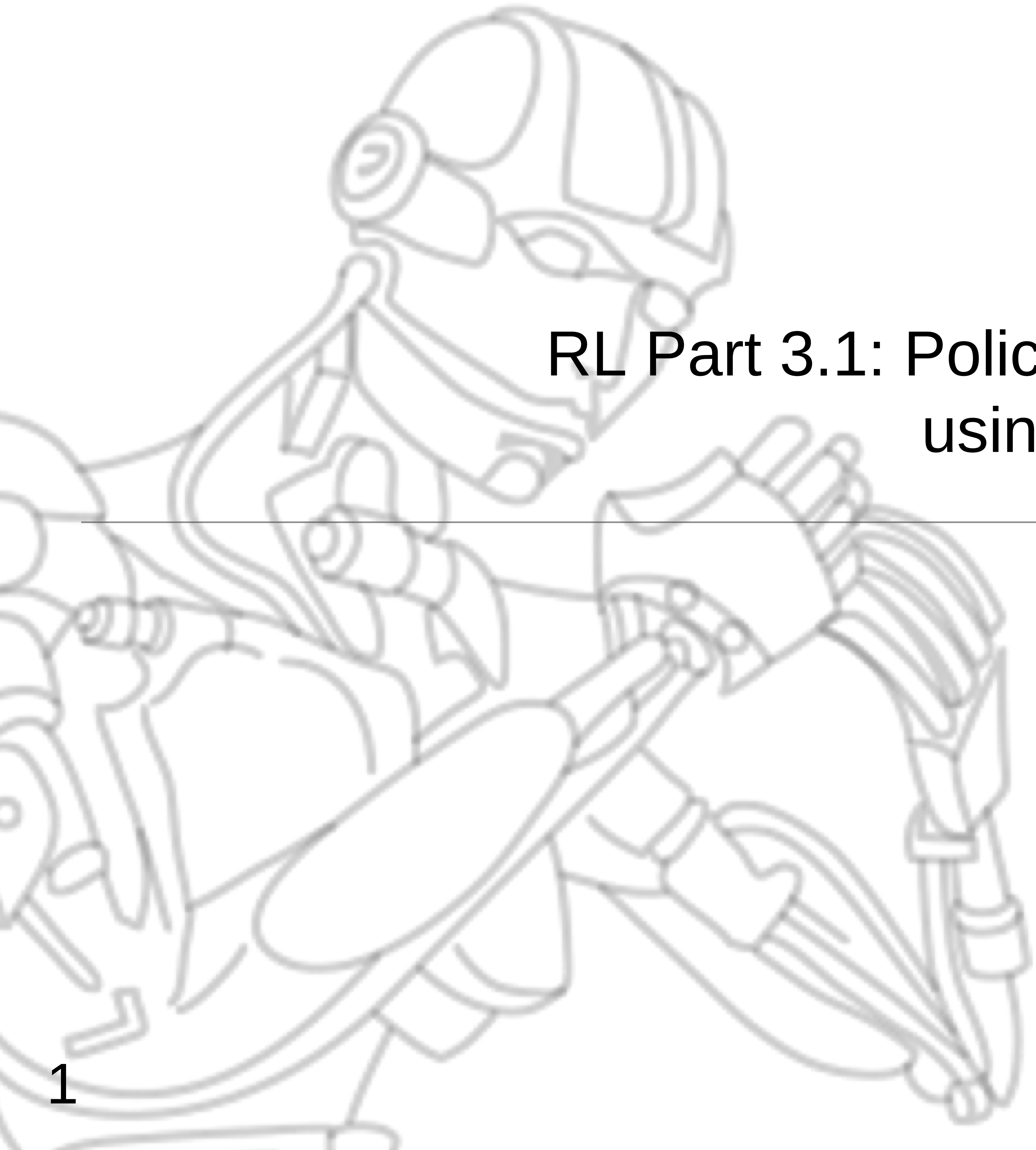


RL Part 3.1: Policy Search Methods using Policy Gradients

Jan Peters
Gerhard Neumann

Motivation

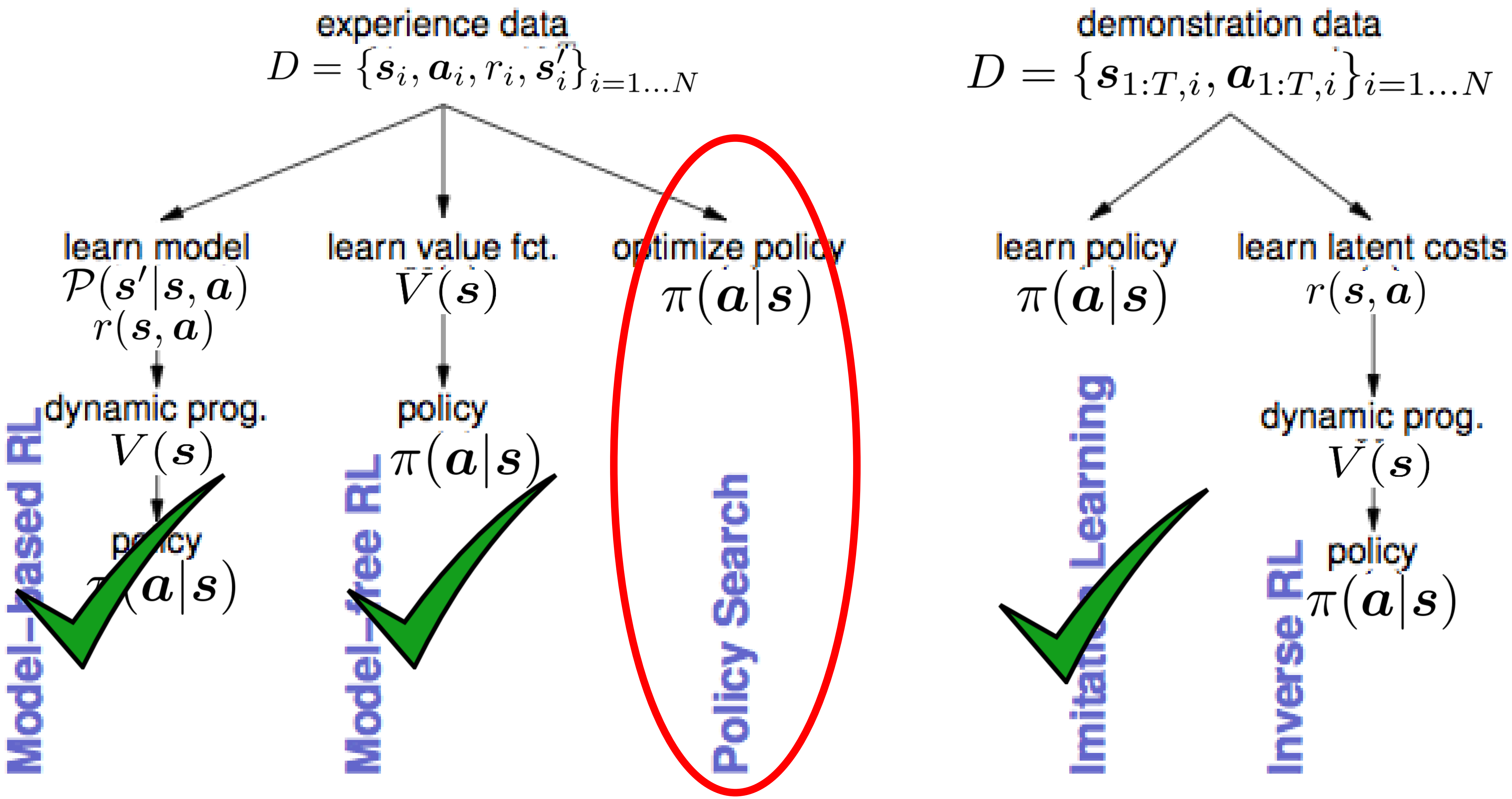
Limits of Value Functions:

- Fill-up state-space: Exponential explosion with the number of dimensions
- **Continuous actions**?
- **Value Function Approximation Error** might propagate and arbitrarily distort the policy update!
- **Exploration** on the real system?

Many of these problems can be fixed by **using parametric policies** and policy search

- Improving upon demonstrations
- Value function is not (always) needed
- Using task-appropriate policies is possible

Bigger Picture



experience data

$D = \{\boldsymbol{s}_i, \boldsymbol{a}_i, r_i, \boldsymbol{s}'_i\}_{i=1\dots N}$

learn model

$\mathcal{P}(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$

$r(\boldsymbol{s}, \boldsymbol{a})$

dynamic prog.

$V(\boldsymbol{s})$

policy

$\pi(\boldsymbol{a}|\boldsymbol{s})$

Model–based RL

learn value fct.

$V(\boldsymbol{s})$

policy

$\pi(\boldsymbol{a}|\boldsymbol{s})$

Model–free RL

optimize policy

$\pi(\boldsymbol{a}|\boldsymbol{s})$

Policy Search

demonstration data

$D = \{\boldsymbol{s}_{1:T,i}, \boldsymbol{a}_{1:T,i}\}_{i=1\dots N}$

learn policy

$\pi(\boldsymbol{a}|\boldsymbol{s})$

Imitation Learning

learn latent costs

$r(\boldsymbol{s}, \boldsymbol{a})$

dynamic prog.

$V(\boldsymbol{s})$

policy

$\pi(\boldsymbol{a}|\boldsymbol{s})$

Inverse RL

Outline of the Lecture

1. Categorization of Policy Search

I. Episode-Based versus Step-Based Policy Search

2. Policy Gradients

I. Episode-Based Policy Gradients

II. Step-Based Policy Gradients

3. Relative Entropy and Natural Gradients

[Deisenroth, Neumann, Peters: „A survey on Policy Search in Robotics“, 2013]

Action Selection

... in value-based algorithms:

Greedy or soft-max policy: $\pi(\boldsymbol{a}|\boldsymbol{s}) = \dfrac{\exp(\beta Q(\boldsymbol{s}, \boldsymbol{a}))}{\sum_{\boldsymbol{a}'} \exp(\beta Q(\boldsymbol{s}, \boldsymbol{a}'))}$

Difficult in continuous action spaces

Alternatively, we can use parametrized policies for action selection

For example: Gaussian Policies

$$\pi(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{a}|f_{\boldsymbol{w}}(\boldsymbol{s}), \sigma^2 \boldsymbol{I}), \quad \boldsymbol{\theta} = \{\boldsymbol{w}, \sigma^2\}$$

Continuous actions can be easily incorporated

Policy Search: How to find good parameters $\boldsymbol{\theta}$?

Model-free policy search

Pseudo-Algorithm:

Repeat

1. **Explore:** Generate trajectories $\boldsymbol{\tau}^{[i]}$ following the current policy π_k

2. **Policy Evaluation:** Assess quality of trajectory or actions

   ➡ **Episode-Based Policy Evaluation**

   ➡ **Step-Based Policy Evaluation**

3. **Policy Update:** Compute new policy π_{k+1} from trajectories and evaluations

Episode-based evaluation strategy

Evaluation Strategy:

- We directly asses the quality of a parameter vector $\boldsymbol{\theta}^{[i]}$ by the returns

$$R_{[i]} = \sum_{t=1}^{T} r_t^{[i]}$$

- High variance in returns (sum of T random variables)

Data-set used for policy update

$$\mathcal{D}_{\mathrm{episode}} = \left\{\boldsymbol{\theta}^{[i]}, R^{[i]}\right\}_{i=1\ldots N}$$

- One data-point per trajectory
- Works for a **moderate number of parameters**

Explore in parameter space at each episode

Step-based evaluation strategy

Evaluation Strategy:

We assess the quality of single state-action pairs by using the reward to come

$$Q_t^{[i]} = \sum_{h=t}^{T} r_h^{[i]}$$

Less variance in Q_t (sum of T-t random variables)

Data-set used for policy update:

$$\mathcal{D}_{\text{step}} = \left\{ \boldsymbol{s}_t^{[i]}, \boldsymbol{a}_t^{[i]}, Q_t^{[i]} \right\}_{i=1\ldots N, t=1\ldots T}$$

One data-point per state-action pair

Explore in action space at each time step with stochastic low-level policy

$$\boldsymbol{a}_t \sim \pi(\cdot | \boldsymbol{s}; \boldsymbol{\theta})$$

Summary

Step-based:

- Exploration in Action Space
- Less variance in quality assessment.
- More data-points to fit policy
- Less likely to create unstable policies

Uses the structure of the RL problem

- **decomposition in single timesteps**

Episode-based:

- Exploration in Parameter Space
- Allows for more sophisticated exploration strategies
- Is often very efficient for a small amount of parameters
- Generalization and multi-task learning
- E.g. open loop policies such as DMPs

Structure-less optimization

- **„Black-Box Optimizer“**

Episode-based Policy Search

We learn a search distribution $\pi(\boldsymbol{\theta};\boldsymbol{\omega})$ over the parameters of the low-level control policy $\pi(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{\theta})$

$\pi(\boldsymbol{\theta};\boldsymbol{\omega})$ is called **upper-level policy**

For example, $\pi(\boldsymbol{\theta};\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$

$\boldsymbol{\omega} = \{\boldsymbol{\mu},\boldsymbol{\Sigma}\}$ … parameters of upper level policy

To reduce variance in the returns, $\pi(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{\theta})$ is often modelled as determinstic policy, i.e.,

$$\pi(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{\theta}) \rightarrow \boldsymbol{a} = \pi(\boldsymbol{s})$$

Episode-based Policy Search

Search for policy $\pi(\boldsymbol{\theta}; \boldsymbol{\omega})$ that maximizes the expected return

$$J_{\boldsymbol{\omega}} = \int \pi(\boldsymbol{\theta}; \boldsymbol{\omega}) R_{\boldsymbol{\theta}} d\boldsymbol{\theta}$$

Upper-Level Policy $\pi(\boldsymbol{\theta}; \boldsymbol{\omega})$:

Stochastic, chooses parameters of low-level policy / movement primitive

Implements exploration in parameter space for information gathering

Return $R_{\boldsymbol{\theta}}$: Expected long-term reward for the trajectory $\boldsymbol{\tau}$ that corresponds to $\boldsymbol{\theta}$

$$R_{\boldsymbol{\theta}} = \mathbb{E}\left[\sum_{t=1}^{T} r_t | \boldsymbol{\theta}\right]$$

Episode-based Policy Search Algorithms

Policy Search Algorithms:

Given: initial upper level policy $\pi(\boldsymbol{\theta}; \boldsymbol{\omega}_0)$
Repeat until convergence

Exploration:

Sample from stochastic policy:

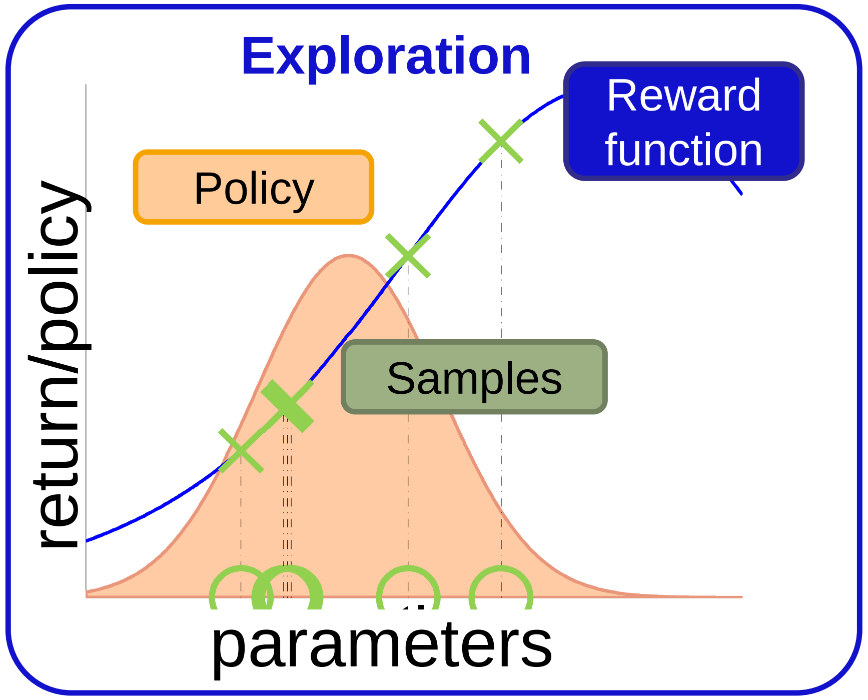
$$\boldsymbol{\theta}_i \sim \pi(\boldsymbol{\theta}; \boldsymbol{\omega}_k), i = 1 \dots N$$

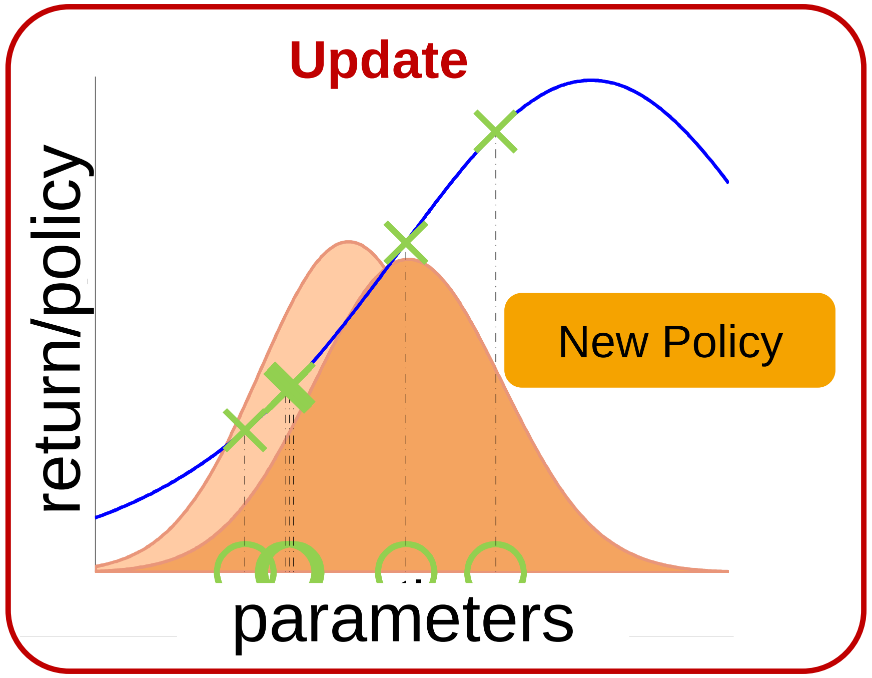
Collect returns by executing $\boldsymbol{\theta}_i$

$$R_i = R_{\boldsymbol{\theta}_i}$$

Update:

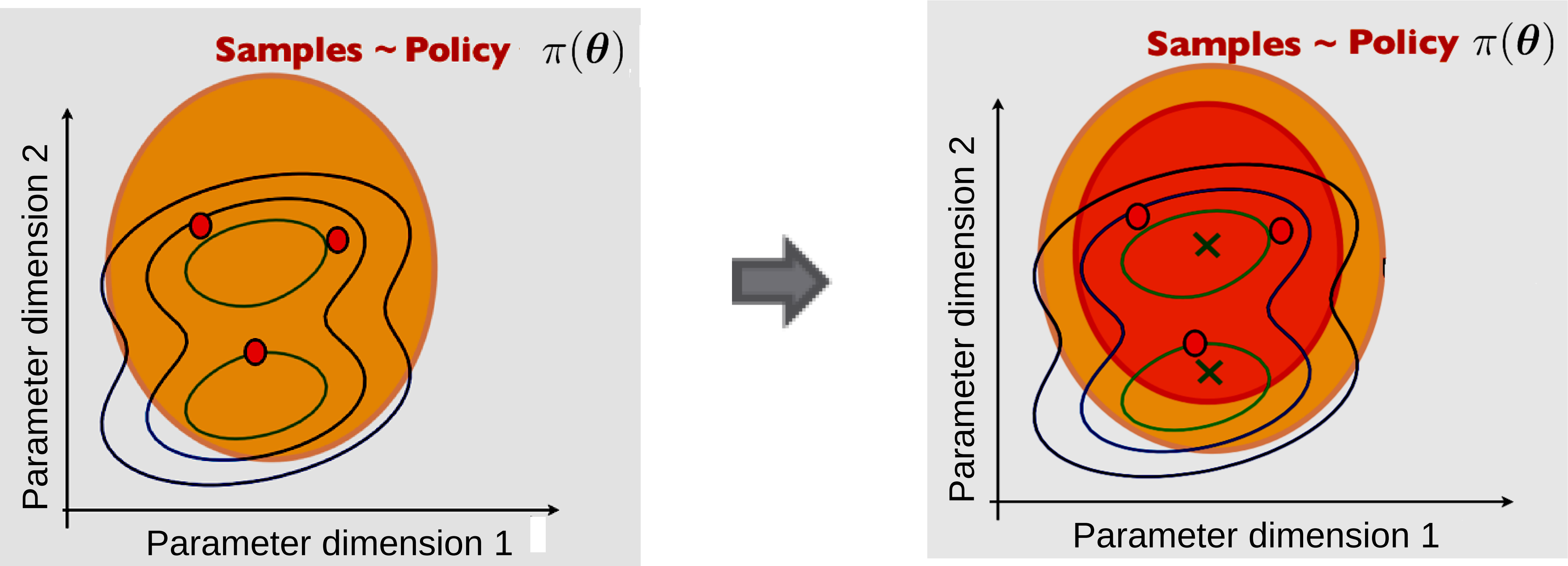
Obtain new policy $\pi(\boldsymbol{\theta}; \boldsymbol{\omega}_{k+1})$ from samples





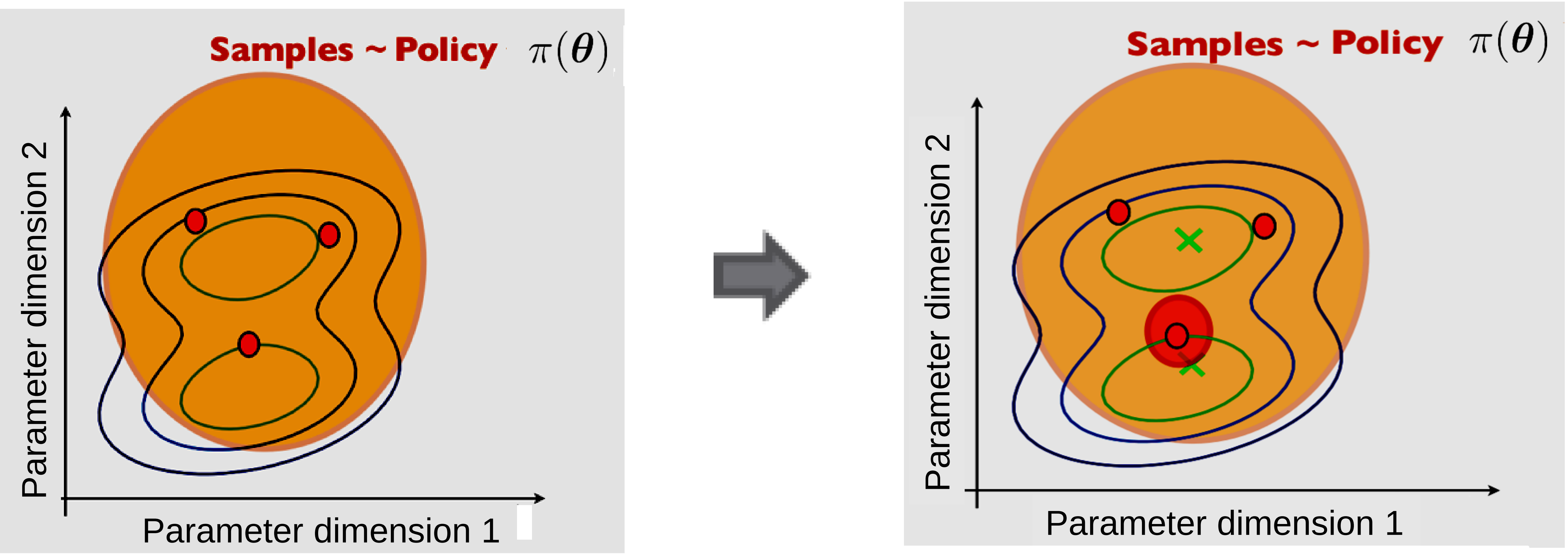
Exploration versus Exploitation

How should we update the policy?


Samples ~ Policy $\pi(\boldsymbol{\theta})$
Parameter dimension 2
Parameter dimension 1
Samples ~ Policy $\pi(\boldsymbol{\theta})$
Parameter dimension 2
Parameter dimension 1

Exploration versus Exploitation

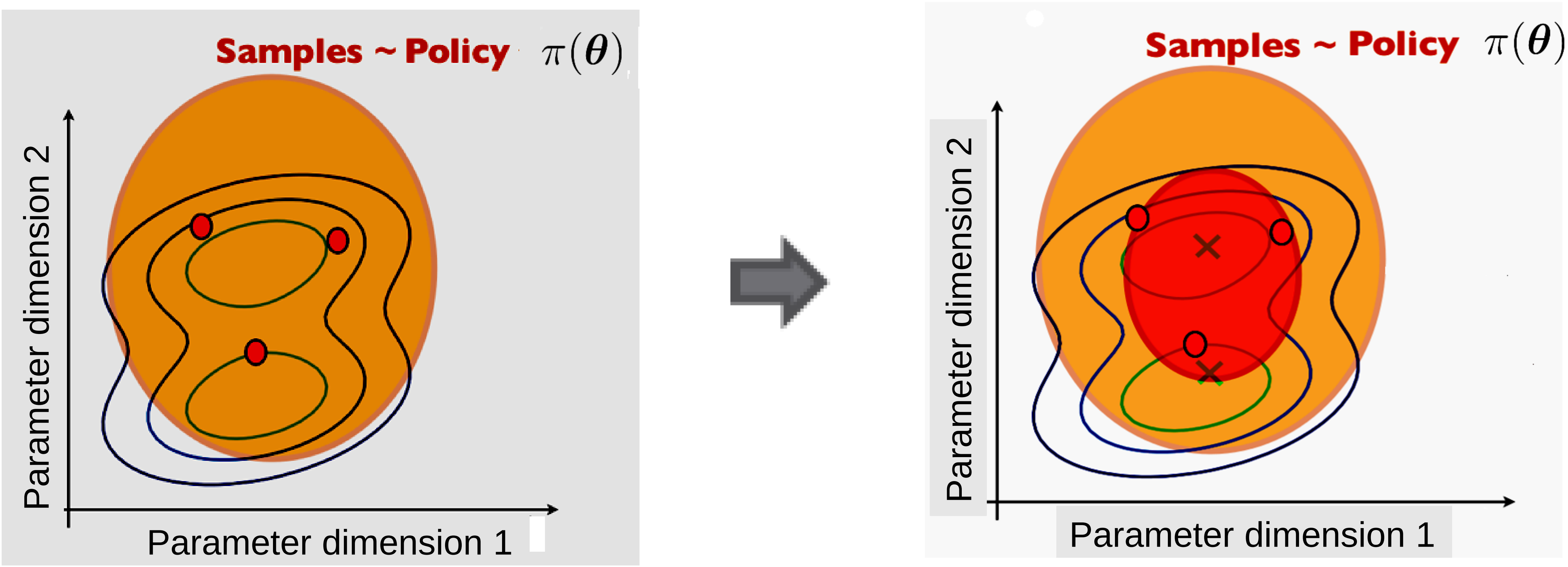
How greedily should we update the policy?


Samples ~ Policy $\pi(\boldsymbol{\theta})$
Parameter dimension 2
Parameter dimension 1
Samples ~ Policy $\pi(\boldsymbol{\theta})$
Parameter dimension 2
Parameter dimension 1

Exploration versus Exploitation

How greedily should we update the policy?

- **How can we control this update?**



- **We need to find a metric to measure the „distance“ between two policies**

Exploration versus Exploitation

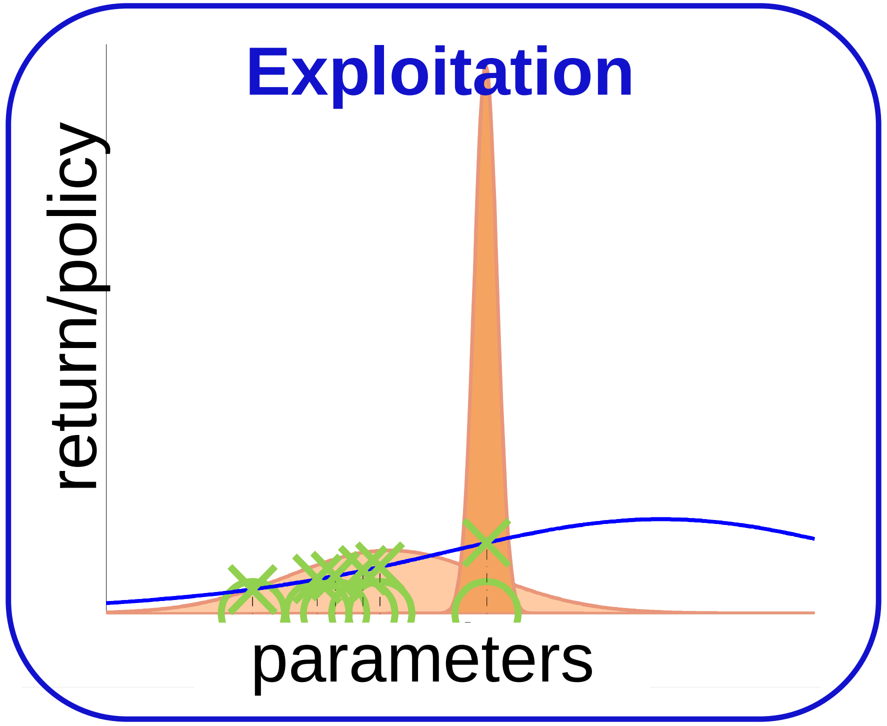
We have to choose a tradeoff between

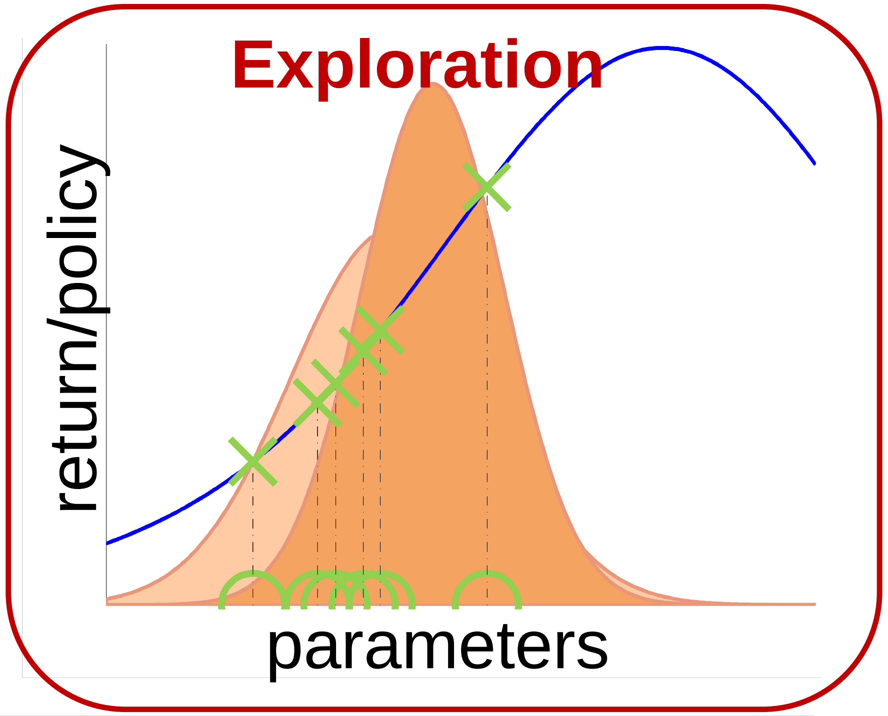
- **Exploitation:** Maximizes reward on the samples
- **Exploration:** Continue to explore in the next iteration

Fundamental Question in Policy Search

- How can we control the trade-off between exploration and exploitation?
- We need to quantify the difference between two policies
- We will get to know different metrics for policies

Typically, we want to limit the distance between two subsequent policies for the update



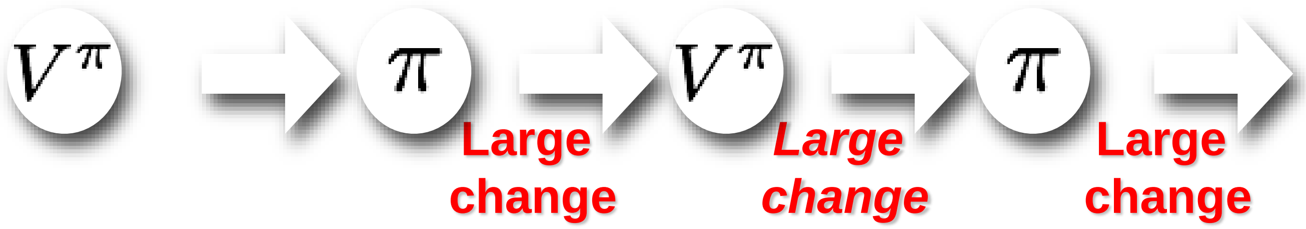


Greedy vs Incremental

Why is it useful to control the step-width of the policy update?

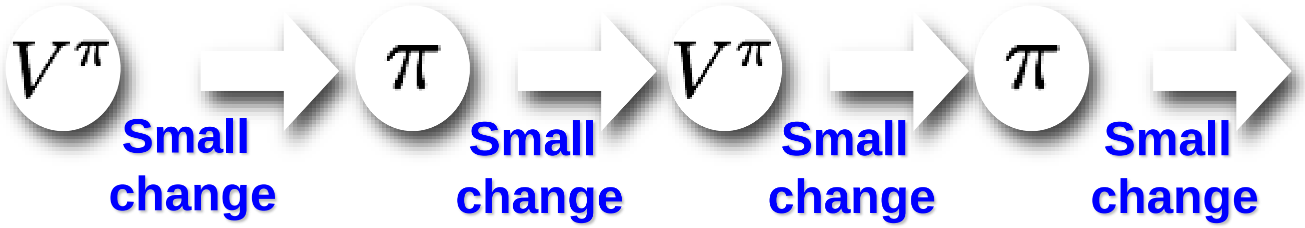
Greedy Updates:

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



potentially unstable learning process with large policy jumps

Policy Search Updates:



stable learning process with smooth policy improvement

Outline of the Lecture

1. Categorization of Policy Search

  I. Episode-Based versus Step-Based Policy Search

2. Policy Gradients

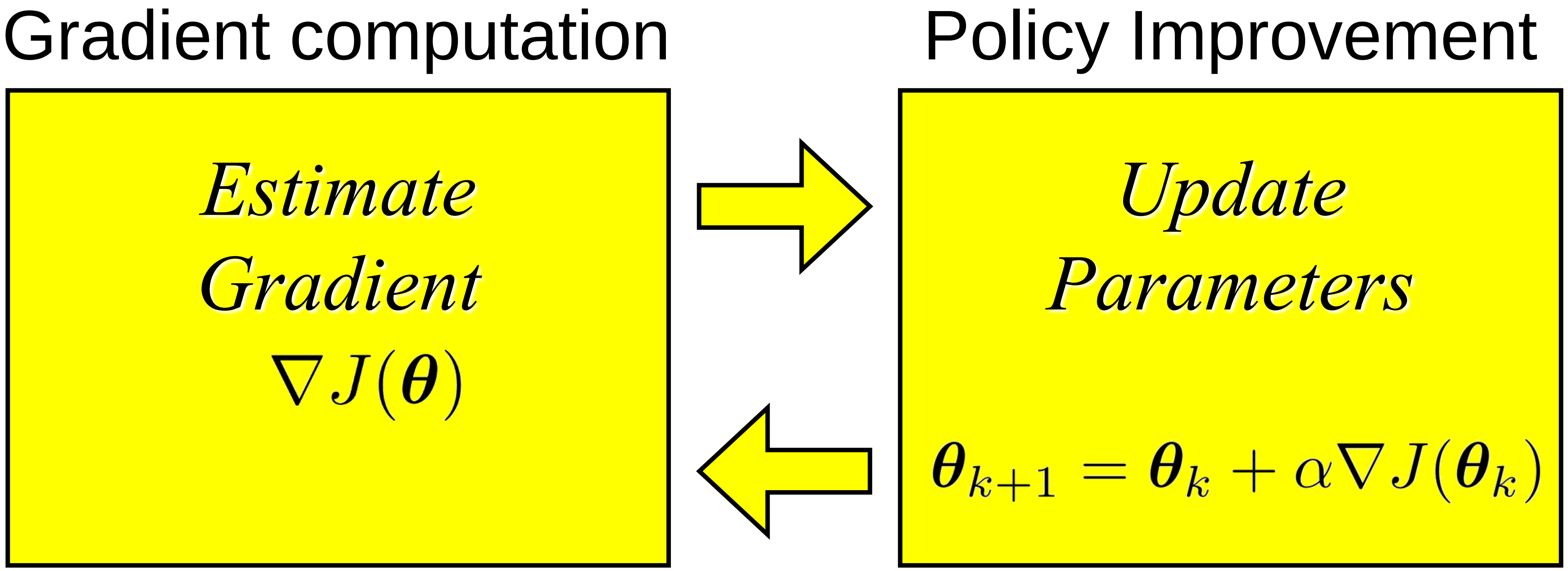
  I. Episode-Based Policy Gradients

  II. Step-Based Policy Gradients

3. Relative Entropy and Natural Gradients

[Deisenroth, Neumann, Peters: „A survey on Policy Search in Robotics“, 2013]

Gradient-based Policy Updates



Gradient computation

Policy Improvement

Estimate Gradient

$\nabla J(\boldsymbol{\theta})$

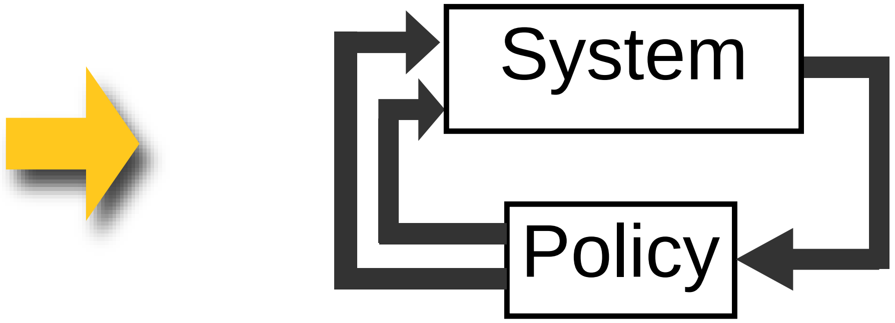
Update Parameters

$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha \nabla J(\boldsymbol{\theta}_k)$

Finite Differences

1. Perturb the parameters of your policy:

$$\theta + \delta\theta$$



$$\delta J = J(\boldsymbol{\theta} + \delta\boldsymbol{\theta}) - J(\boldsymbol{\theta})$$

2. Approximate J by first order Taylor approximation

$$J(\boldsymbol{\theta} + \delta\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \frac{\partial J(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}\delta\boldsymbol{\theta}$$

3. Solve for $\frac{\partial J(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}$ in a least squares sense (linear regression):

$$\nabla_{\boldsymbol{\theta}}^{\mathrm{FD}} J = \frac{\partial J(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}} = (\Delta\boldsymbol{\Theta}^T\Delta\boldsymbol{\Theta})^{-1}\Delta\boldsymbol{\Theta}^T\Delta\boldsymbol{J}$$

Can be used to update a single parameter estimate θ (e.g. mean)

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

Likelihood Policy Gradients

How can we update a distribution $\pi(\theta;\omega)$ over the parameter vector (including variance)? **Log-ratio trick**

$$\nabla \log f(x) = \frac{1}{f(x)} \nabla f(x) \quad \Rightarrow \quad \nabla f(x) = f(x) \nabla \log f(x)$$

Gradient of the expected return

$$\begin{aligned}
\nabla_{\boldsymbol{\omega}} J_{\boldsymbol{\omega}} = \nabla_{\boldsymbol{\omega}} \int \pi(\boldsymbol{\theta};\boldsymbol{\omega}) R_{\boldsymbol{\theta}} d\boldsymbol{\theta} &= \int \nabla_{\boldsymbol{\omega}} \pi(\boldsymbol{\theta};\boldsymbol{\omega}) R_{\boldsymbol{\theta}} d\boldsymbol{\theta} \\
&= \int \pi(\boldsymbol{\theta};\boldsymbol{\omega}) \nabla_{\boldsymbol{\omega}} \log \pi(\boldsymbol{\theta};\boldsymbol{\omega}) R_{\boldsymbol{\theta}} d\boldsymbol{\theta} \\
&\approx \sum_{i=1}^{N} \nabla_{\boldsymbol{\omega}} \log \pi(\boldsymbol{\theta}_i;\boldsymbol{\omega}) R_i
\end{aligned}$$

Only needs samples!

This gradient is called **Parameter Exploring Policy Gradient (PGPE)**

Baselines…

We can always **subtract a baseline** from the gradient…

$$\nabla_{\boldsymbol{\omega}} J_{\boldsymbol{\omega}} = \sum_{i=1}^{N} \nabla_{\boldsymbol{\omega}} \log \pi(\boldsymbol{\theta}_i; \boldsymbol{\omega})(R_i - b)$$

Why?

The gradient estimate can have a high variance

Subtracting a baseline can reduce the variance

Its still unbiased…

$$\mathbb{E}_{p(\boldsymbol{x};\boldsymbol{\omega})}[\nabla_{\boldsymbol{\omega}} \log p(\boldsymbol{x};\boldsymbol{\omega}) b] = b \int \nabla_{\boldsymbol{x}} p(\boldsymbol{x};\boldsymbol{\omega}) = b \nabla_{\boldsymbol{x}} \int p(\boldsymbol{x};\boldsymbol{\omega}) = 0$$

Good baseline: Average reward

but there are optimal baselines for each alg. that minimize the variance

Step-based Policy Gradient Methods

The returns can still have a lot of variance

$$R_{\boldsymbol{\theta}} = \mathbb{E}\left[\sum_{t=1}^{T} r_t | \boldsymbol{\theta}\right]$$

- It is the sum over T random variables

There is less variance in the rewards to come: $Q_t^{[i]} = \sum_{h=t}^{T} r_h^{[i]}$

- Step-based algorithms can be more efficient when estimating the gradient

- For step-based algorithms, we have to compute the gradient $\nabla_{\boldsymbol{\theta}} J$ for the low-level policy $\pi(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{\theta})$

Outline of the Lecture

1. Categorization of Policy Search

I. Episode-Based versus Step-Based Policy Search

2. Policy Gradients

I. Episode-Based Policy Gradients

II. Step-Based Policy Gradients

3. Relative Entropy and Natural Gradients

[Deisenroth, Neumann, Peters: „A survey on Policy Search in Robotics“, 2013]

Step-based Policy Gradient Methods

Some more basic notation

Trajectory distribution: $p(\boldsymbol{\tau};\boldsymbol{\theta}) = p(\boldsymbol{s}_1)\prod_{t=1}^{T-1}\pi(\boldsymbol{a}_t|\boldsymbol{s}_t;\boldsymbol{\theta})p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t,\boldsymbol{a}_t)$

Return for a single trajectory: $R(\boldsymbol{\tau}) = \sum_{t=1}^{T-1} r_t + r_T$

Expected long term reward $J(\boldsymbol{\theta})$ can be written as expectation over the trajectory distribution

$$J(\boldsymbol{\theta}) = \mathbb{E}_{p(\boldsymbol{\tau};\boldsymbol{\theta})}[R(\boldsymbol{\tau})] = \int p(\boldsymbol{\tau};\boldsymbol{\theta})R(\boldsymbol{\tau})d\tau$$

Step-Based Likelihood Ratio Gradient

Instead of computing the gradient of the upper-level policy, we compute the gradient of the trajectory distribution

$$\nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}} = \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}^{[i]}; \boldsymbol{\theta}) R(\boldsymbol{\tau}^{[i]})$$

How do we compute $\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}^{[i]}; \boldsymbol{\theta})$?

$$p(\boldsymbol{\tau}; \boldsymbol{\theta}) = p(\boldsymbol{s}_1) \prod_{t=1}^{T-1} \pi(\boldsymbol{a}_t | \boldsymbol{s}_t; \boldsymbol{\theta}) p(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t)$$

$$\log p(\boldsymbol{\tau}; \boldsymbol{\theta}) = \sum_{t=1}^{T-1} \log \pi(\boldsymbol{a}_t | \boldsymbol{s}_t; \boldsymbol{\theta}) + \mathrm{const}$$

Model-dependent terms do not depend on parameters, derivative is now easy

$$\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}; \boldsymbol{\theta}) = \sum_{t=1}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_t | \boldsymbol{s}_t; \boldsymbol{\theta})$$

Lets plug it in…

Result:

$$
\begin{aligned}
\nabla_{\boldsymbol{\theta}} J &= \sum_{i=1}^{N} \sum_{t=1}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_t^{[i]} | \boldsymbol{s}_t^{[i]}; \boldsymbol{\theta}) R(\boldsymbol{\tau}^{[i]}) \\
&= \sum_{i=1}^{N} \sum_{t=1}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_t^{[i]} | \boldsymbol{s}_t^{[i]}; \boldsymbol{\theta}) \left(\sum_{h=1}^{T-1} r_h^{[i]} + r_T^{[i]} \right)
\end{aligned}
$$

This algorithm is called the REINFORCE Policy Gradient

- Wait... we still use the returns $R(\boldsymbol{\tau}) = \sum_{t=1}^{T-1} r_t + r_T$ (high variance)
- What did we gain with our step-based version? Not too much yet...

Using the rewards to come…

Simple Observation:

Rewards in the past are not correlated with actions in the future

$$\mathbb{E}_{p(\boldsymbol{\tau})}[r_t \log \pi(\boldsymbol{a}_h|\boldsymbol{s}_h)] = 0, \forall t < h$$

This observation leads to the Policy Gradient Theorem

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}^{\mathrm{PG}} J &= \sum_{i=1}^{N}\sum_{t=1}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_t^{[i]}|\boldsymbol{s}_t^{[i]};\boldsymbol{\theta})\left(\sum_{h=t}^{T-1} r_h^{[i]} + r_T^{[i]}\right)\\ &= \sum_{i=1}^{N}\sum_{t=1}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_t^{[i]}|\boldsymbol{s}_t^{[i]};\boldsymbol{\theta}) Q_h^{[i]}\end{aligned}$$

- The rewards to come have less variance
- We can do it again with a baseline...

Metric in standard gradients

How can we choose the step-size to control our policy update?

Simple (naive) idea:

- Use distance in parameter space as metric
- Episode-based: $L_2(\pi_{k+1}, \pi_k) = ||\boldsymbol{\omega}_{k+1} - \boldsymbol{\omega}_k||$
- Step-based: $L_2(\pi_{k+1}, \pi_k) = ||\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k||$

Choose step size, such that $L_2(\pi_{k+1}, \pi_k) \leq \epsilon$

$$\alpha_k = \frac{1}{||\nabla J||}\epsilon$$

Metric in standard gradients

Is the distance in parameter space a good idea?

Consider the following policy:

$$\pi(a|\boldsymbol{s};\boldsymbol{\theta}) = \mathcal{N}(a|\theta_1 s_1 + \theta_2 s_2, \sigma^2)$$

with $s_1 \in [0,1]$ and $s_2 \in [0,1000]$

Lets consider the distances of $\boldsymbol{\theta}_1 = [1,1]^T, \quad \boldsymbol{\theta}_2 = [1.1, 1]^T, \boldsymbol{\theta}_3 = [1, 1.1]^T$

The distances $||\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2||$ and $||\boldsymbol{\theta}_1 - \boldsymbol{\theta}_3||$ are the same

Policy $\pi(a|\boldsymbol{s},\boldsymbol{\theta}_3)$ is much more different from $\pi(a|\boldsymbol{s},\boldsymbol{\theta}_1)$ than $\pi(a|\boldsymbol{s},\boldsymbol{\theta}_2)$

The euclidian metric is not invariant to scaling of the variables!

Metric in standard gradients

Can we define a metric that is invariant to transformation of the parameters?

Idea:

- Define a matrix M that captures the „influence“ of the parameters on the policy
- Use matrix M to define a new metric that incorporates this influence
- $$L_M(\pi_{k+1}, \pi_k) = ||\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k||_{\boldsymbol{M}} = (\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k)^T \boldsymbol{M} (\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k)$$
- Large change in parameters are more expensive in directions with large influence

Metric in standard gradients

How to use such metric for gradient ascent?

Find an update direction $\Delta\boldsymbol{\theta}$ that is most similar to the standard gradient $\nabla_{\boldsymbol{\theta}} J = [\frac{dJ}{d\theta_1}, \dots, \frac{dJ}{d\theta_n}]$

$$\Delta\boldsymbol{\theta}^* = \mathrm{argmax}_{\Delta\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}} J \Delta\boldsymbol{\theta}$$

with limited distance, i.e.,

$$\text{s.t.:} \quad L_M(\pi_{k+1}, \pi_k) = \Delta\boldsymbol{\theta}^T \boldsymbol{M} \Delta\boldsymbol{\theta} \leq \epsilon$$

Solution to this constraint optimization problem (see lecture notes)

$$\Delta\boldsymbol{\theta}^* = \lambda \boldsymbol{M}^{-1} \nabla_{\boldsymbol{\theta}} J \propto \boldsymbol{M}^{-1} \nabla_{\boldsymbol{\theta}} J$$

➡ Now we „only“ have to find a proper matrix M

Outline of the Lecture

1. Categorization of Policy Search

I. Episode-Based versus Step-Based Policy Search

2. Policy Gradients

I. Episode-Based Policy Gradients

II. Step-Based Policy Gradients

3. Relative Entropy and Natural Gradients

[Deisenroth, Neumann, Peters: „A survey on Policy Search in Robotics“, 2013]

We need to find a better metric…

What do we want?

1. Invariance to the representation of the policy (e.g. parameter transformations)

2. Invariance to transformations of the rewards

Alternative way to measure the distance between two policies

Policies are probabilty distributions

We can measure „distances“ of distributions

For example, Relative Entropy or Kullback-Leibler divergence

$$\mathrm{KL}(p||q) = \sum_{\boldsymbol{x}} p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}$$

Information-theoretic „distance“ measure between distributions

Kullback-Leibler Divergence

Properties: $\mathrm{KL}(p||q) = \sum_{\boldsymbol{x}} p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}$

$$\mathrm{KL}(q||p) \geq 0, \quad \mathrm{KL}(q||p) = 0 \Rightarrow p = q$$

Not symetric, so not a real distance

$$\mathrm{KL}(q||p) \neq \mathrm{KL}(p||q)$$

KL for Gaussians:

$$\mathrm{KL}(p||q) = \log \frac{|\boldsymbol{B}|}{|\boldsymbol{A}|} + \mathrm{tr}(\boldsymbol{B}^{-1}\boldsymbol{A}) + (\boldsymbol{b} - \boldsymbol{a})^T \boldsymbol{B}^{-1}(\boldsymbol{b} - \boldsymbol{a}) - n$$

with $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{a}, \boldsymbol{A})$ and $q(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{b}, \boldsymbol{B})$

... compare with euclidian metric:

Distance scales with inverse covariance matrix of q

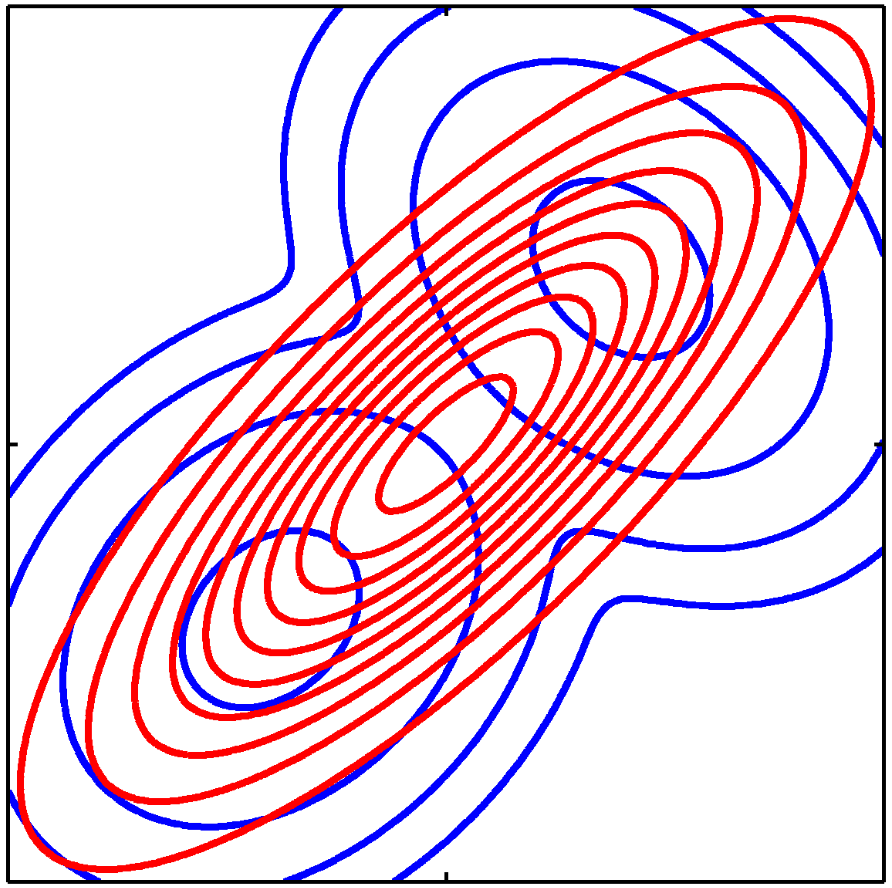
Kullback-Leibler Divergence

2 types of KL:

Moment projection: $\text{argmin}_q \text{KL}(p||q) = \sum_{\boldsymbol{x}} p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}$

q is large whereever p is large

Same as **Maximum Likelihood** estimate (blackboard)!



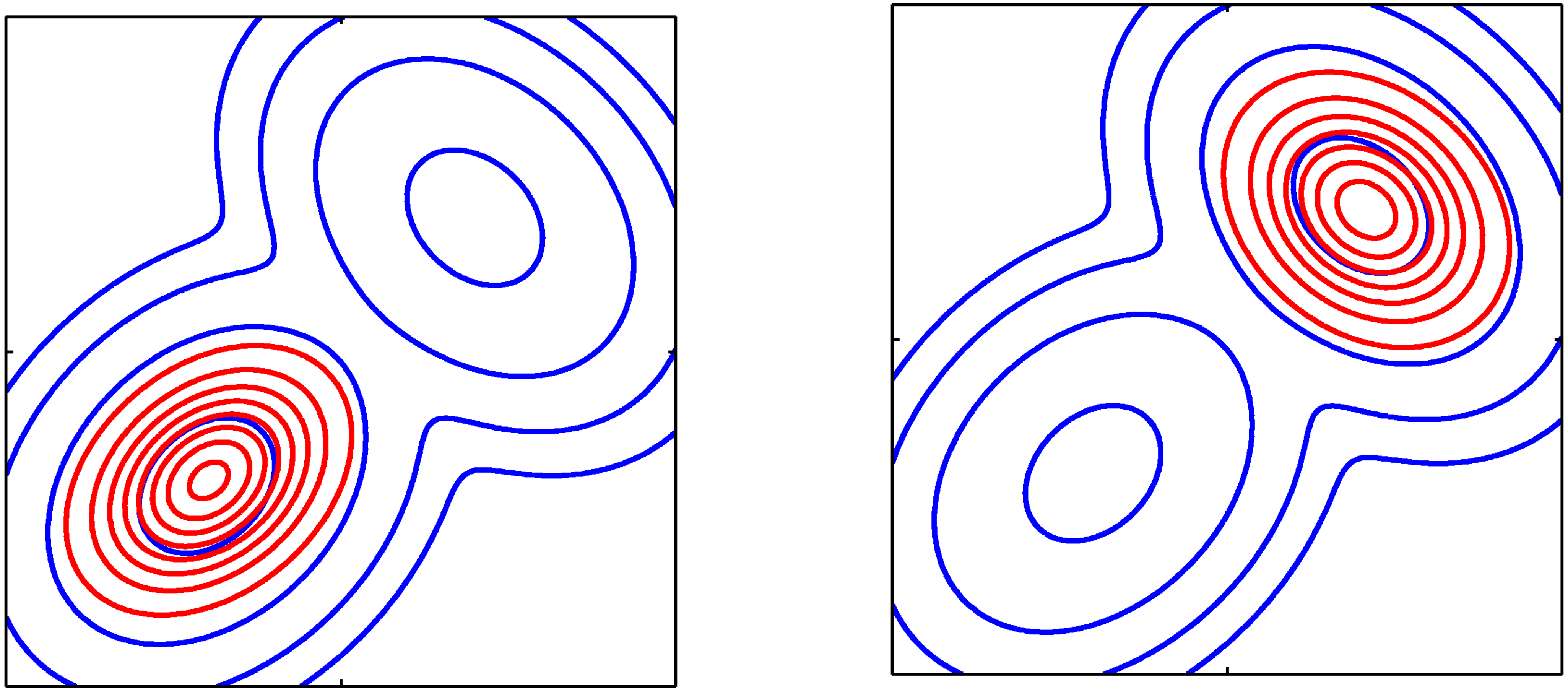
Kullback-Leibler Divergence

2 types of KL:

Information projection: $\mathrm{argmin}_q \mathrm{KL}(q||p) = \sum_{\boldsymbol{x}} q(\boldsymbol{x}) \log \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})}$

q is zero whereever p is zero (zero forcing)

not unique for most distributions



KL divergences and the Fisher information matrix

The Kullback Leibler divergence can be **approximated by the Fisher information matrix (2nd order Taylor approximation)**

$$\mathrm{KL}(p_{\boldsymbol{\theta}+\Delta\boldsymbol{\theta}}||p_{\boldsymbol{\theta}}) \approx \Delta\boldsymbol{\theta}^T \boldsymbol{G}(\boldsymbol{\theta})\Delta\boldsymbol{\theta}$$

where $\boldsymbol{G}(\boldsymbol{\theta})$ is the **Fisher information matrix (FIM)**

$$\boldsymbol{G}(\boldsymbol{\theta}) = \mathbb{E}_p[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x})\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x})^T]$$

- Captures information how the **single parameters influence the distribution**

Properties of the Fisher information matrix

$$\boldsymbol{G}(\boldsymbol{\theta}) = \mathbb{E}_p[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x})^T]$$

If the distribution is Gaussian, i.e., $p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{\boldsymbol{\alpha}}, \boldsymbol{\Sigma}_{\boldsymbol{\beta}})$
with $\boldsymbol{\alpha} \in \mathbb{R}^n$ and $\boldsymbol{\beta} \in \mathbb{R}^m$

then the FIM is a $(n+m) \times (n+m)$ matrix and is given by

$G(\boldsymbol{\theta}) = \mathrm{diag}\big(\boldsymbol{G}_1(\boldsymbol{\alpha}), \boldsymbol{G}_2(\boldsymbol{\beta})\big)$ with $\boldsymbol{\theta} = [\boldsymbol{\alpha}^T, \boldsymbol{\beta}^T]^T$

$$\boldsymbol{G}_1(\boldsymbol{\alpha})_{i,j} = \frac{\partial \boldsymbol{\mu}_{\boldsymbol{\alpha}}}{\partial \boldsymbol{\alpha}_i} \boldsymbol{\Sigma}^{-1} \frac{\partial \boldsymbol{\mu}_{\boldsymbol{\alpha}}}{\partial \boldsymbol{\alpha}_j}^T$$

$$\boldsymbol{G}_2(\boldsymbol{\beta}) = 0.5\mathrm{tr}\left(\boldsymbol{\Sigma}_{\boldsymbol{\beta}}^{-1} \frac{\partial \boldsymbol{\Sigma}_{\boldsymbol{\beta}}}{\partial \boldsymbol{\beta}_i} \boldsymbol{\Sigma}_{\boldsymbol{\beta}}^{-1} \frac{\partial \boldsymbol{\Sigma}_{\boldsymbol{\beta}}}{\partial \boldsymbol{\beta}_j}\right)$$

Homework: Check G_1 for $\boldsymbol{\mu}_{\boldsymbol{\alpha}} = \boldsymbol{\alpha}$ and $\boldsymbol{\mu}_{\boldsymbol{\alpha}} = \boldsymbol{\phi}(\boldsymbol{s})^T \boldsymbol{\alpha}$

Kullback Leibler divergences

The Natural gradient uses the Fisher information matrix as metric

$$\nabla_{\boldsymbol{\theta}}^{\mathrm{NG}} J = \mathrm{argmax}_{\Delta\boldsymbol{\theta}} \Delta\boldsymbol{\theta}^T \nabla_{\boldsymbol{\theta}} J$$

$$\text{s.t.:} \quad \mathrm{KL}(p_{\boldsymbol{\theta}+\Delta\boldsymbol{\theta}} || p_{\boldsymbol{\theta}}) \approx \Delta\boldsymbol{\theta}^T \boldsymbol{G}(\boldsymbol{\theta}) \Delta\boldsymbol{\theta} \leq \epsilon$$

The solution to this optimization problem is given as:

$$\nabla_{\boldsymbol{\theta}}^{\mathrm{NG}} J \propto G(\boldsymbol{\theta})^{-1} \nabla_{\boldsymbol{\theta}} J$$

As every parameter has the same influence under metric M, the natural gradient is invariant to linear transformations of the parameter space!

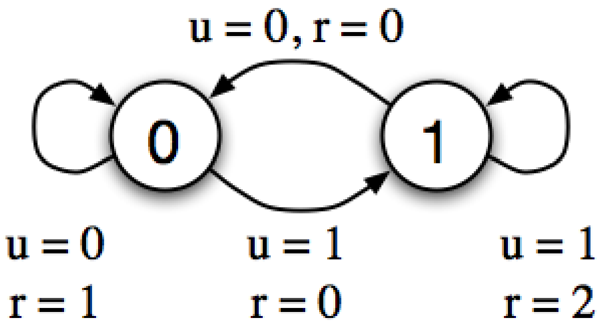
(Amari, 1998)

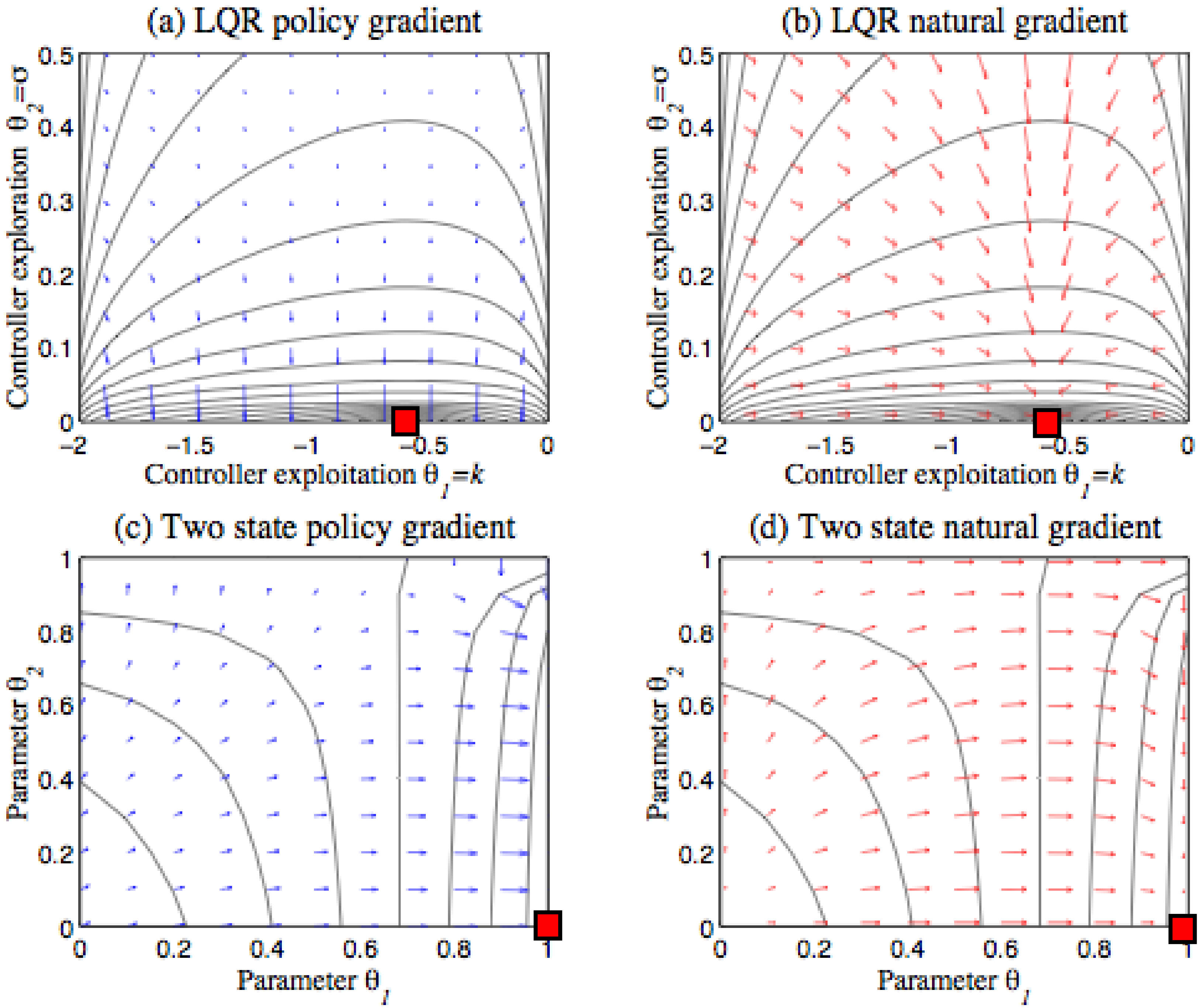
Are they useful?

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





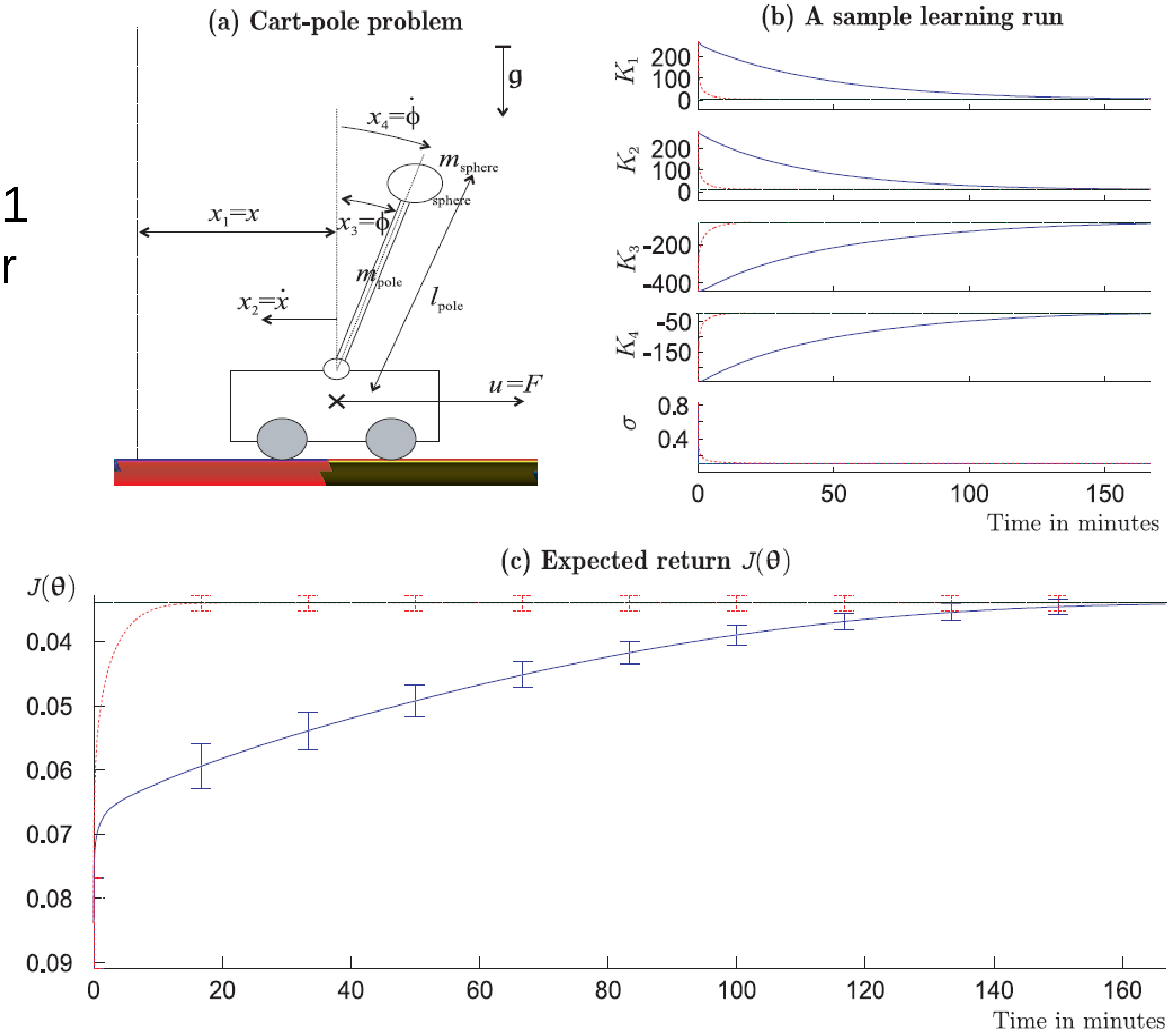
(Peters et al. 2003, 2005)

The standard gradient reduces the exploration too quickly!

Comparison

Cart-Pole Balancing

- Learn 4 gains and 1 variance parameter
- Blue: standard gradient
- Red: natural gradient


(a) Cart-pole problem

(b) A sample learning run

(c) Expected return $J(\theta)$

Computing the NG (episode-based)

In the episode-based case, if the distribution is Gaussian, the Fisher Information matrix can be computed in closed form

Used by the Natural Evolution Strategy (NES)

```
Initialize: $\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0$
For  k = 0 to L
    Create evaluate samples: $\boldsymbol{\theta}_i \sim \pi(\boldsymbol{\theta}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$   $R_i = \sum_{t=1}^{T} r_{i,t}$
    Compute Gradient: $\nabla_{\boldsymbol{\omega}} J = \sum_{i=1}^{N} \nabla_{\boldsymbol{\omega}} \log \pi(\boldsymbol{\theta}_i|\boldsymbol{\omega}) R_i$
    Compute FIM $\boldsymbol{G}(\boldsymbol{\omega})$ in closed form for Gaussians
    Compute Natural Gradient: $\nabla_{\boldsymbol{\omega}}^{\mathrm{NES}} J = \boldsymbol{G}(\boldsymbol{\omega})^{-1} \nabla_{\boldsymbol{\omega}} J$
    Update Parameters: $\boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \eta \nabla_{\boldsymbol{\omega}}^{\mathrm{NES}} J$
end
```

Computing the NG (step-based)

Policy Gradient Theorem with baseline

$$\nabla_{\boldsymbol{\theta}}^{\mathrm{PG}} J = \sum_{i=1}^{N} \sum_{t=1}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_t^{[i]} | \boldsymbol{s}_t^{[i]}; \boldsymbol{\theta})(Q_h^{[i]} - b_h(\boldsymbol{s}))$$

To further improve the gradient estimate we can try to estimate the reward to come

$$f_{\boldsymbol{w}}(\boldsymbol{s}, \boldsymbol{a}) = \psi(\boldsymbol{s}, \boldsymbol{a})^T \boldsymbol{w} \approx (Q_h^{[i]} - b_h(\boldsymbol{s}^{[i]}))$$

and use $\nabla_{\boldsymbol{\theta}}^{\mathrm{FA}} J = \sum_{i=1}^{N} \sum_{t=1}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_t^{[i]} | \boldsymbol{s}_t^{[i]}; \boldsymbol{\theta}) f_w(\boldsymbol{s}^{[i]}, \boldsymbol{a}^{[i]})$ as gradient

It can be shown that this gradient is still unbiased if:

$$\psi(\boldsymbol{s}, \boldsymbol{a}) = \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a} | \boldsymbol{s})$$

Combatible Function Approximation

Compatible Function Approximation:

$$f_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a}) = \psi(\boldsymbol{s},\boldsymbol{a})^T\boldsymbol{w} \approx (Q_h^{[i]} - b_h(\boldsymbol{s}^{[i]})) \qquad \psi(\boldsymbol{s},\boldsymbol{a}) = \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}|\boldsymbol{s})$$

Basis functions of Q(s,a) are combatible to the policy

The compatible function approximation is mean-zero!

$$\mathbb{E}_{p(\boldsymbol{\tau})}\left[\nabla \log \pi(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{\theta})^T\boldsymbol{w}\right] = 0$$

Thus, it can only represent the Advantage Function:

Baseline

$$f_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a}) = \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{\theta})^T\boldsymbol{w} = Q^{\pi}(\boldsymbol{s},\boldsymbol{a}) - V^{\pi}(\boldsymbol{s})$$

The advantage function tells us, how much better an action is in comparison to the expected performance

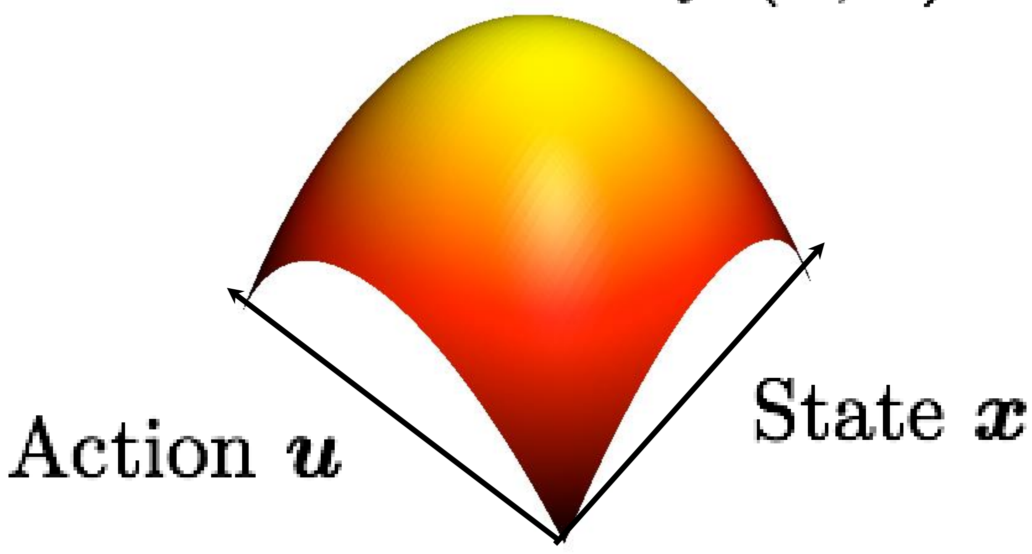
Can the Compatible FA be learned?

The compatible function approximation represents an advantage function

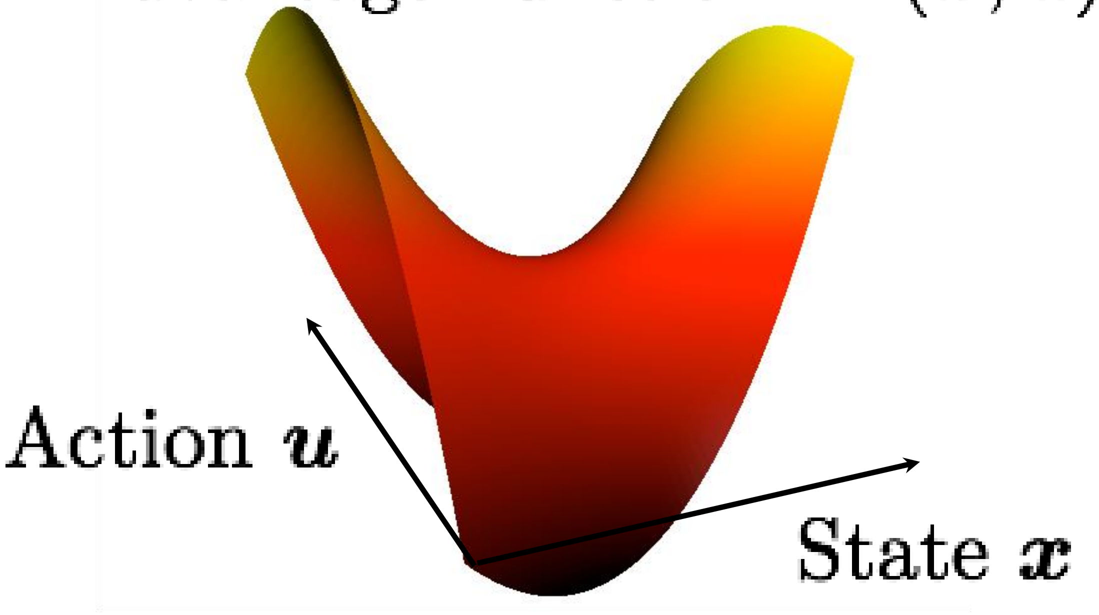
$$f_{\boldsymbol{w}}^{\pi}(\boldsymbol{x},\boldsymbol{u}) = Q^{\pi}(\boldsymbol{x},\boldsymbol{u}) - V^{\pi}(\boldsymbol{x}) = A^{\pi}(\boldsymbol{x},\boldsymbol{u}).$$

The advantage function is very different from the value functions

Value Function $Q^{\pi}(\boldsymbol{x},\boldsymbol{u})$



Advantage Function $A^{\pi}(\boldsymbol{x},\boldsymbol{u})$



In order to learn $f_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a})$ **we need to learn** $V^{\pi}(\boldsymbol{s})$

(Peters et al. 2003, 2005)

Compatible Function Approximation

Gradient with Compatible Function Approximation:

$$\nabla_{\boldsymbol{\theta}}^{\mathrm{FA}} J = \sum_{i=1}^{N} \sum_{t=1}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_t^{[i]} | \boldsymbol{s}_t^{[i]}; \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_t^{[i]} | \boldsymbol{s}_t^{[i]}; \boldsymbol{\theta})^T \boldsymbol{w}$$

$$\nabla_{\boldsymbol{\theta}}^{\mathrm{FA}} J = \mathbb{E}_{p(\tau)} \left[\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_t^{[i]} | \boldsymbol{s}_t^{[i]}; \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_t^{[i]} | \boldsymbol{s}_t^{[i]}; \boldsymbol{\theta})^T \right] \boldsymbol{w}$$

$$\nabla_{\boldsymbol{\theta}}^{\mathrm{FA}} J = \boldsymbol{F}(\boldsymbol{\theta}) \boldsymbol{w}$$

We showed [Peters & Schaal, 2008]**:**

$$\begin{aligned} \boldsymbol{F}(\boldsymbol{\theta}) &= \mathbb{E}_{p(\tau)} \left[\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_t^{[i]} | \boldsymbol{s}_t^{[i]}; \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_t^{[i]} | \boldsymbol{s}_t^{[i]}; \boldsymbol{\theta})^T \right] \\ &= \mathbb{E}_{p(\tau)} \left[\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}; \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau}; \boldsymbol{\theta})^T \right] = \boldsymbol{G}(\boldsymbol{\theta}) \end{aligned}$$

Connection to V-Function approximation

Combatible Function Approximation:

$$\nabla_{\boldsymbol{\theta}}^{\mathrm{FA}} J = \boldsymbol{F}(\boldsymbol{\theta})\boldsymbol{w}$$

We showed: F is the Fisher information matrix!

$$\boldsymbol{F}(\boldsymbol{\theta}) = \boldsymbol{G}(\boldsymbol{\theta})$$

That makes the natural gradient very simple !

$$\nabla_{\boldsymbol{\theta}}^{\mathrm{NG}} J = \boldsymbol{G}(\boldsymbol{\theta})^{-1} \nabla_{\boldsymbol{\theta}}^{\mathrm{FA}} J = \boldsymbol{G}(\boldsymbol{\theta})^{-1} F(\boldsymbol{\theta})\boldsymbol{w} = \boldsymbol{w}$$

So we just have to learn w

What about this additional FA?

In many cases, we don't have a good basis functions for $V^{\pi}(\boldsymbol{s})$

For one rollout i, if we sum up the Bellman Equations

$$Q_1^{\pi}(\boldsymbol{s}_1^{[i]}, \boldsymbol{a}_1^{[i]}) = r(\boldsymbol{s}_1^{[i]}, \boldsymbol{a}_1^{[i]}) + V_2^{\pi}(\boldsymbol{s}_2^{[i]})$$

$$V_1^{\pi}(\boldsymbol{s}_1^{[i]}) + f_{\boldsymbol{w}}(\boldsymbol{s}_1^{[i]}, \boldsymbol{a}_1^{[i]}) = r(\boldsymbol{s}_1^{[i]}, \boldsymbol{a}_1^{[i]}) + V_2^{\pi}(\boldsymbol{s}_2^{[i]})$$

$$V_1^{\pi}(\boldsymbol{s}_1^{[i]}) + \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_1^{[i]} | \boldsymbol{s}_1^{[i]}; \boldsymbol{\theta})\boldsymbol{w} = r(\boldsymbol{s}_1^{[i]}, \boldsymbol{a}_1^{[i]}) + V_2^{\pi}(\boldsymbol{s}_2^{[i]})$$

for each time step

$$V_1^{\pi}(\boldsymbol{s}_1^{[i]}) + \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_1^{[i]} | \boldsymbol{s}_1^{[i]}; \boldsymbol{\theta})\boldsymbol{w} = r(\boldsymbol{s}_1^{[i]}, \boldsymbol{a}_1^{[i]}) + V_2^{\pi}(\boldsymbol{s}_2^{[i]}) \qquad | + \text{both sides}$$

$$V_2^{\pi}(\boldsymbol{s}_2^{[i]}) + \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_2^{[i]} | \boldsymbol{s}_2^{[i]}; \boldsymbol{\theta})\boldsymbol{w} = r(\boldsymbol{s}_2^{[i]}, \boldsymbol{a}_2^{[i]}) + V_3^{\pi}(\boldsymbol{s}_3^{[i]}) \qquad | + \text{both sides}$$

$$\vdots \qquad | + \text{both sides}$$

$$V_{T-1}^{\pi}(\boldsymbol{s}_{T-1}^{[i]}) + \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_{T-1}^{[i]} | \boldsymbol{s}_{T-1}^{[i]}; \boldsymbol{\theta})\boldsymbol{w} = r(\boldsymbol{s}_{T-1}^{[i]}, \boldsymbol{a}_{T-1}^{[i]}) + V_T^{\pi}(\boldsymbol{s}_T^{[i]})$$

(Peters et al. 2003, 2005)

What about this additional FA?

We can now eliminate the values $V^{\pi}(\boldsymbol{s})$ of the intermediate states, we obtain

$$\underbrace{V^{\pi}(\boldsymbol{s}_1^{[i]})}_{J} + \underbrace{\left(\sum_{t=1}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_t^{[i]}|\boldsymbol{s}_t^{[i]};\boldsymbol{\theta})\right)}_{\boldsymbol{\varphi}^T} \boldsymbol{w} = \sum_{t=1}^{T} r(\boldsymbol{s}_t^{[i]},\boldsymbol{a}_t^{[i]})$$

ONE offset parameter J suffices as additional function approximation!

at least if we only have one initial state

(Peters et al. 2003, 2005)

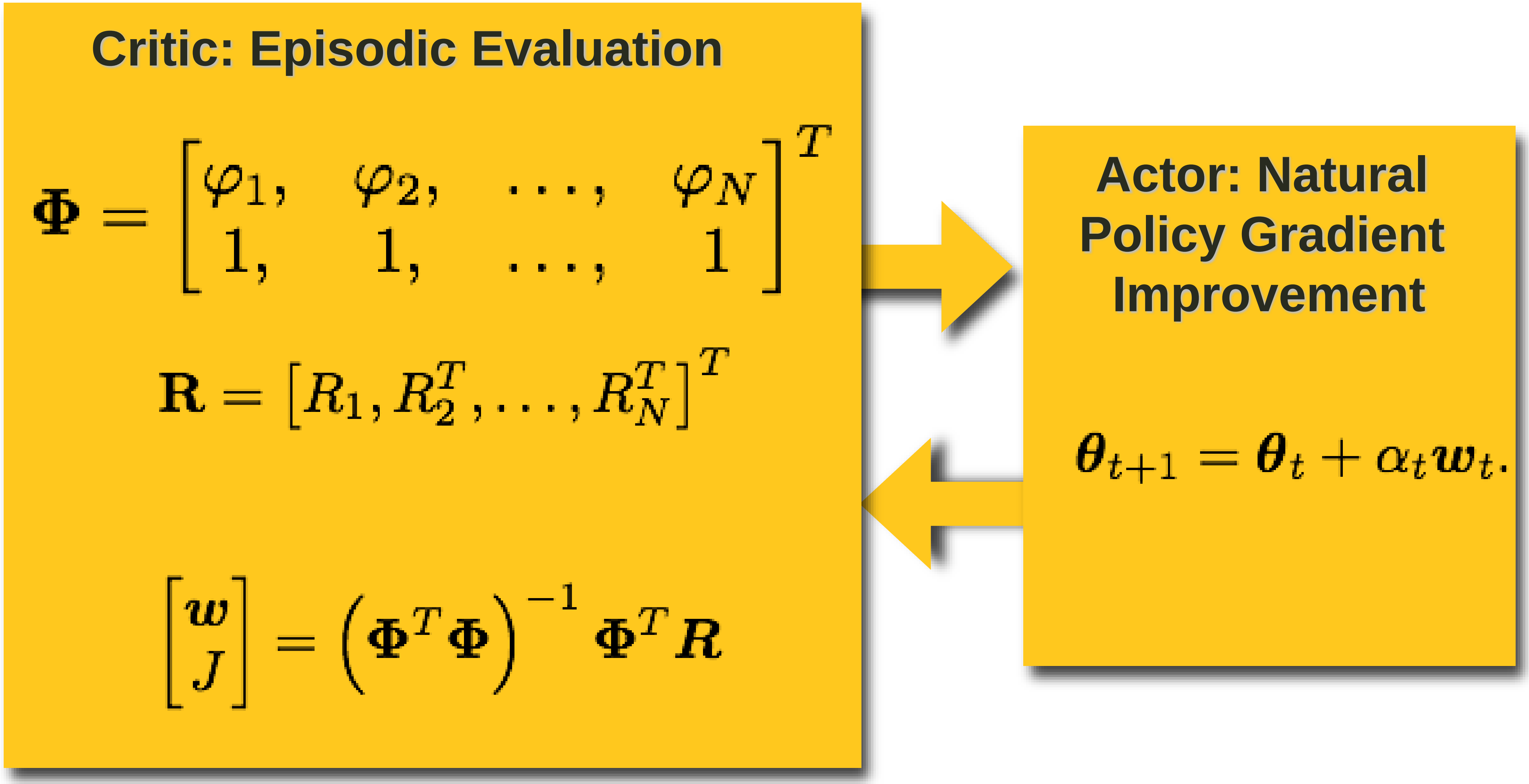
Episodic Natural Actor-Critic

In order to get $\boldsymbol{w}$, we can use linear regression

$$\underbrace{V^{\pi}(\boldsymbol{s}_1^{[i]})}_{J} + \underbrace{\left(\sum_{t=1}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_t^{[i]} | \boldsymbol{s}_t^{[i]}; \boldsymbol{\theta})\right)}_{\boldsymbol{\varphi}^T} \boldsymbol{w} = \sum_{t=1}^{T} r(\boldsymbol{s}_t^{[i]}, \boldsymbol{a}_t^{[i]})$$

Policy Evaluation

Linear Regression



Critic: Episodic Evaluation

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

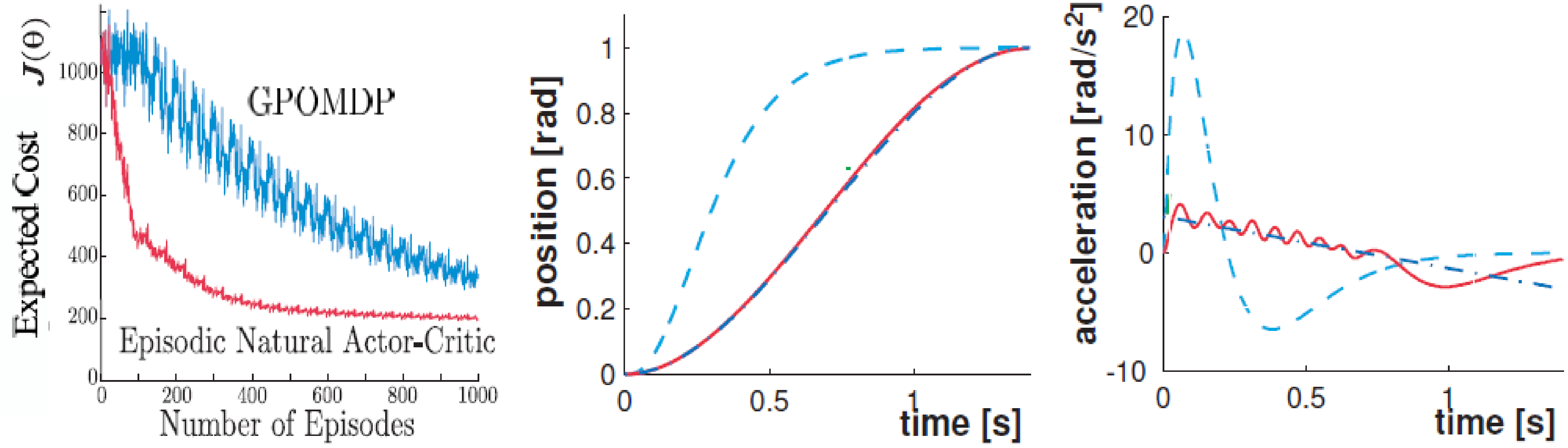
$$\mathbf{R} = [R_1, R_2^T, \ldots, R_N^T]^T$$

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

Actor: Natural Policy Gradient Improvement

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

Results…



(a) Expected Cost

(b) Position of motor primitives

(c) Controls of motor primitives

Toy Task: Optimal point to point movements with DMPs

GPOMP: Standard Gradient (Equivalent to Policy Gradient Theorem)

(Peters et al. 2003, 2005)

Learning T-Ball

1) Teach motor primitives by imitation

2) Improve movement by Episodic Natural-Actor Critic

Good performance often after 150-300 trials.

Important Points

Points worth highlighting:

- The metric really matters in policy search
- 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!

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.
- Say still need a lot of samples and we need to tune the learning rate
- Learning the learning rate is still an open problem
- Newer methods focus on probabilistic policy search approaches.