\Phi} = \begin{bmatrix} \varphi_1, & \varphi_2, & \ldots, & \varphi_N \\ 1, & 1, & \ldots, & 1 \end{bmatrix}^T$$

$$\mathbf{R} = \left[R_1, R_2^T, \ldots, R_N^T\right]^T$$

Linear Regression

$$\begin{bmatrix} \boldsymbol{w} \\ J \end{bmatrix} = \left(\mathbf{\Phi}^T \mathbf{\Phi}\right)^{-1} \mathbf{\Phi}^T \boldsymbol{R}$$

Episodic Natural Actor-Critic

Critic: Episodic Evaluation

Sufficient Statistics

$$\mathbf{\Phi} = \begin{bmatrix} \varphi_1, & \varphi_2, & \ldots, & \varphi_N \\ 1, & 1, & \ldots, & 1 \end{bmatrix}^T$$

$$\mathbf{R} = \left[R_1, R_2^T, \ldots, R_N^T\right]^T$$

Linear Regression

$$\begin{bmatrix} \boldsymbol{w} \\ J \end{bmatrix} = \left(\mathbf{\Phi}^T \mathbf{\Phi}\right)^{-1} \mathbf{\Phi}^T \boldsymbol{R}$$

Actor: Natural Policy Gradient Improvement

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t \boldsymbol{w}_t.$$

Episodic Natural Actor-Critic

Critic: Episodic Evaluation

Sufficient Statistics

$$\mathbf{\Phi} = \begin{bmatrix} \varphi_1, & \varphi_2, & \ldots, & \varphi_N \\ 1, & 1, & \ldots, & 1 \end{bmatrix}^T$$

$$\mathbf{R} = \left[R_1, R_2^T, \ldots, R_N^T\right]^T$$

Linear Regression

$$\begin{bmatrix} \boldsymbol{w} \\ J \end{bmatrix} = \left(\mathbf{\Phi}^T \mathbf{\Phi}\right)^{-1} \mathbf{\Phi}^T \boldsymbol{R}$$

Actor: Natural Policy Gradient Improvement

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t \boldsymbol{w}_t.$$

Important Points

Important Points

Points worth highlighting:

Important Points

Points worth highlighting:

★ Natural policy gradients are ***independent*** of the chosen policy parameterization!

Important Points

Points worth highlighting:

★ Natural policy gradients are ***independent*** of the chosen policy parameterization!

★ They correspond to ***steepest descent in policy space*** and not in the parameter space.

Important Points

Points worth highlighting:

★ Natural policy gradients are ***independent*** of the chosen policy parameterization!

★ They correspond to ***steepest descent in policy space*** and not in the parameter space.

★ ***Convergence to a local minimum*** is guaranteed!

Important Points

Points worth highlighting:

★ Natural policy gradients are ***independent*** of the chosen policy parameterization!

★ They correspond to ***steepest descent in policy space*** and not in the parameter space.

★ ***Convergence to a local minimum*** is guaranteed!

? ...but we still need to estimate the natural gradient!

Important Points

Points worth highlighting:

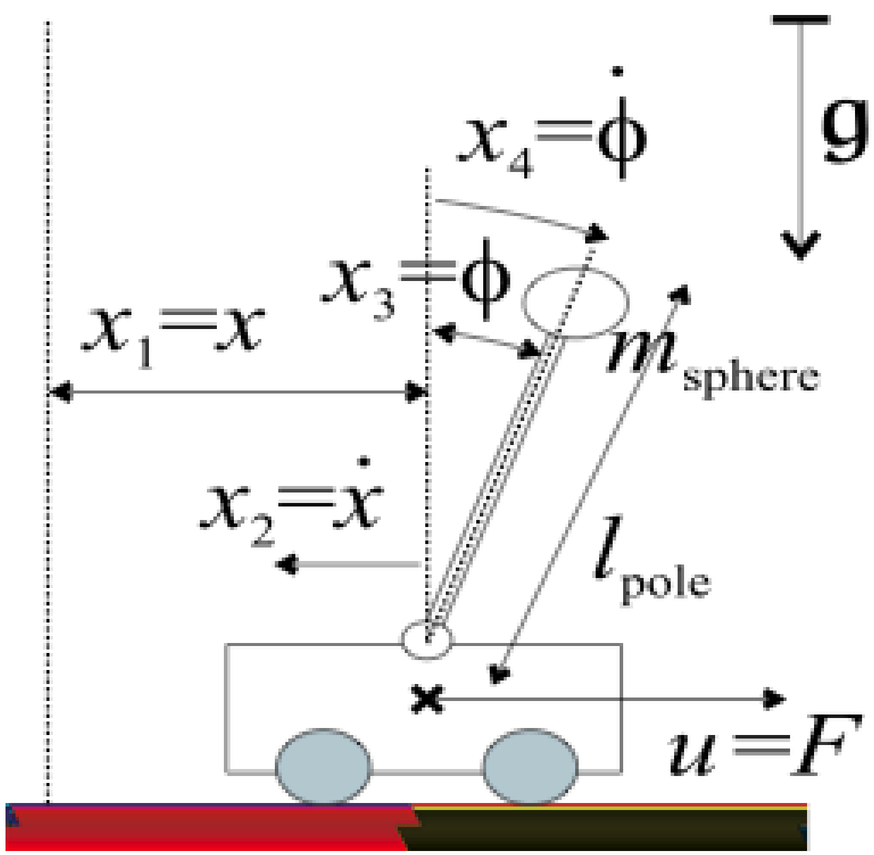
★ Natural policy gradients are ***independent*** of the chosen policy parameterization!

★ They correspond to ***steepest descent in policy space*** and not in the parameter space.

★ ***Convergence to a local minimum*** is guaranteed!

? ...but we still need to estimate the natural gradient!

Benchmarking on Cart-Pole Regulation



- standard benchmark
- maximize time inside the target area:

$$r(\mathbf{x},\mathbf{u}) = \begin{cases} 0 & \text{if } |\phi| < 0.05\text{rad}, |x| < 0.05\text{m} \\ -1 & \text{otherwise.} \end{cases}$$

- episodic restarts

Finite Difference Gradients

Algorithm	Fair performance (>-120) after	Good performance (>-80) after	best performance
Finite Difference Gradients with Standard Descent	12,300	Not reached	-84
Finite Difference Gradients with RPROP Rule	7,450	45,650	-76

Vanilla Policy Gradients

Algorithm	Fair performance (>-120) after	Good performance (>-80) after	best performance
Vanilla PG without Baseline	22,200	Not reached	-102
Vanilla PG with Optimal Baseline	1,200	26,450	-76
Vanilla PG with Optimal Baseline and RPROP	450	3,000	-64

Vanilla Policy Gradients

Algorithm	Fair performance (>-120) after	Good performance (>-80) after	best performance
Vanilla PG without Baseline	22,200	Not reached	-102
Vanilla PG with Optimal Baseline	1,200	26,450	-76
Vanilla PG with Optimal Baseline and RPROP	450	3,000	-64

Fastest Initial Improvement

Vanilla Policy Gradients

Algorithm	Fair performance (>-120) after	Good performance (>-80) after	best performance
Vanilla PG without Baseline	22,200	Not reached	-102
Vanilla PG with Optimal Baseline	1,200	26,450	-76
Vanilla PG with Optimal Baseline and RPROP	450	3,000	Not optimal -64

Fastest Initial Improvement

Episodic Natural Actor-Critic

Algorithm	Fair performance (>-120) after	Good performance (>-80) after	best performance
Episodic Natural Actor-Critic	750	5,050	-55
Episodic Natural Actor-Critic with RPROP	Not reached	Not reached	-130

Episodic Natural Actor-Critic

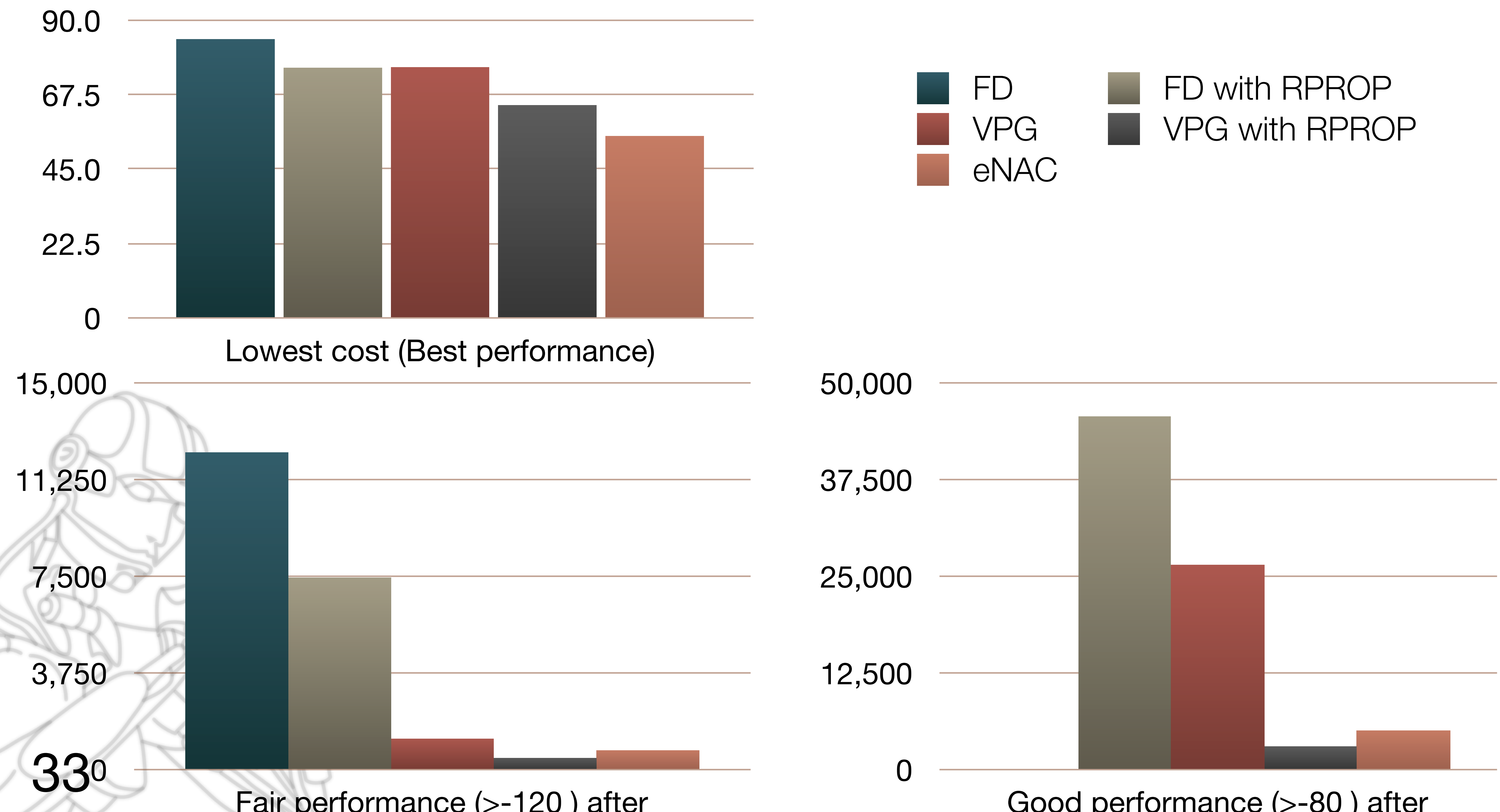
Algorithm	Fair performance (>-120) after	Good performance (>-80) after	best performance Best Final Performance
Episodic Natural Actor-Critic	750	5,050	-55
Episodic Natural Actor-Critic with RPROP	Not reached	Not reached	-130

Episodic Natural Actor-Critic

Algorithm	Fair performance (>-120) after	Good performance (>-80) after	best performance Best Final Performance
Episodic Natural Actor-Critic	750	5,050	-55
Episodic Natural Actor-Critic with RPROP	Not reached	Not reached	-130

RPROP Updates do not seem to be compatible with the Episodic Natural Actor-Critic

Comparison of the Results



Natural Actor-Critic

Given: A parameterized stochastic policy (e.g., Gaussian)

Natural Actor-Critic

Given: A parameterized stochastic policy (e.g., Gaussian)

1. Perform trajectories and collect data.

Natural Actor-Critic

Given: A parameterized stochastic policy (e.g., Gaussian)

1. Perform trajectories and collect data.
2. Estimate the (natural) gradient using the compatible function approximation.

Natural Actor-Critic

Given: A parameterized stochastic policy (e.g., Gaussian)

1. Perform trajectories and collect data.
2. Estimate the (natural) gradient using the compatible function approximation.
3. Update the policy with gradient descent.

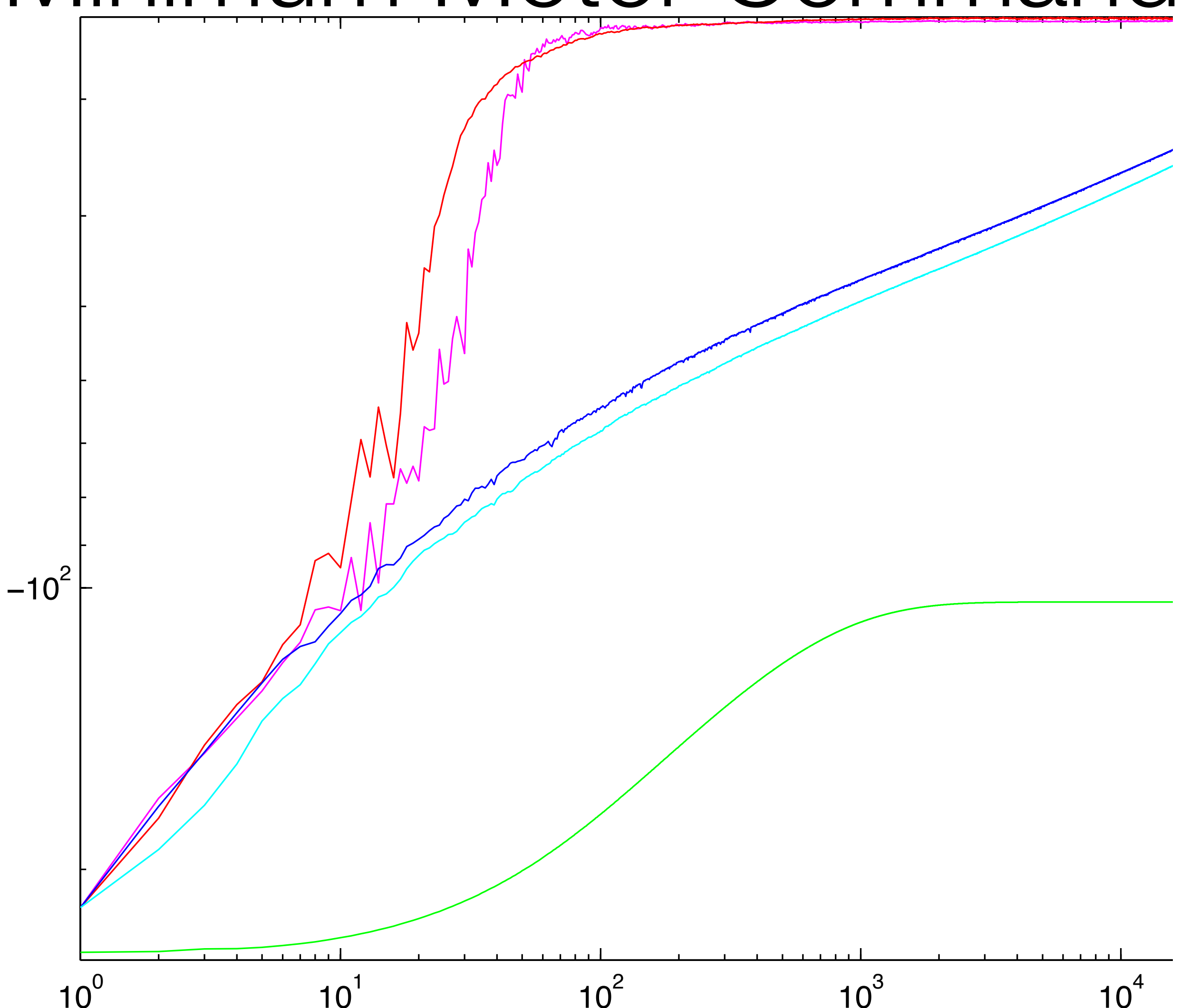
Natural Actor-Critic

Given: A parameterized stochastic policy (e.g., Gaussian)

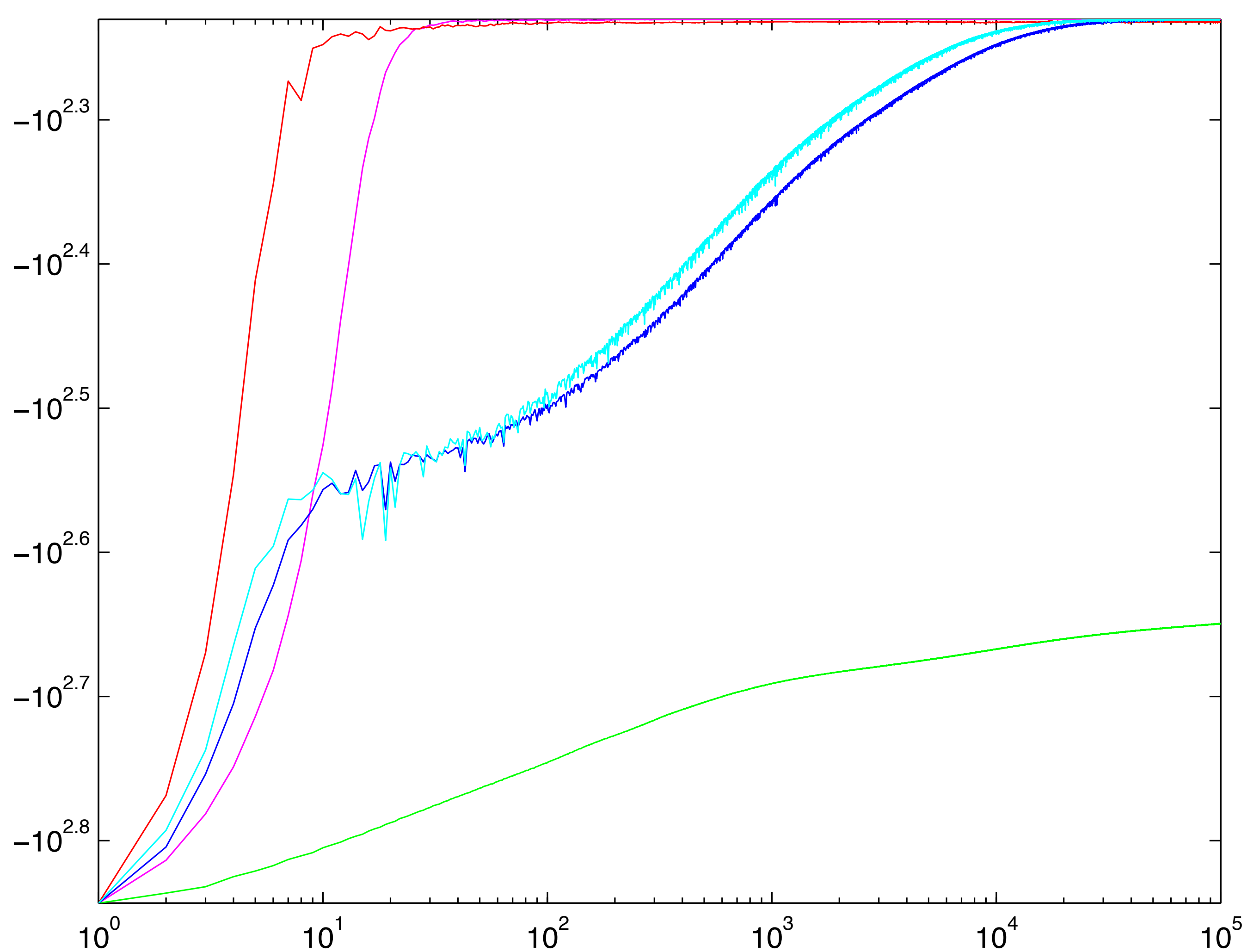
1. Perform trajectories and collect data.
2. Estimate the (natural) gradient using the compatible function approximation.
3. Update the policy with gradient descent.
4. Return to 1.

Improving MPs

Minimum Motor Command



Two Goals Policies



Learning T-Ball

Learning T-Ball

1) Teach motor primitives by imitation

Learning T-Ball

1) Teach motor primitives by imitation
2) Improve movement by Episodic Natural-Actor Critic

Learning T-Ball

1) Teach motor primitives by imitation
2) Improve movement by Episodic Natural-Actor Critic

Good performance often after 150-300 trials.

Outline of the Lecture

1. Introduction with Policy Gradients

2. Recent Advances in Policy Gradients

3. Probabilistic Policy Search with EM-like Approaches

4. Conclusion

Objective & Assumptions

Objective: maximize expected return

$$J(\boldsymbol{\theta}) = \int_{\mathbb{T}} p(\boldsymbol{\tau}) R(\boldsymbol{\tau}) d\boldsymbol{\tau}$$

Assumptions: Markovian & accumulated reward

path distribution

$$p(\boldsymbol{\tau}) = p(\mathbf{x}_1) \prod_{t=1}^{T} p(\mathbf{x}_{t+1} | \mathbf{x}_t, \mathbf{u}_t) \pi(\mathbf{u}_t | \mathbf{x}_t)$$

return

$$R(\boldsymbol{\tau}) = \frac{1}{T} \sum_{t=1}^{T} r(\mathbf{x}_t, \mathbf{u}_t)$$

!

Success Matching Principle

"When learning from a set of their own trials in iterated decision problems, humans attempt to match not the best taken action but the reward-weighted frequency of their actions and outcomes" (Arrow, 1958).

Success Matching Principle

> "When learning from a set of their own trials in iterated decision problems, humans attempt to match not the best taken action but the reward-weighted frequency of their actions and outcomes" (Arrow, 1958).

Thus, why don't we create policies such that $\pi'(\mathbf{u}|\mathbf{x})$ matches $\pi(\mathbf{u}|\mathbf{x})r(\mathbf{x},\mathbf{u})$?

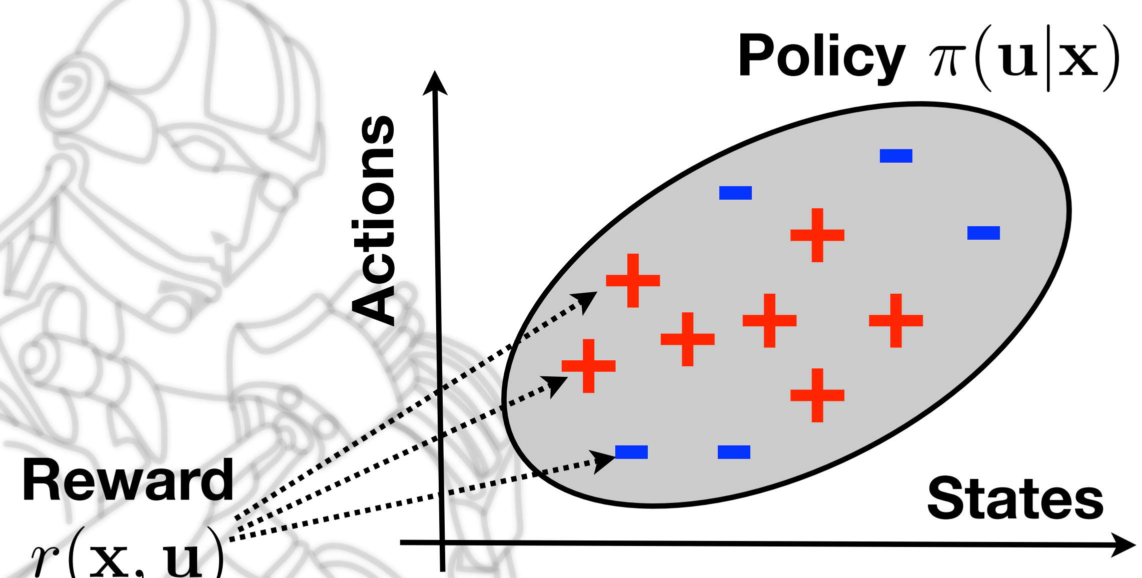
(Dayan & Hinton, 1998)

Success Matching Principle

> “When learning from a set of their own trials in iterated decision problems, humans attempt to match not the best taken action but the reward-weighted frequency of their actions and outcomes” (Arrow, 1958).

Thus, why don’t we create policies such that $\pi'(\mathbf{u}|\mathbf{x})$ matches $\pi(\mathbf{u}|\mathbf{x})r(\mathbf{x},\mathbf{u})$?

(Dayan & Hinton, 1998)



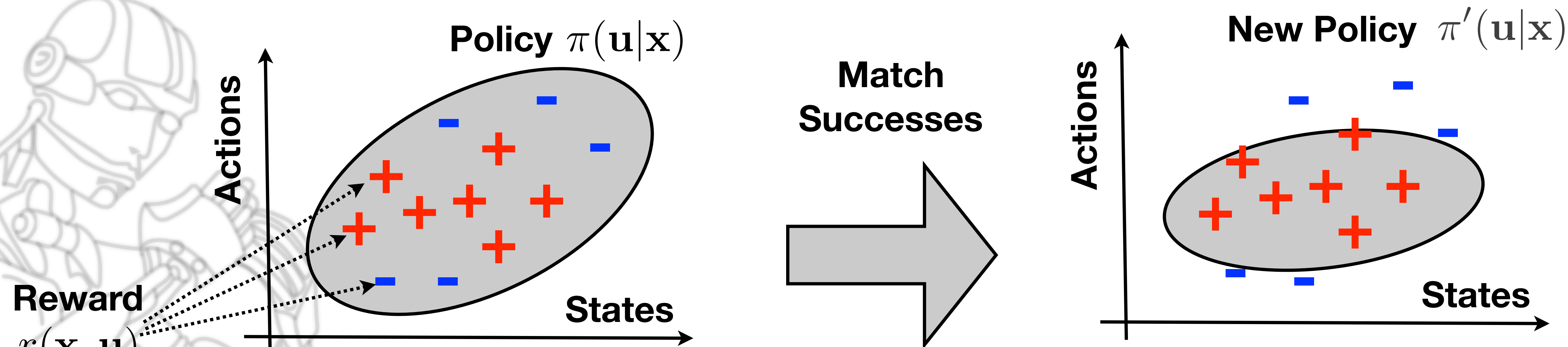
+ Succes (high reward) **- Failure (low reward)**

Success Matching Principle

> “When learning from a set of their own trials in iterated decision problems, humans attempt to match not the best taken action but the reward-weighted frequency of their actions and outcomes” (Arrow, 1958).

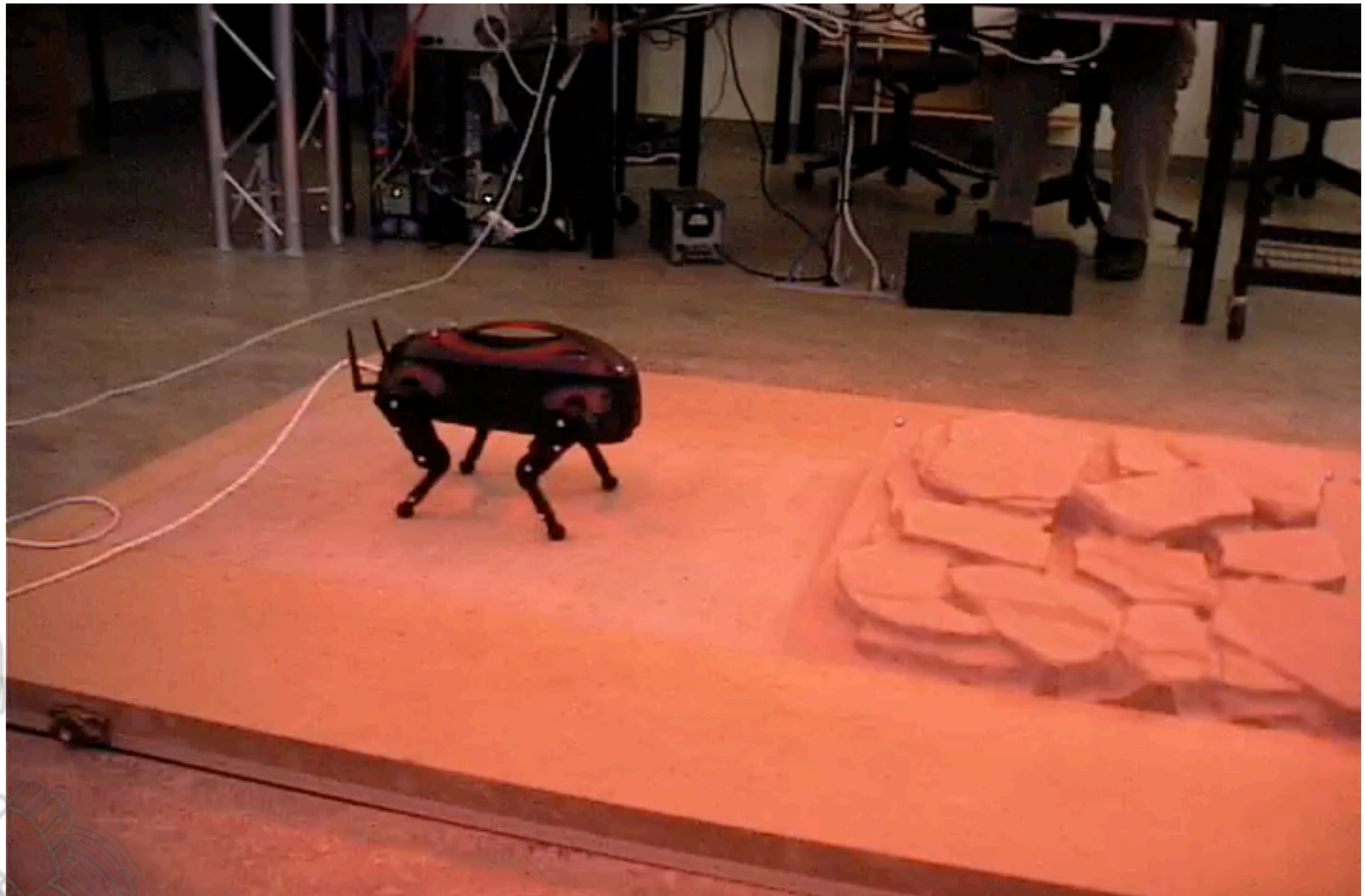
Thus, why don’t we create policies such that $\pi'(\mathbf{u}|\mathbf{x})$ matches $\pi(\mathbf{u}|\mathbf{x})r(\mathbf{x},\mathbf{u})$?

(Dayan & Hinton, 1998)



+ Succes (high reward) **- Failure (low reward)**

Selecting Footholds



Match successful footholds!

From Success Matching to Reward-Weighted Regression

Matching successful actions corresponds to minimizing the Kullback-Leibler 'distance'

$$D(r(\mathbf{x},\mathbf{u})\pi(\mathbf{u}||\mathbf{x})||\pi'(\mathbf{u}||\mathbf{x})) \rightarrow \min$$

or

$$D(p(\boldsymbol{\tau}|\pi)R(\boldsymbol{\tau})||p(\boldsymbol{\tau}|\pi')) \rightarrow \min$$

➡**This minimization can be shown to correspond to optimizing a lower bound on the expected return!**

!

Basic Intuition

Basic Intuition

- Lower Bound on Expected Return

Basic Intuition

- Lower Bound on Expected Return
  - reward is an improper probability distribution

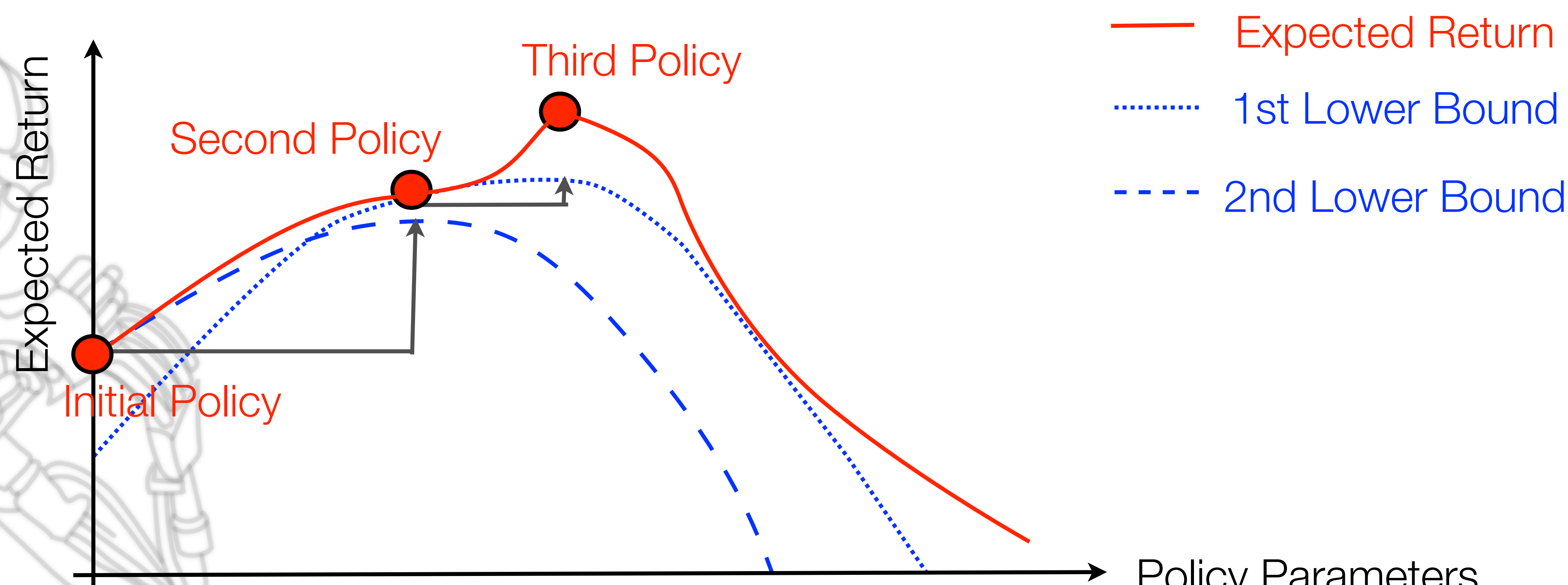
Basic Intuition

- Lower Bound on Expected Return
  - reward is an improper probability distribution
  - log-likelihood → log(expected return)
    (Dayan & Hinton, Neural Computation 1997; Peters & Schaal, ICML 2007)

Basic Intuition

- Lower Bound on Expected Return
  - reward is an improper probability distribution
  - log-likelihood → log(expected return)
    (Dayan & Hinton, Neural Computation 1997; Peters & Schaal, ICML 2007)

$$\log J(\boldsymbol{\theta}') \geq \int_{\mathbb{T}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau})\, R(\boldsymbol{\tau}) \log \frac{p_{\boldsymbol{\theta}'}(\boldsymbol{\tau})}{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})} d\boldsymbol{\tau} + \text{const} = L_{\boldsymbol{\theta}}(\boldsymbol{\theta}')$$



Resulting Algorithms

Resulting Algorithms

Policy Gradients: maximize lower bound by following the gradient

Resulting Algorithms

Policy Gradients: maximize lower bound by following the gradient

Resulting Algorithms

Policy Gradients: maximize lower bound by following the gradient

Resulting Algorithms

Policy Gradients: maximize lower bound by following the gradient

EM-like Methods: maximize lower bound by expectation-maximization

Resulting Algorithms

Policy Gradients: maximize lower bound by following the gradient

$$\lim_{\boldsymbol{\theta}' \to \boldsymbol{\theta}} \partial_{\boldsymbol{\theta}'} L_{\boldsymbol{\theta}}(\boldsymbol{\theta}') = \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$\boldsymbol{\theta}' \approx \boldsymbol{\theta} + \alpha \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

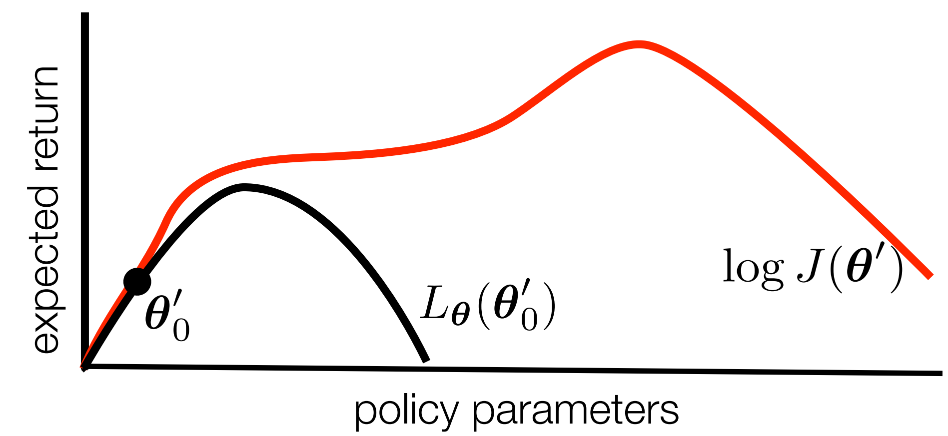
EM-like Methods: maximize lower bound by expectation-maximization

Resulting Algorithms

Policy Gradients: maximize lower bound by following the gradient

$$\lim_{\boldsymbol{\theta}' \to \boldsymbol{\theta}} \partial_{\boldsymbol{\theta}'} L_{\boldsymbol{\theta}}(\boldsymbol{\theta}') = \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$\boldsymbol{\theta}' \approx \boldsymbol{\theta} + \alpha \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$



EM-like Methods: maximize lower bound by expectation-maximization

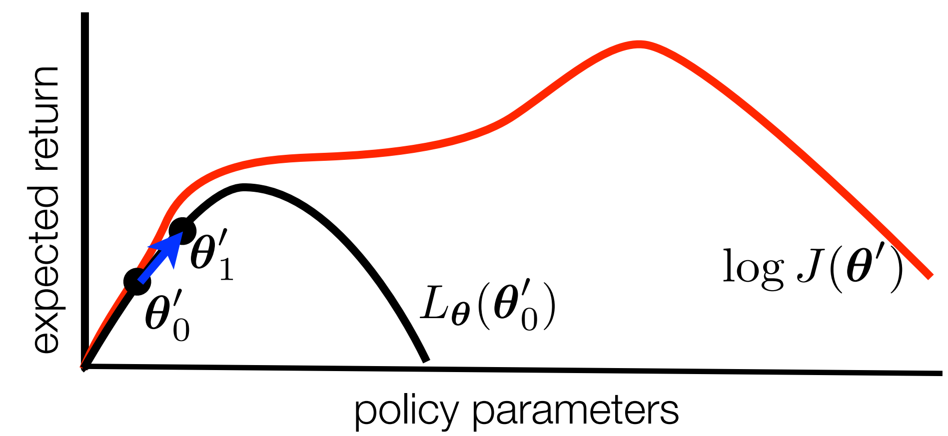
Thursday, May 17, 2012

Resulting Algorithms

Policy Gradients: maximize lower bound by following the gradient

$$\lim_{\boldsymbol{\theta}' \to \boldsymbol{\theta}} \partial_{\boldsymbol{\theta}'} L_{\boldsymbol{\theta}}(\boldsymbol{\theta}') = \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$\boldsymbol{\theta}' \approx \boldsymbol{\theta} + \alpha \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$



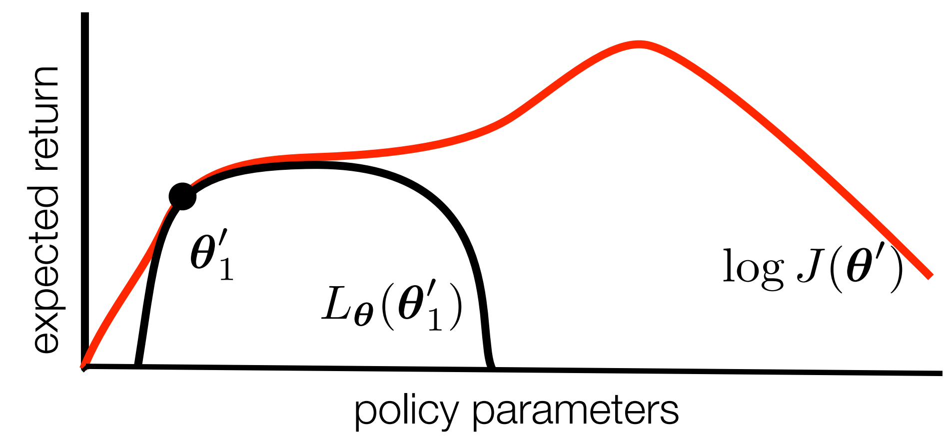
EM-like Methods: maximize lower bound by expectation-maximization

Resulting Algorithms

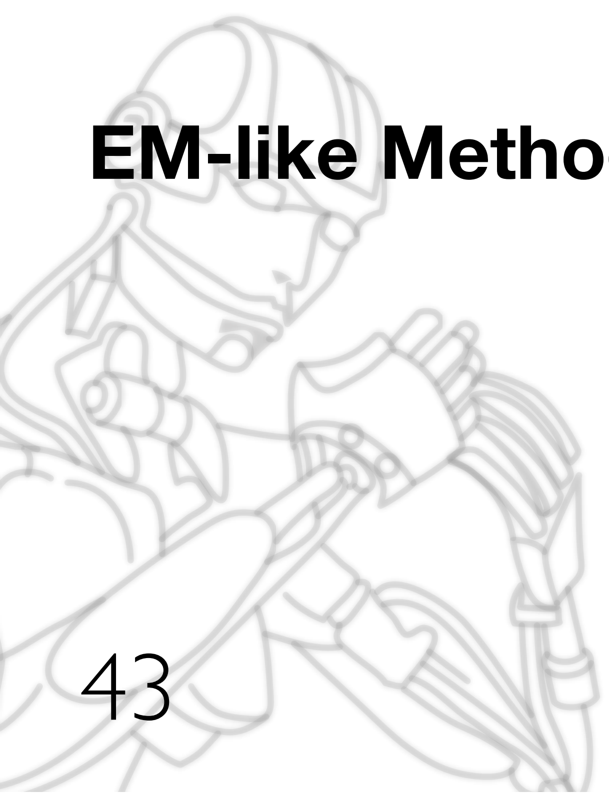
Policy Gradients: maximize lower bound by following the gradient

$$\lim_{\boldsymbol{\theta}' \to \boldsymbol{\theta}} \partial_{\boldsymbol{\theta}'} L_{\boldsymbol{\theta}}(\boldsymbol{\theta}') = \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$\boldsymbol{\theta}' \approx \boldsymbol{\theta} + \alpha \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$



EM-like Methods: maximize lower bound by expectation-maximization

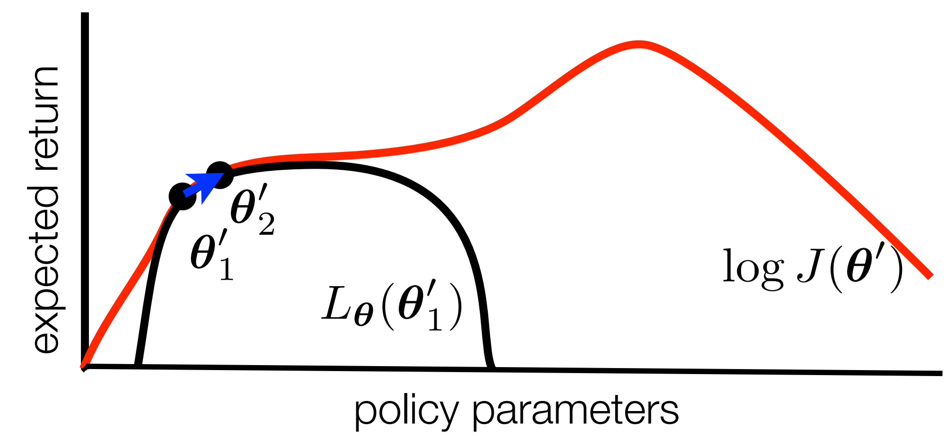


Resulting Algorithms

Policy Gradients: maximize lower bound by following the gradient

$$\lim_{\boldsymbol{\theta}' \to \boldsymbol{\theta}} \partial_{\boldsymbol{\theta}'} L_{\boldsymbol{\theta}}(\boldsymbol{\theta}') = \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$\boldsymbol{\theta}' \approx \boldsymbol{\theta} + \alpha \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$



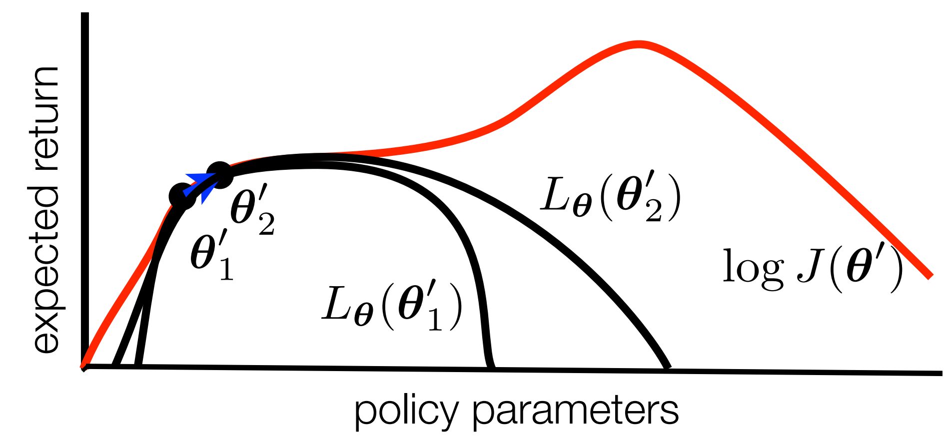
EM-like Methods: maximize lower bound by expectation-maximization

Resulting Algorithms

Policy Gradients: maximize lower bound by following the gradient

$$\lim_{\boldsymbol{\theta}' \to \boldsymbol{\theta}} \partial_{\boldsymbol{\theta}'} L_{\boldsymbol{\theta}}(\boldsymbol{\theta}') = \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$\boldsymbol{\theta}' \approx \boldsymbol{\theta} + \alpha \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$



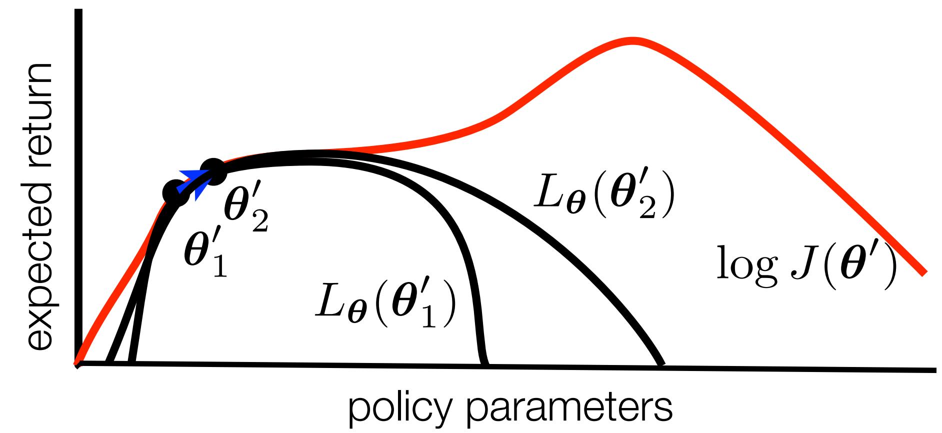
EM-like Methods: maximize lower bound by expectation-maximization

Resulting Algorithms

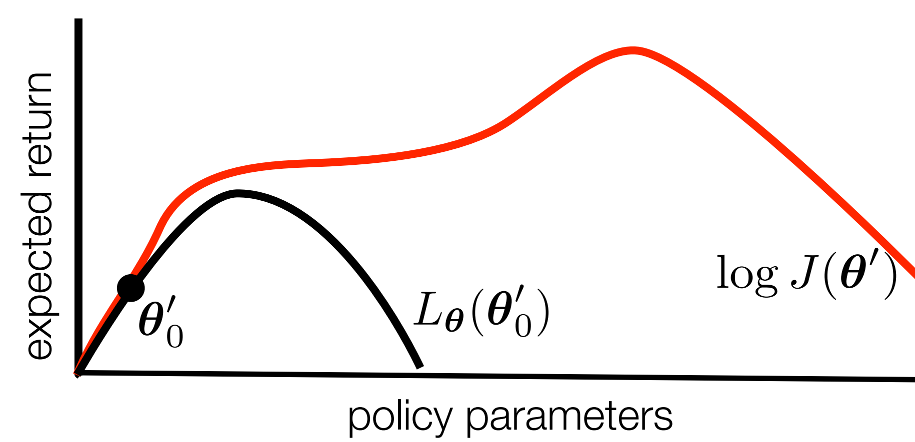
Policy Gradients: maximize lower bound by following the gradient

$$\lim_{\boldsymbol{\theta}' \to \boldsymbol{\theta}} \partial_{\boldsymbol{\theta}'} L_{\boldsymbol{\theta}}(\boldsymbol{\theta}') = \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$\boldsymbol{\theta}' \approx \boldsymbol{\theta} + \alpha \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$



EM-like Methods: maximize lower bound by expectation-maximization

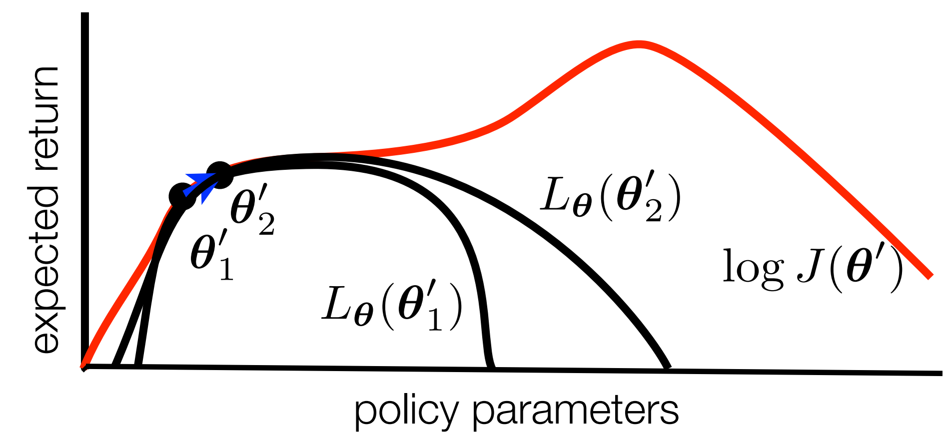


Resulting Algorithms

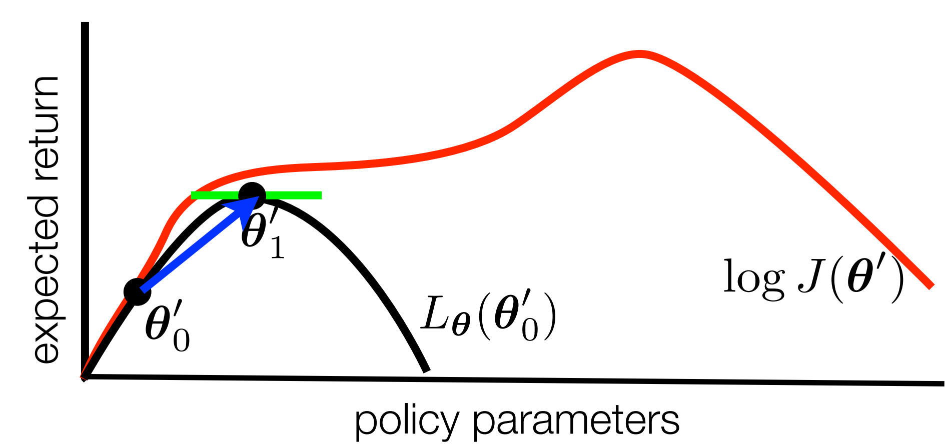
Policy Gradients: maximize lower bound by following the gradient

$$\lim_{\boldsymbol{\theta}' \to \boldsymbol{\theta}} \partial_{\boldsymbol{\theta}'} L_{\boldsymbol{\theta}}(\boldsymbol{\theta}') = \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$\boldsymbol{\theta}' \approx \boldsymbol{\theta} + \alpha \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$



EM-like Methods: maximize lower bound by expectation-maximization

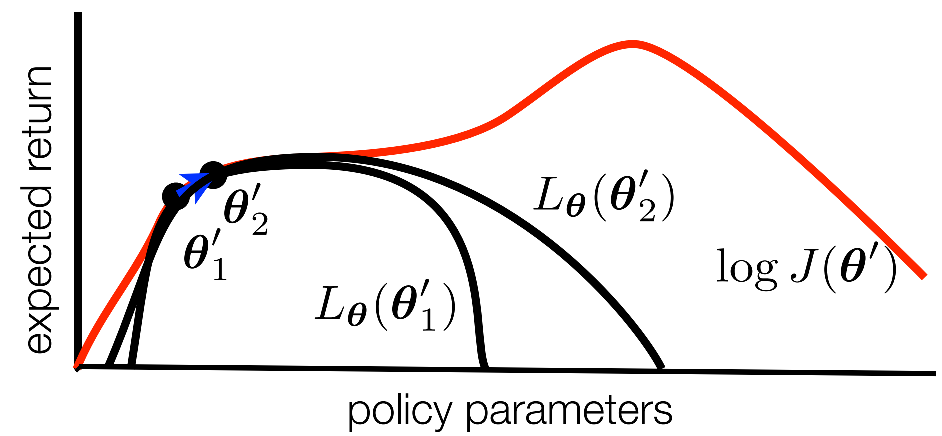


Resulting Algorithms

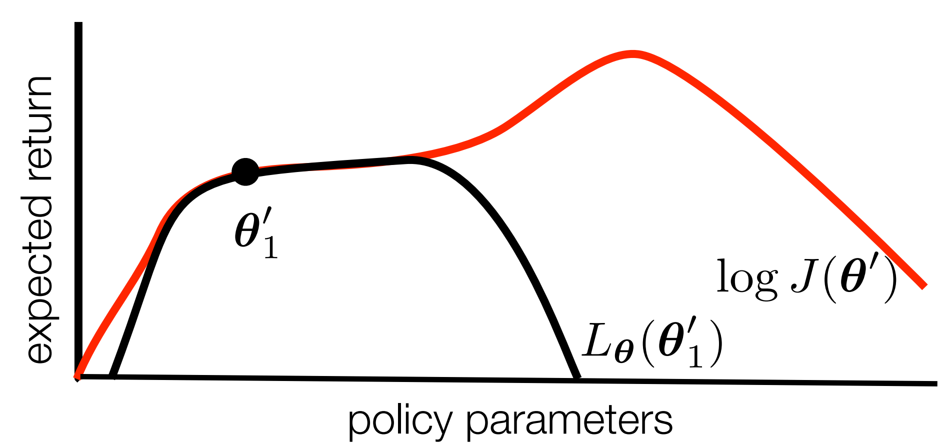
Policy Gradients: maximize lower bound by following the gradient

$$\lim_{\boldsymbol{\theta}' \to \boldsymbol{\theta}} \partial_{\boldsymbol{\theta}'} L_{\boldsymbol{\theta}}(\boldsymbol{\theta}') = \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$\boldsymbol{\theta}' \approx \boldsymbol{\theta} + \alpha \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$



EM-like Methods: maximize lower bound by expectation-maximization

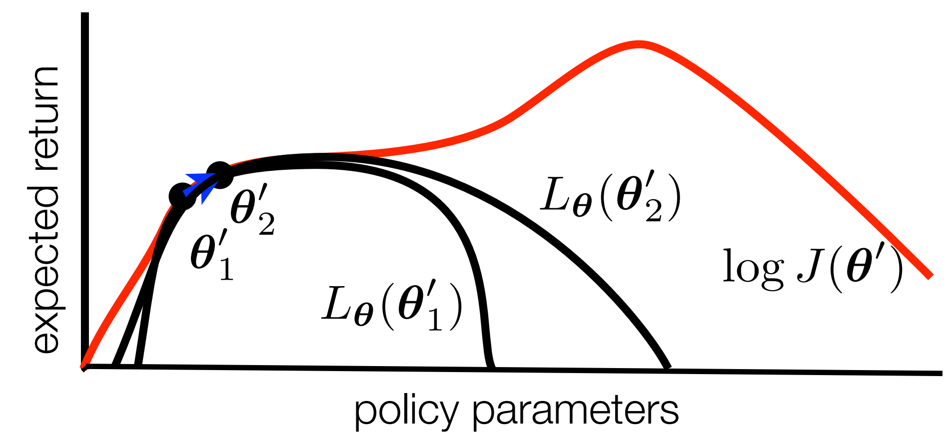


Resulting Algorithms

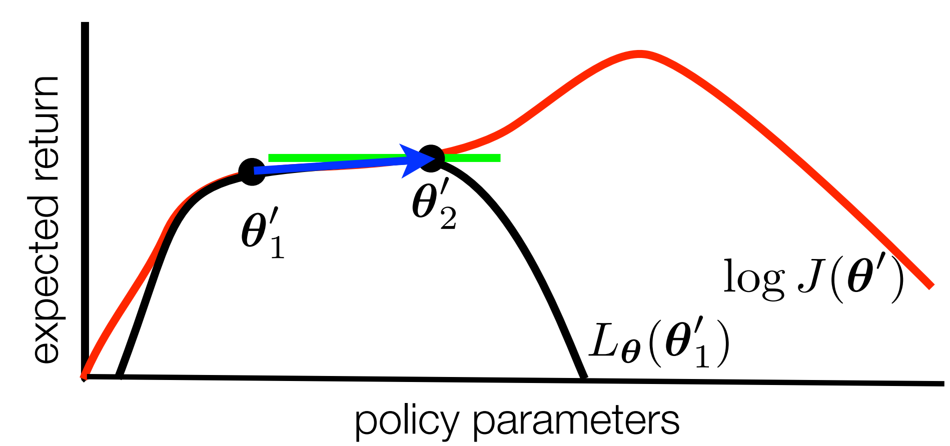
Policy Gradients: maximize lower bound by following the gradient

$$\lim_{\boldsymbol{\theta}' \to \boldsymbol{\theta}} \partial_{\boldsymbol{\theta}'} L_{\boldsymbol{\theta}}(\boldsymbol{\theta}') = \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$\boldsymbol{\theta}' \approx \boldsymbol{\theta} + \alpha \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$



EM-like Methods: maximize lower bound by expectation-maximization

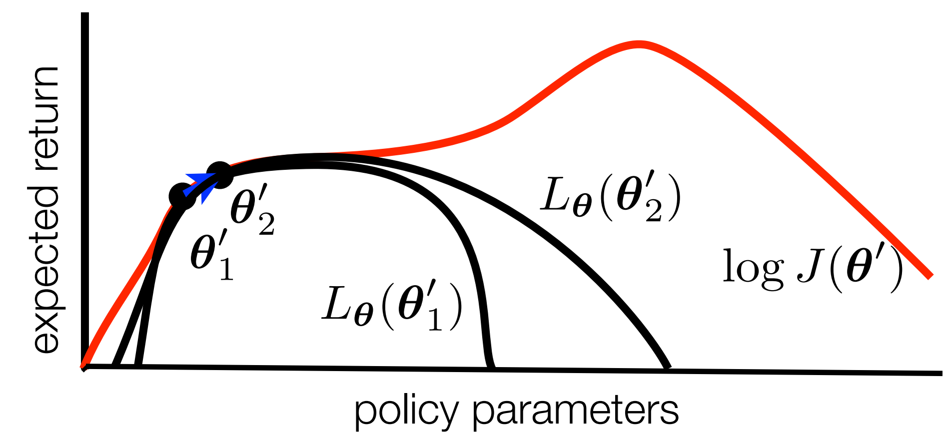


Resulting Algorithms

Policy Gradients: maximize lower bound by following the gradient

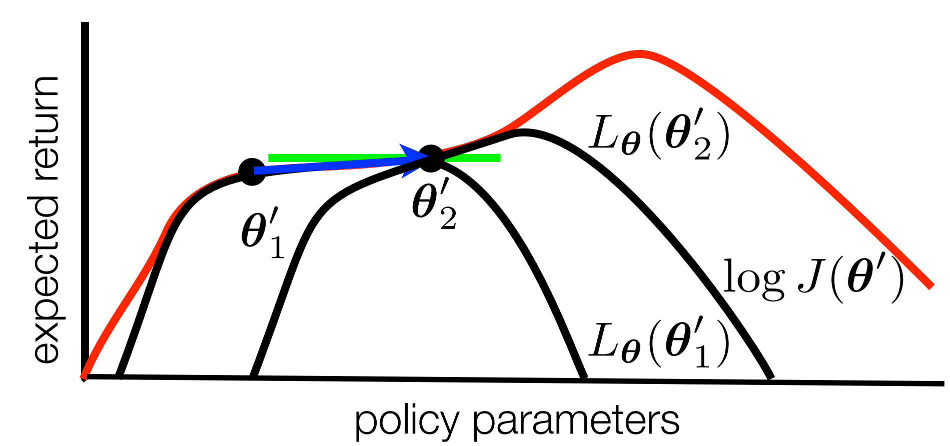
$$\lim_{\boldsymbol{\theta}' \to \boldsymbol{\theta}} \partial_{\boldsymbol{\theta}'} L_{\boldsymbol{\theta}}(\boldsymbol{\theta}') = \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$

$$\boldsymbol{\theta}' \approx \boldsymbol{\theta} + \alpha \partial_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$$



EM-like Methods: maximize lower bound by expectation-maximization

$$\boldsymbol{\theta}' = \operatorname{argmax} L_{\boldsymbol{\theta}}(\boldsymbol{\theta}')$$



Stochastic Policies

Use the **Policy**:

$$\mathbf{u} = f(\mathbf{x}) + \boldsymbol{\epsilon} = \boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}) + \boldsymbol{\epsilon}$$

with Gaussian exploration

$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2)$ → episodic Reward Weighted Regression

with State-dependent exploration

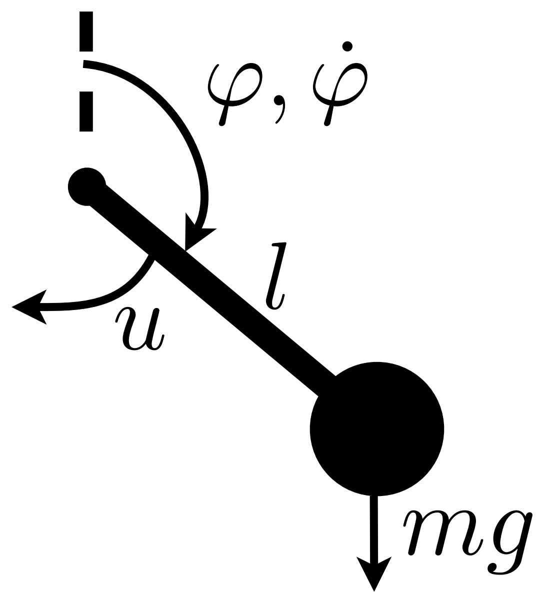
$\boldsymbol{\epsilon} = \boldsymbol{\varepsilon}^T \phi(\mathbf{x})$ with $\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma^2)$ → PoWER

Underactuated Swing-Up

(Schaal, NIPS 1997; Atkeson, ICML 1997)

Underactuated Swing-Up

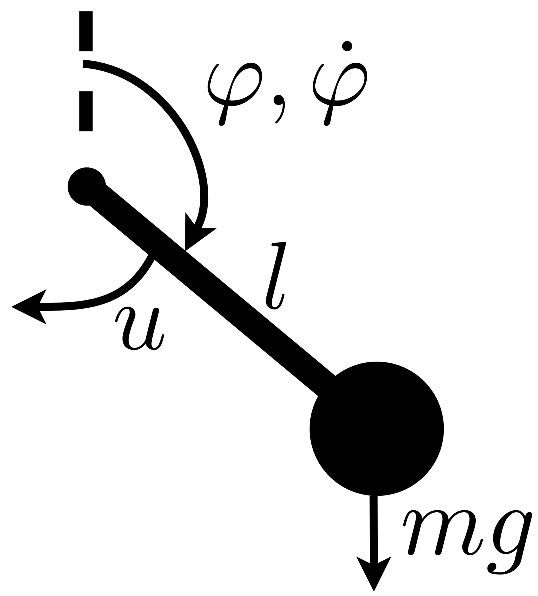
- swing heavy pendulum up



(Schaal, NIPS 1997; Atkeson, ICML 1997)

Underactuated Swing-Up

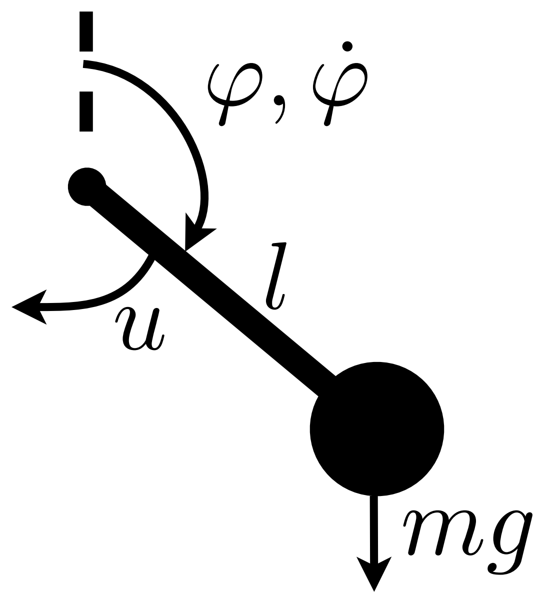
- swing heavy pendulum up



(Schaal, NIPS 1997; Atkeson, ICML 1997)

Underactuated Swing-Up

- swing heavy pendulum up



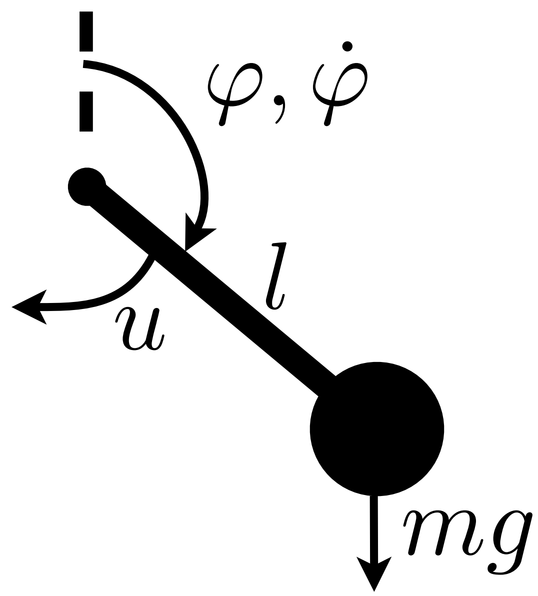
- motor torques limited

$$|u| \leq u_{max}$$

(Schaal, NIPS 1997; Atkeson, ICML 1997)

Underactuated Swing-Up

- swing heavy pendulum up



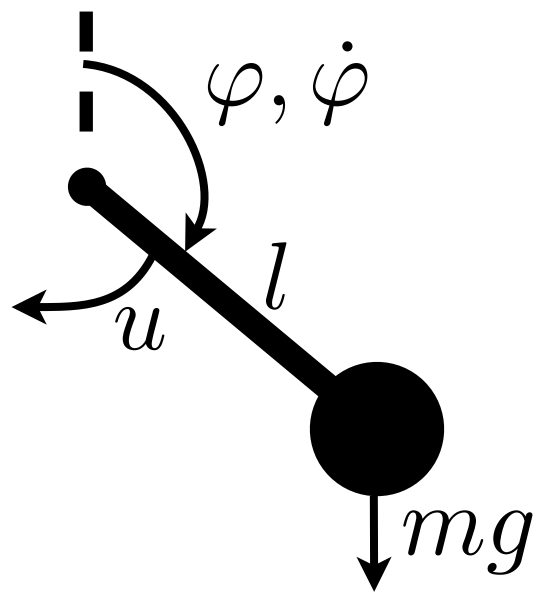
- motor torques limited

$$|u| \leq u_{max}$$

(Schaal, NIPS 1997; Atkeson, ICML 1997)

Underactuated Swing-Up

- swing heavy pendulum up



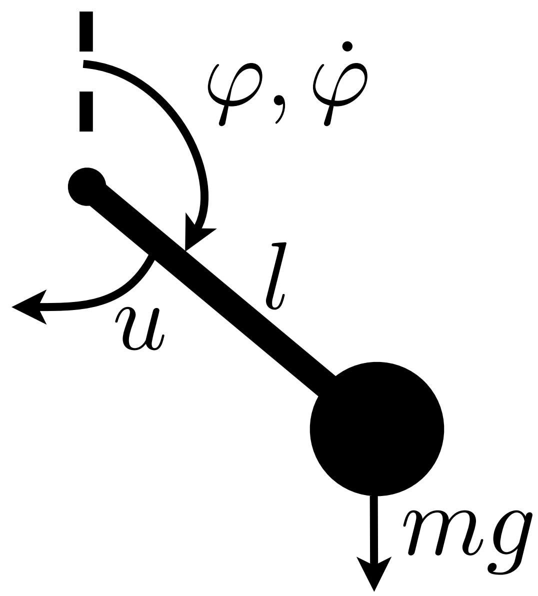
- motor torques limited

$$|u| \leq u_{max}$$

(Schaal, NIPS 1997; Atkeson, ICML 1997)

Underactuated Swing-Up

- swing heavy pendulum up



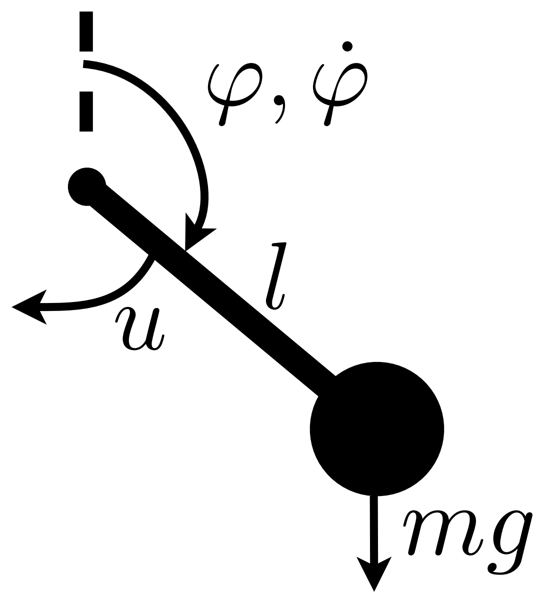
- motor torques limited

$$|u| \leq u_{max}$$

(Schaal, NIPS 1997; Atkeson, ICML 1997)

Underactuated Swing-Up

- swing heavy pendulum up



$$ml^2\ddot{\varphi} = -\mu\dot{\varphi} + mgl\sin\varphi + u$$
$$\varphi \in [-\pi, \pi]$$

- motor torques limited

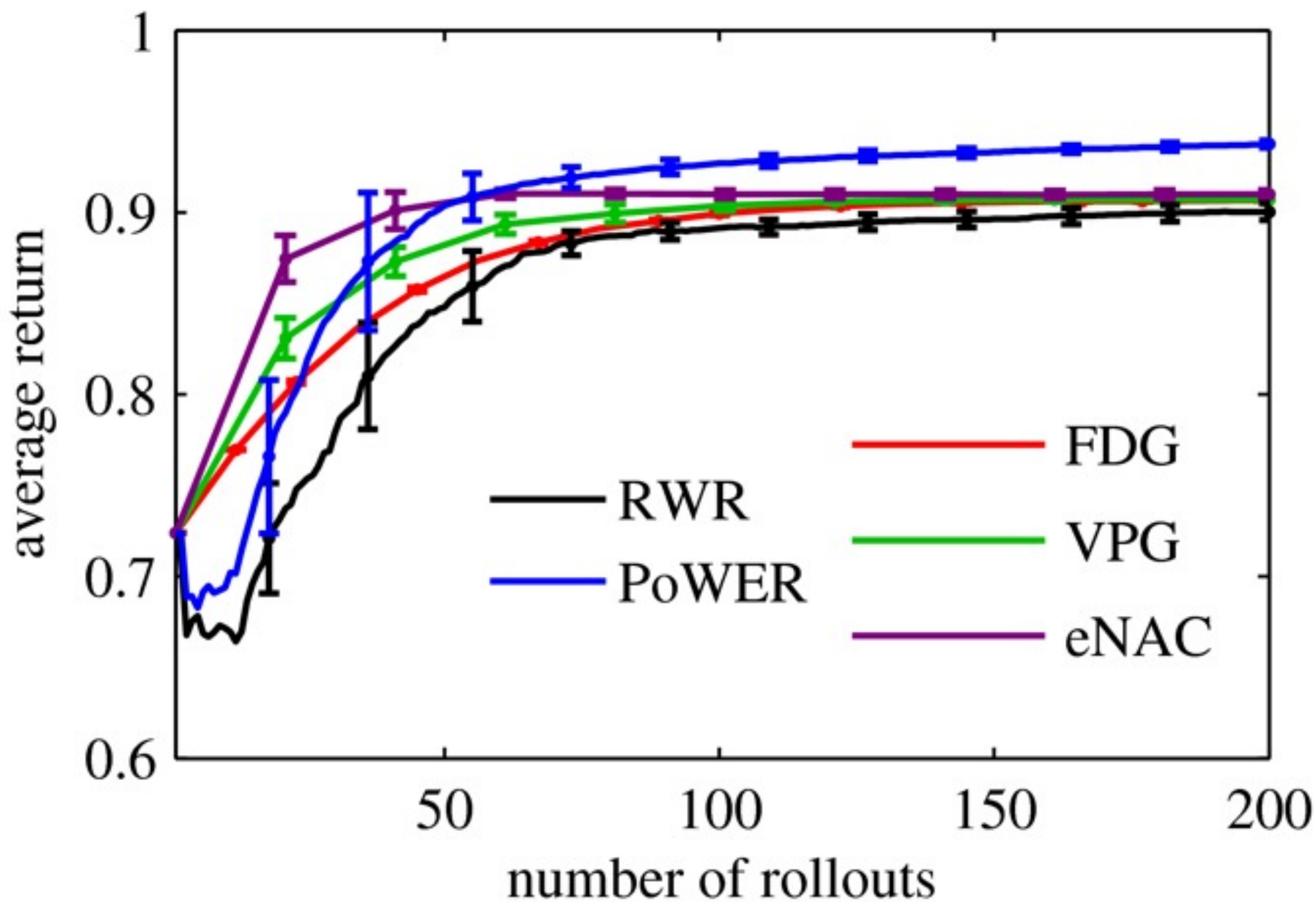
$$|u| \leq u_{max}$$

- reward function

$$r = \exp\left(-\alpha\left(\frac{\varphi}{\pi}\right)^2 - \beta\left(\frac{2}{\pi}\right)^2 \log\cos\left(\frac{\pi}{2}\frac{u}{u_{max}}\right)\right)$$

(Schaal, NIPS 1997; Atkeson, ICML 1997)

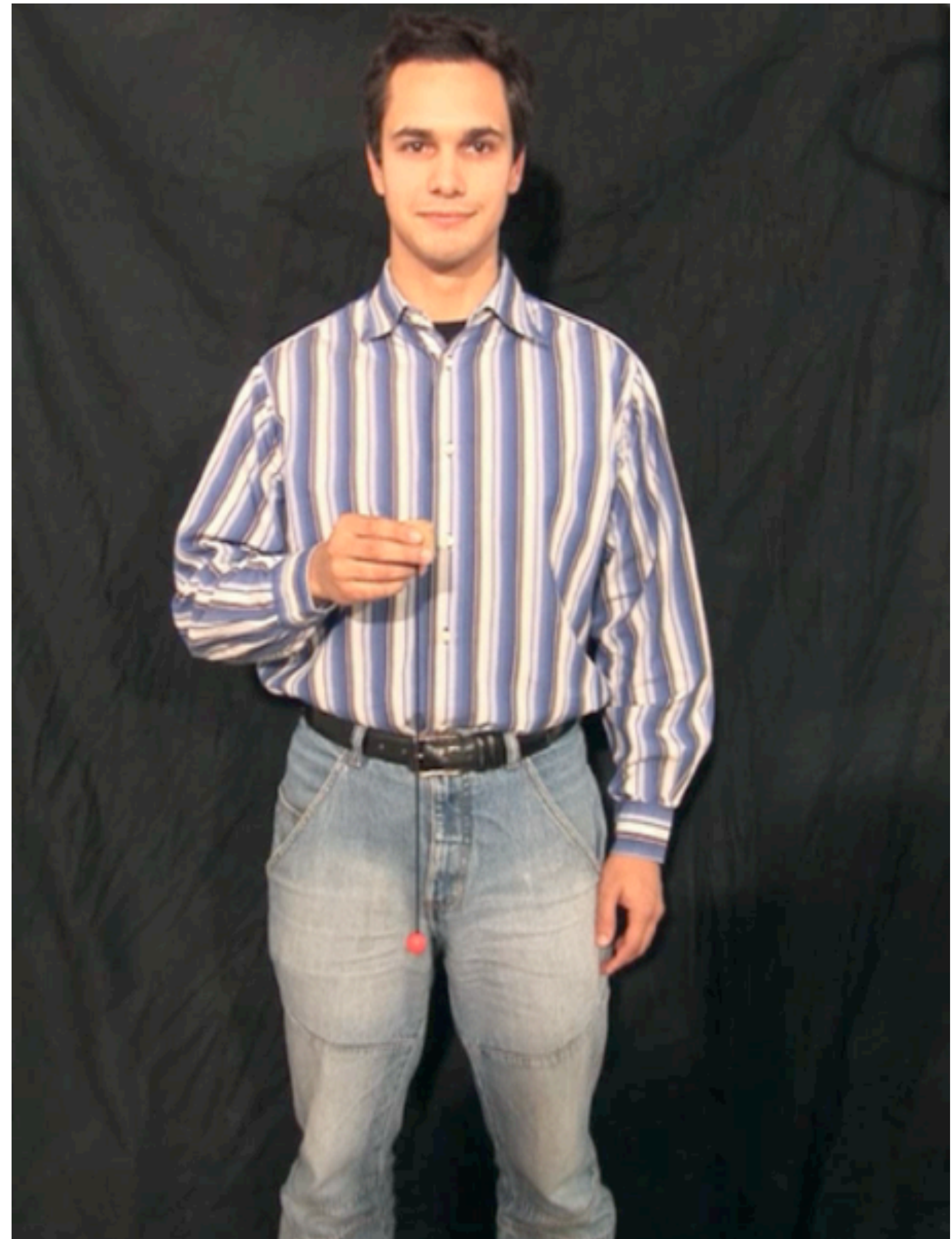
Underactuated Swing-Up



(Peters & Schaal, IROS 2006; Peters & Schaal, ICML 2007)

Ball-in-a-Cup

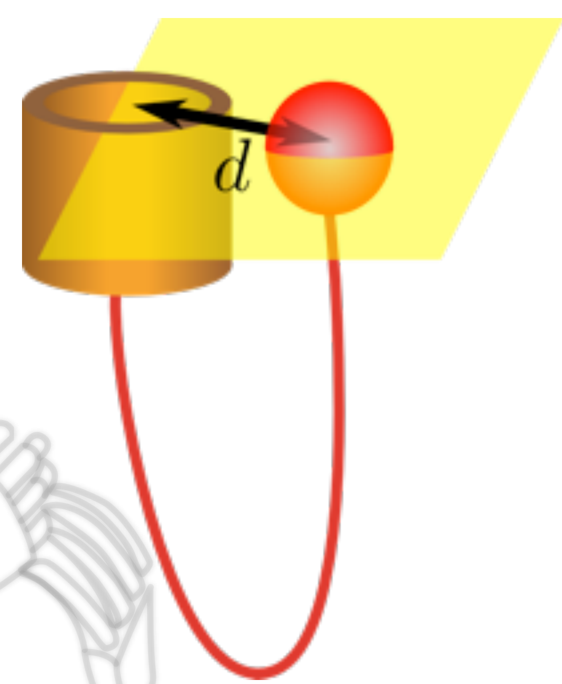
Ball-in-a-Cup



Ball-in-a-Cup

- reward function

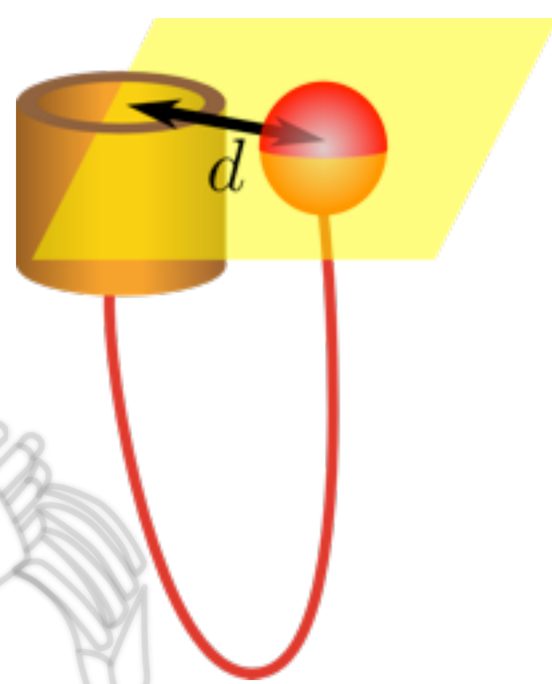
$$r_t = \begin{cases} \exp\left(-\alpha\left(\left(x_c - x_b\right)^2 + \left(y_c - y_b\right)^2\right)\right) & \text{if } t = t_c \\ 0 & \text{if } t \neq t_c \end{cases}$$

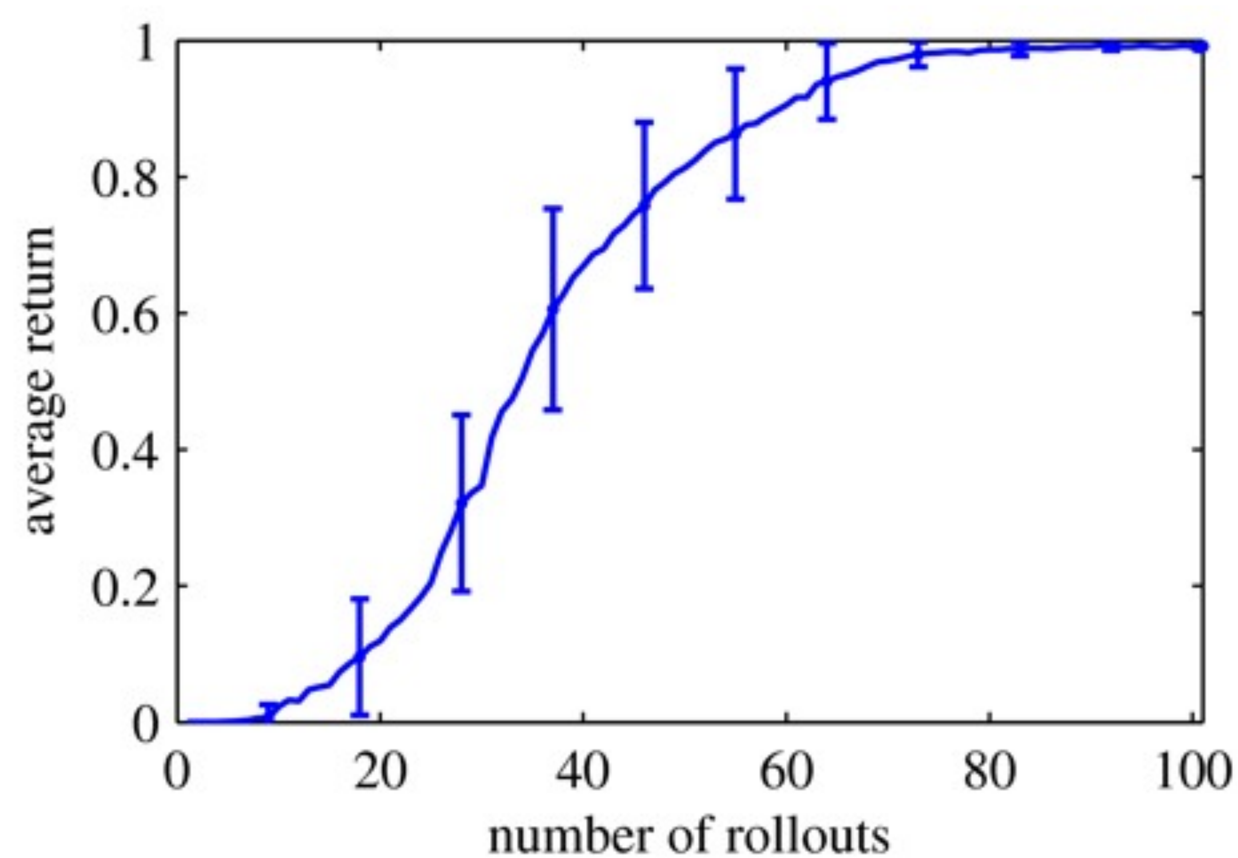


Ball-in-a-Cup

- reward function

$$r_t = \begin{cases} \exp\left(-\alpha\left((x_c - x_b)^2 + (y_c - y_b)^2\right)\right) & \text{if } t = t_c \\ 0 & \text{if } t \neq t_c \end{cases}$$





Cost-regularized Gaussian Processes

The Reward-Weighted Regression required known basis functions.
Using the Kernel-Trick

$$\begin{aligned}\bar{\mathbf{u}}_i &= \phi(\mathbf{x})^{\mathrm{T}}\mathbf{w} = \phi(\mathbf{x})^{\mathrm{T}}\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{R}\mathbf{\Phi} + \lambda\mathbf{I}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{R}\mathbf{U}_i \\ &= \phi(\mathbf{x})^{\mathrm{T}}\mathbf{\Phi}^{\mathrm{T}}\left(\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}} + \lambda\mathbf{R}^{-1}\right)^{-1}\mathbf{U}_i\end{aligned}$$

we can turn this in *a cost-regularized Gaussian Process approach*.
The predictive variance acts as a policy

$$\mathbf{u}^j \sim \pi_j(\mathbf{u}|\mathbf{x}^j) = \mathcal{N}(\mathbf{u}|\boldsymbol{\gamma}(\mathbf{x}^j), \sigma^2(\mathbf{x}^j)\mathbf{I})$$

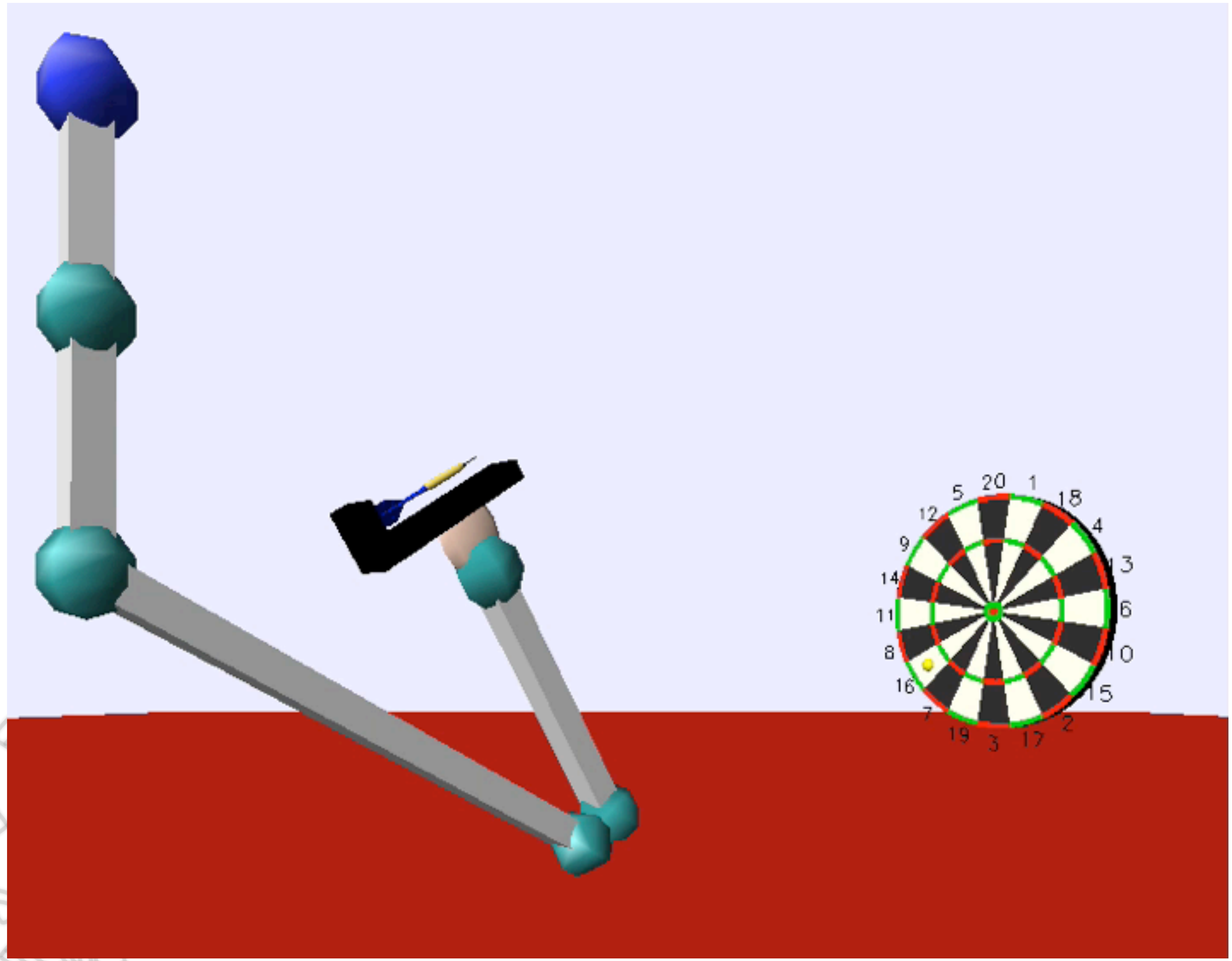
with

$$\gamma_i(\mathbf{x}^j) = \mathbf{k}(\mathbf{x}^j)^{\mathrm{T}}(\mathbf{K} + \lambda\mathbf{C})^{-1}\mathbf{U}_i$$

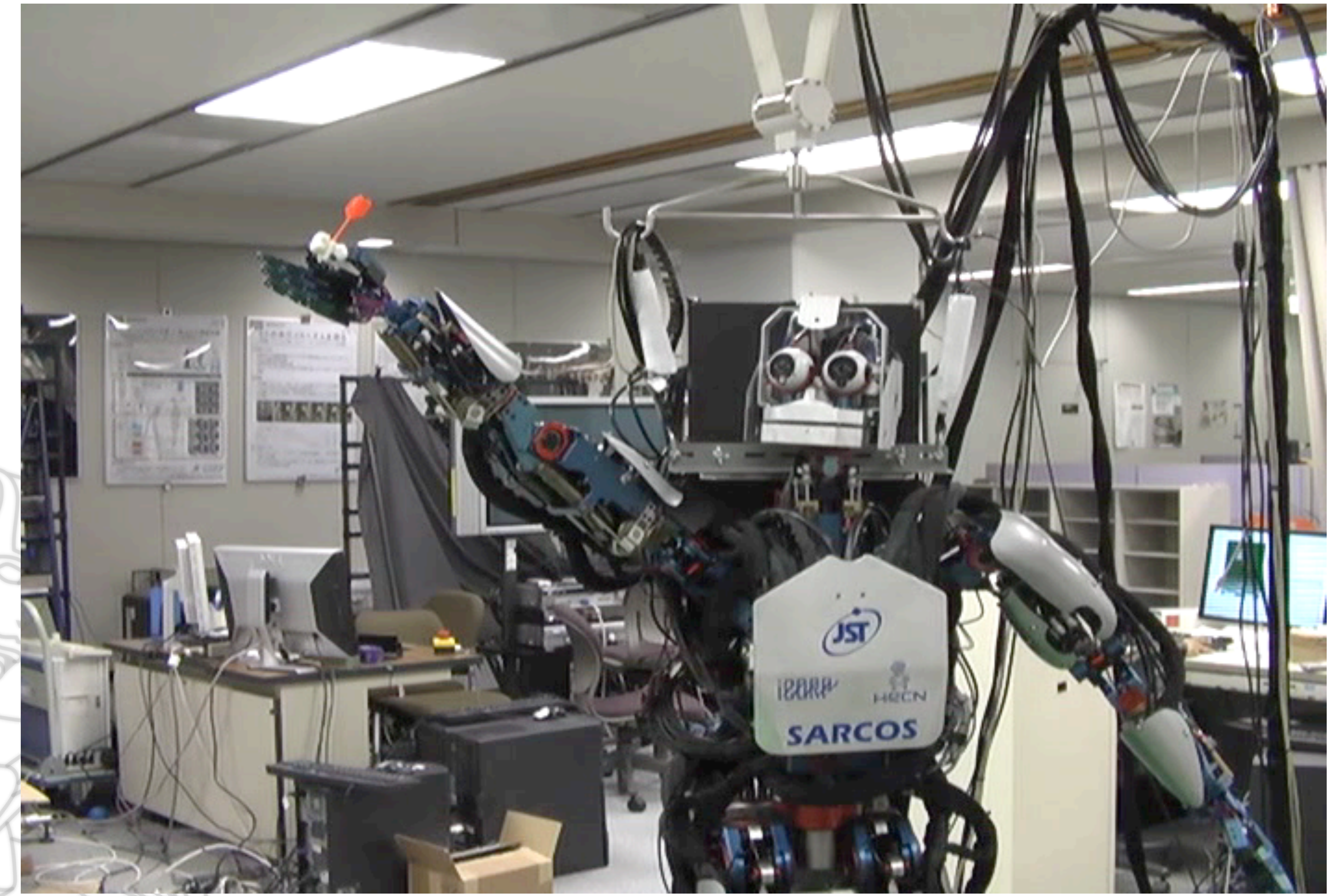
$$\sigma^2(\mathbf{x}^j) = k(\mathbf{x}^j, \mathbf{x}^j) - \mathbf{k}(\mathbf{x}^j)^{\mathrm{T}}(\mathbf{K} + \lambda\mathbf{C})^{-1}\mathbf{k}(\mathbf{x}^j)$$

(Kober, Oztop & Peters, R:SS 2010)

Dart-Throwing with Sledge



Dart-Throwing with Fingers



Learning for Table Tennis

Throwing and Catching

Outline of the Lecture

1. Introduction with Policy Gradients

2. Recent Advances in Policy Gradients

3. Probabilistic Policy Search with EM-like Approaches

4. Conclusion

Outline of the Lecture

1. Introduction with Policy Gradients

2. Recent Advances in Policy Gradients

3. Probabilistic Policy Search with EM-like Approaches

4. Conclusion

Conclusion

- Policy Search is a powerful and practical alternative to value function and model-based methods.

- Policy gradients have dominated this area for a long time and solidly working methods exist.

- Newer methods focus on probabilistic policy search approaches.

Further Reading

- Peters, J.;Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients, Neural Networks, 21, 4, pp.682-97

- Kober, J.; Peters, J. (2011). Policy Search for Motor Primitives in Robotics, Machine Learning, 84, 1-2, pp.171-203

- Peters, J.; Muelling, K.; Altun, Y. (2010). Relative Entropy Policy Search, Proceedings of the Twenty-Fourth National Conference on Artificial Intelligence (AAAI), Physically Grounded AI Track