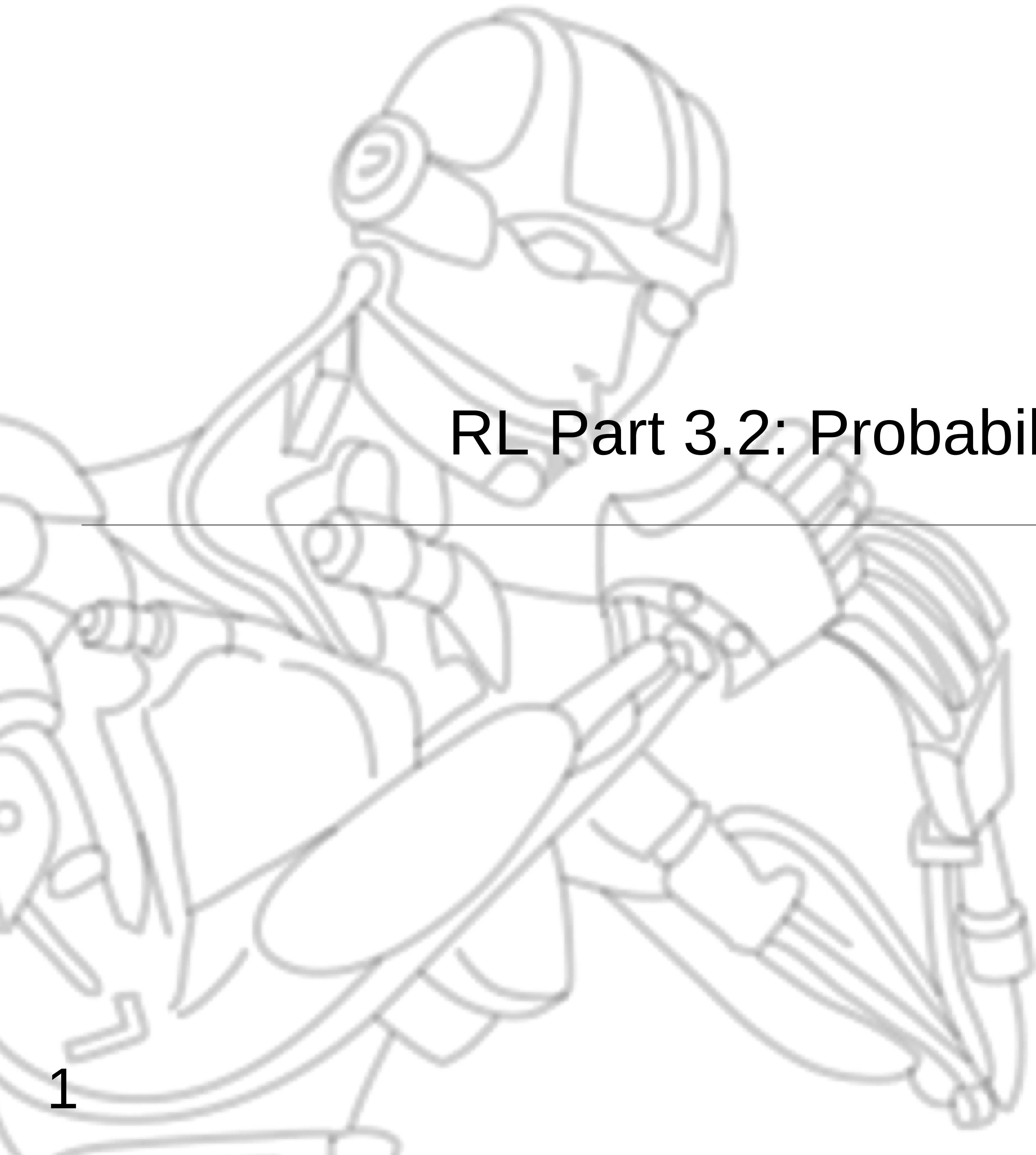
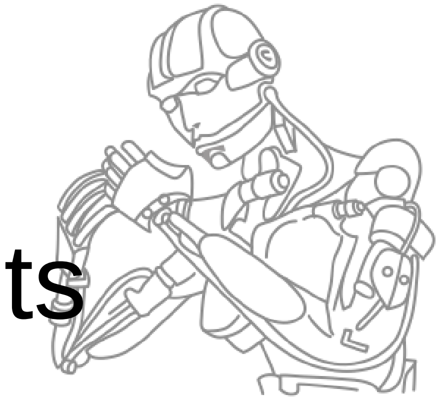


RL Part 3.2: Probabilistic Policy Search

Jan Peters
Gerhard Neumann

What we have seen from the policy gradients



- 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.
- They still need a lot of samples and **we need to tune the learning rate**

Outline of the Lecture

1. Introduction

2. Policy Updates by Weighted Maximum Likelihood

3. Relative Entropy Policy Search (REPS)

4. REPS for Contextual Policy Search

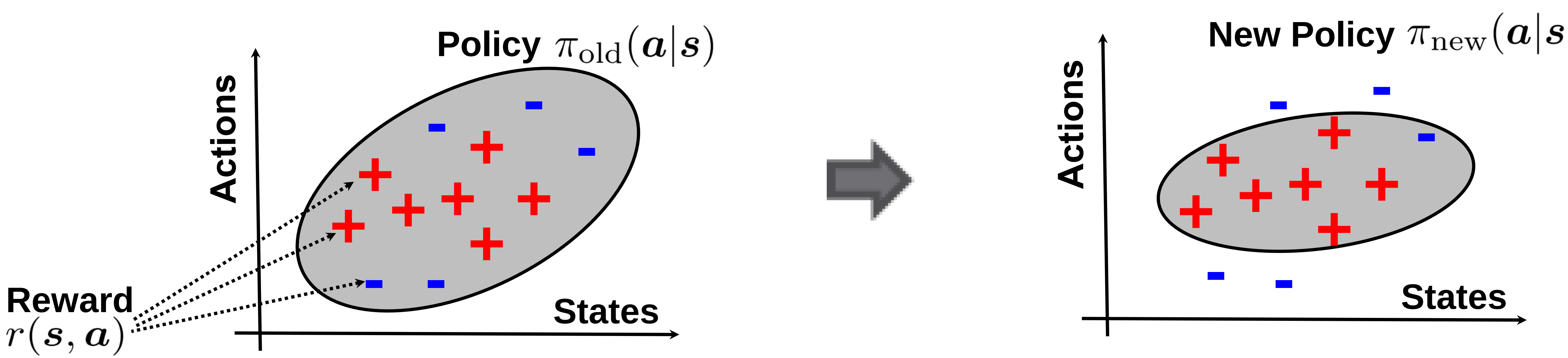
5. Learning Versatile Solutions

6. Sequencing Movement Primitives

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: Policy update learn to reproduce successful outcomes



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

Episode-Based Sucess Matching

Iterate:

Sample and evaluate parameters:

$$\boldsymbol{\theta}^{[i]} \sim \pi(\boldsymbol{\theta}; \boldsymbol{\omega}_k) \qquad R^{[i]} = \sum_{t=1}^{T} r_t^{[i]}$$

Compute „success probability“ for each sample

$$w^{[i]} = f(R^{[i]})$$

➡ transform reward in **a non-negative weight** (improper probability distribution)

Compute „Success“ weighted policy on the samples

$$p_k(\boldsymbol{\theta}^{[i]}) \propto w^{[i]} \pi(\boldsymbol{\theta}^{[i]}; \boldsymbol{\omega}_k)$$

Fit new parametric policy $\pi(\boldsymbol{\theta}^{[i]}; \boldsymbol{\omega}_{k+1})$ that best approximates $p_k(\boldsymbol{\theta}^{[i]})$

Episode-Based Sucess Matching

2 Open issues:

How to fit the policy $\pi(\boldsymbol{\theta}^{[i]}; \boldsymbol{\omega}_{k+1})$ **?**

How to compute $w^{[i]} = f(R^{[i]})$ **?**

Outline of the Lecture

1. Introduction

2. Policy Updates by Weighted Maximum Likelihood

3. Relative Entropy Policy Search (REPS)

4. REPS for Contextual Policy Search

5. Conclusion

Policy Fitting

Problem: We want to find a parametric distribution $\pi(\boldsymbol{\theta};\boldsymbol{\omega}_{k+1})$ that best fits the distribution $p(\boldsymbol{\theta}^{[i]}) \propto w^{[i]}\pi(\boldsymbol{\theta}^{[i]};\boldsymbol{\omega}_k),$

- **We can do that by minimizing:**

$$\begin{aligned}
\boldsymbol{\omega}_{k+1} &= \operatorname{argmin}_{\boldsymbol{\omega}} \quad \mathrm{KL}(p(\boldsymbol{\theta}^{[i]})||\pi(\boldsymbol{\theta}^{[i]};\boldsymbol{\omega})) \\
&= \operatorname{argmin}_{\boldsymbol{\omega}} \quad \int p(\boldsymbol{\theta}) \log \frac{p(\boldsymbol{\theta})}{\pi(\boldsymbol{\theta};\boldsymbol{\omega})} d\boldsymbol{\theta} \\
&\approx \operatorname{argmax}_{\boldsymbol{\omega}} \quad \sum_i \frac{p(\boldsymbol{\theta}^{[i]})}{\pi(\boldsymbol{\theta}^{[i]};\boldsymbol{\omega}_k)} \log \pi(\boldsymbol{\theta}^{[i]};\boldsymbol{\omega})
\end{aligned}$$

We sampled from the old policy

$$\boldsymbol{\omega}_{k+1} = \operatorname{argmax}_{\boldsymbol{\omega}} \sum_i w^{[i]} \log \pi(\boldsymbol{\theta}^{[i]};\boldsymbol{\omega})$$

- The fitting of the policy is obtained by a **weighted maximum likelihood estimate**
- Closed form solutions exists, no learning rates

Weighted Maximum Likelihood Solutions…

For a Gaussian policy: $\pi(\boldsymbol{\theta};\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu},\boldsymbol{\Sigma})$

$$\boldsymbol{\mu} = \frac{\sum_i w^{[i]}\boldsymbol{\theta}^{[i]}}{\sum_i w^{[i]}}$$

Weighted mean

$$\boldsymbol{\Sigma} = \frac{\sum_i w^{[i]}(\boldsymbol{\theta}^{[i]}-\boldsymbol{\mu})(\boldsymbol{\theta}^{[i]}-\boldsymbol{\mu})^T}{\sum_i w^{[i]}}$$

Weighted covariance

But more general: Also for mixture models, GPs and so on…

Difference to policy gradients

Weighted Maximum Likelihood:

$$\boldsymbol{\omega}_{k+1} = \operatorname{argmax}_{\boldsymbol{\omega}} \sum_i w^{[i]} \log \pi(\boldsymbol{\theta}^{[i]}; \boldsymbol{\omega})$$

I.e.: Set $\nabla_{\boldsymbol{\omega}} \sum_i w^{[i]} \log \pi(\boldsymbol{\theta}^{[i]}; \boldsymbol{\omega}) = \sum_i \nabla_{\boldsymbol{\omega}} \log \pi(\boldsymbol{\theta}^{[i]}; \boldsymbol{\omega}) w^{[i]} = 0$

Solve in closed form for $\boldsymbol{\omega}$

Policy Gradients:

$$\nabla_{\boldsymbol{\omega}} J_{\boldsymbol{\omega}} = \sum_{i=1}^{N} \nabla_{\boldsymbol{\omega}} \log \pi(\boldsymbol{\theta}_i; \boldsymbol{\omega}_k) R_i, \quad \boldsymbol{\omega}_{k+1} = \boldsymbol{\omega}_k + \alpha \nabla_{\boldsymbol{\omega}} J_{\boldsymbol{\omega}}$$

Computing the weights...

So **where are the weights** $w^{[i]} = f(R^{[i]})$ coming from?

We need to transform the returns in an **improper probability distribution**

Simple Way: Exponential transformation $w^{[i]} = \exp(\beta(R^{[i]} - \max R^{[i]}))$

$\beta \dots$ temperature of the distribution

Often set by heuristics, e.g.: $\beta = \frac{10}{\max R^{[i]} - \min R^{[i]}}$

Can be justified from different view-points

EM-Algorithms: PoWER, Reward-Weighted Regression

Optimal Control: PI2

Relative Entropy Policy Search

Exponential Transformation

Some notes on the exponential transformation

In stochastic environments, we do not optimize the expected reward any more as…

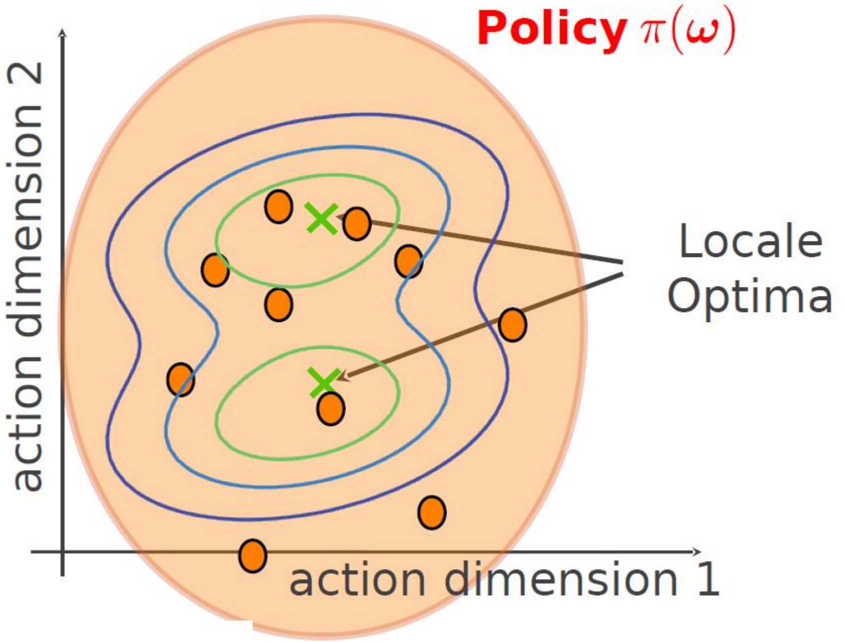
$$\mathbb{E}_{p(\boldsymbol{\tau})}\left[\exp(R(\tau))\right] \neq \exp(\mathbb{E}_{p(\boldsymbol{\tau})}\left[R(\tau)\right])$$

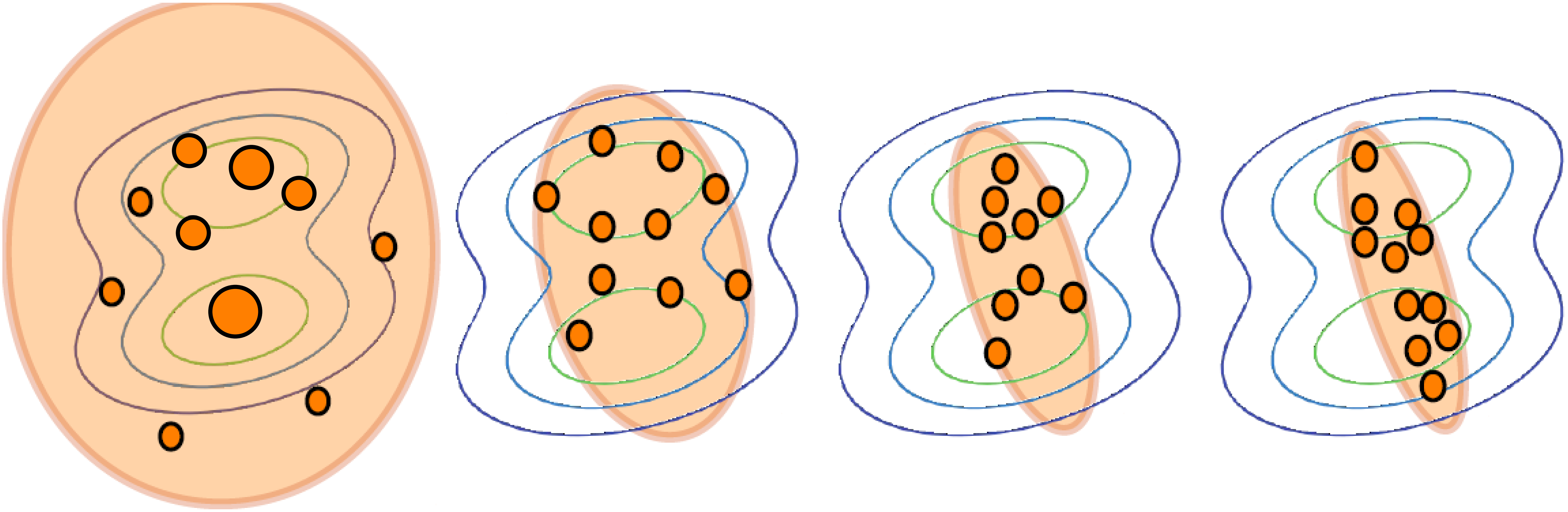
The objective gets „risk attracted“

For moderately stochastic environments it still works well

Illustration on weighted ML

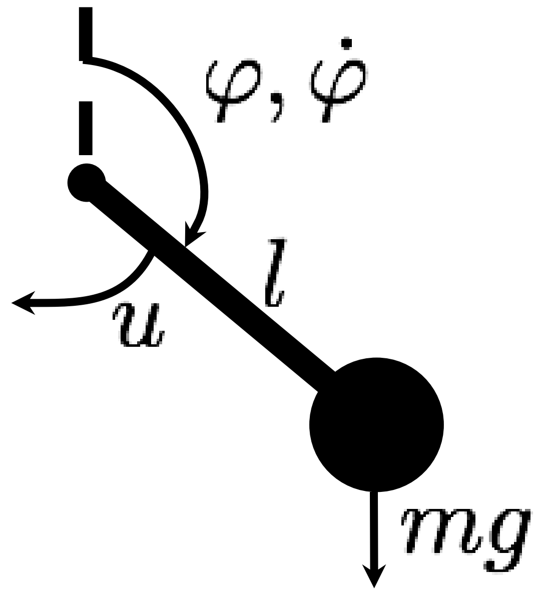
Example for a 2D parameter space:





Underactuated Swing-Up

- swing heavy pendulum up



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

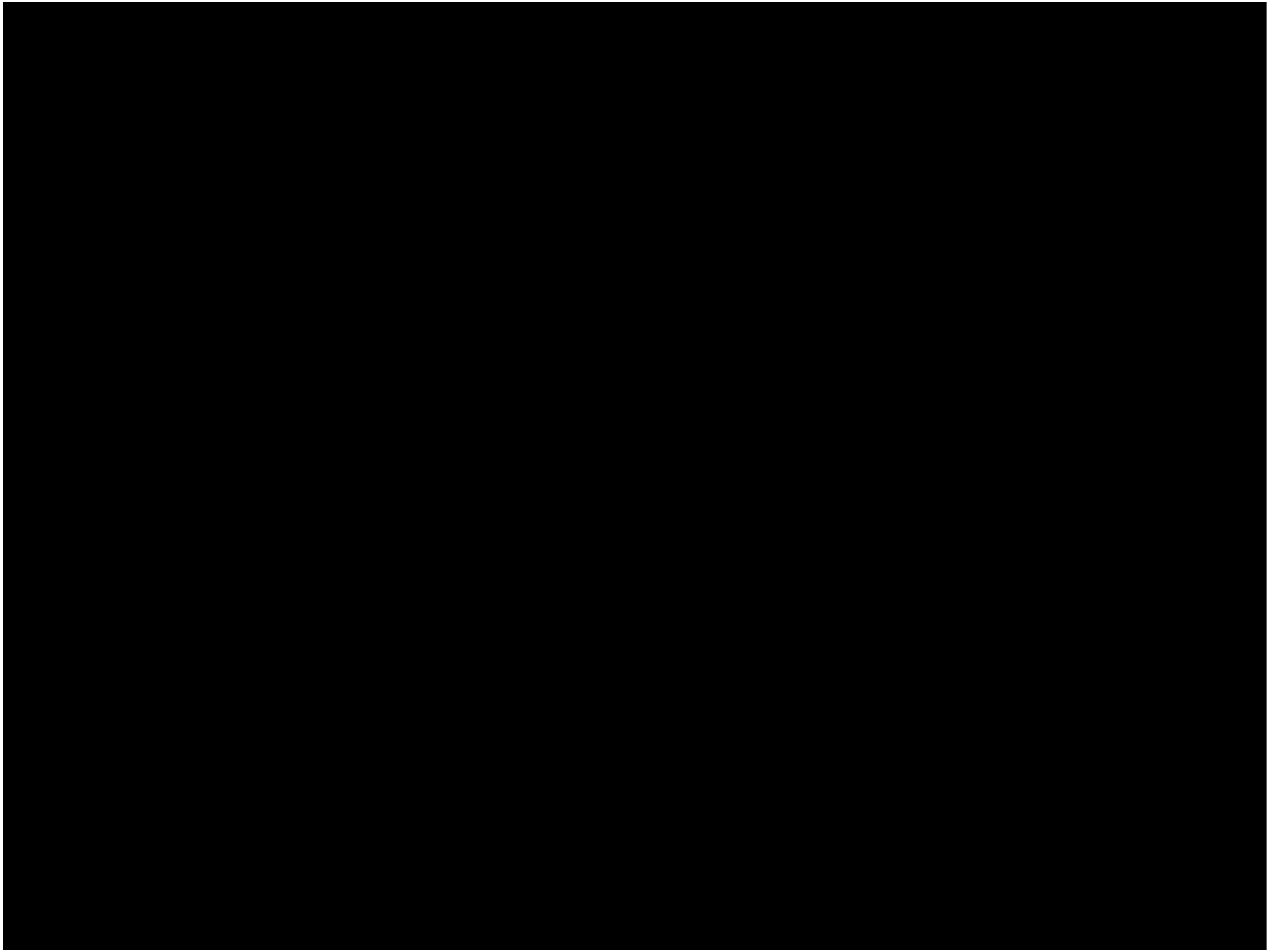
- motor torques limited, Policy: DMPs

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

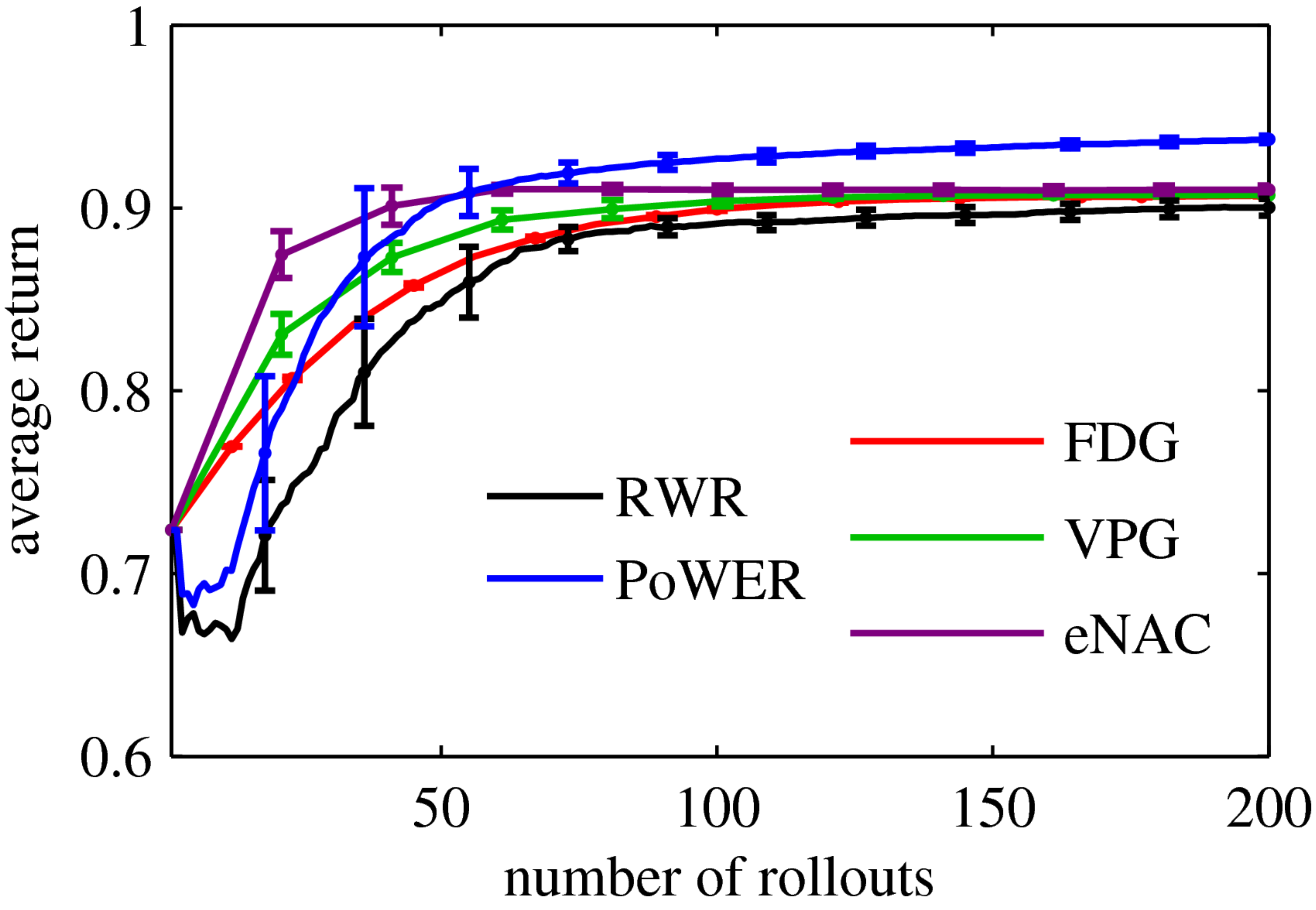
- reward function

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

(Schaal, NIPS 1997; Atkeson, ICML 1997)

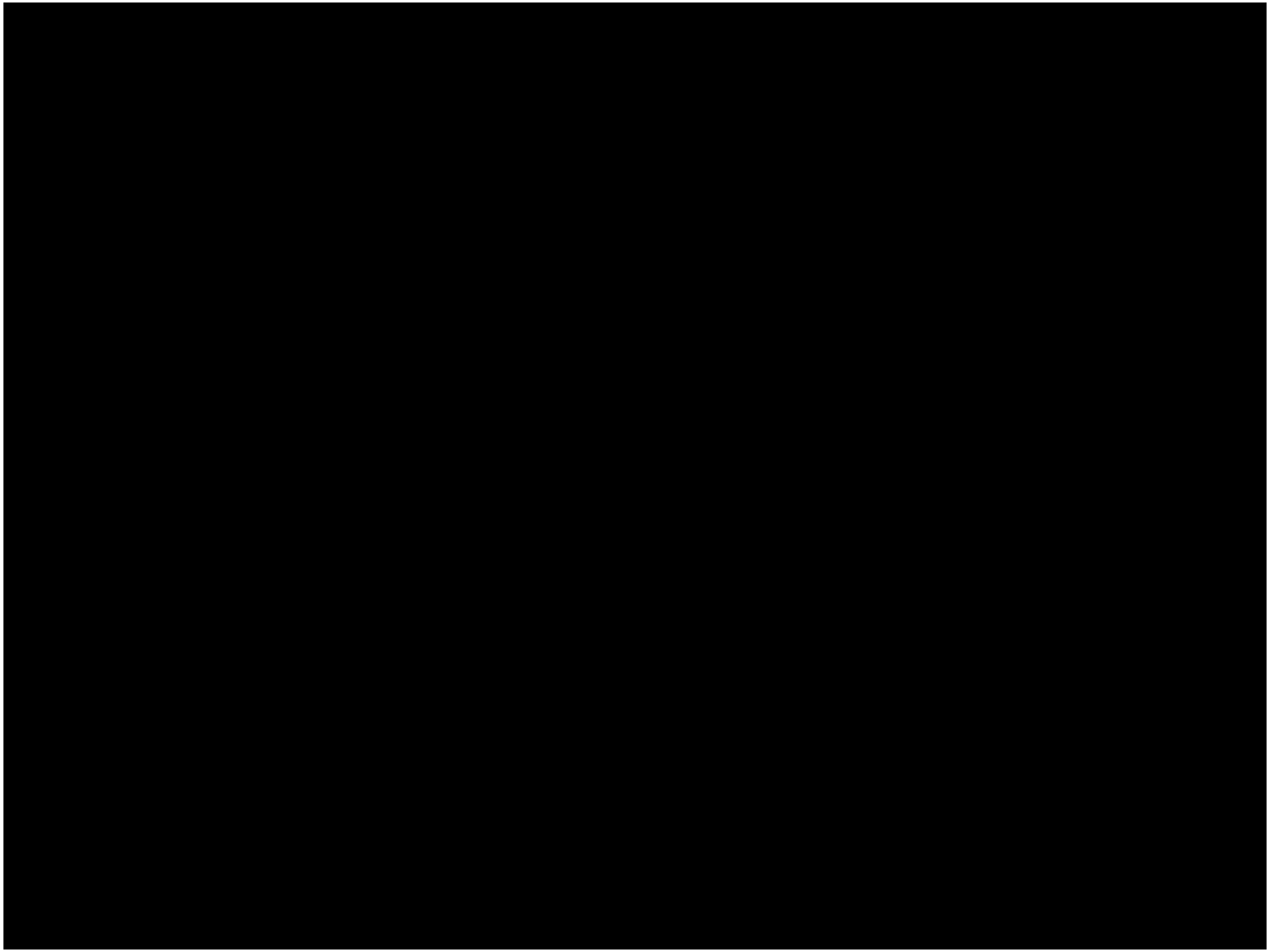


Underactuated Swing-Up



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

Ball in the Cup

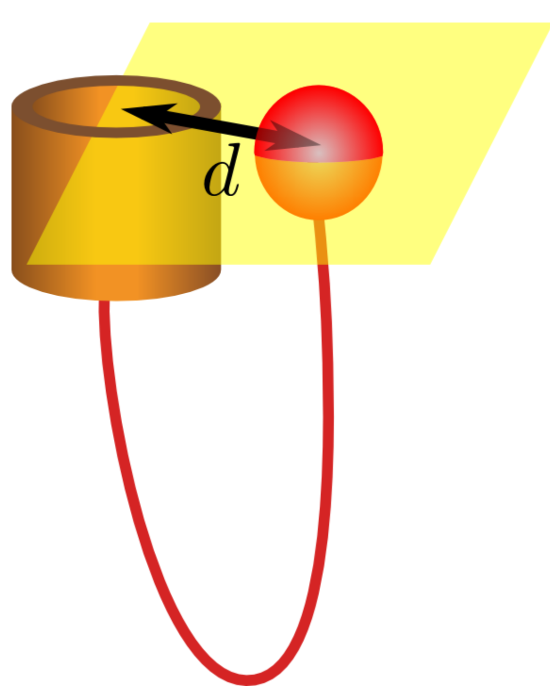


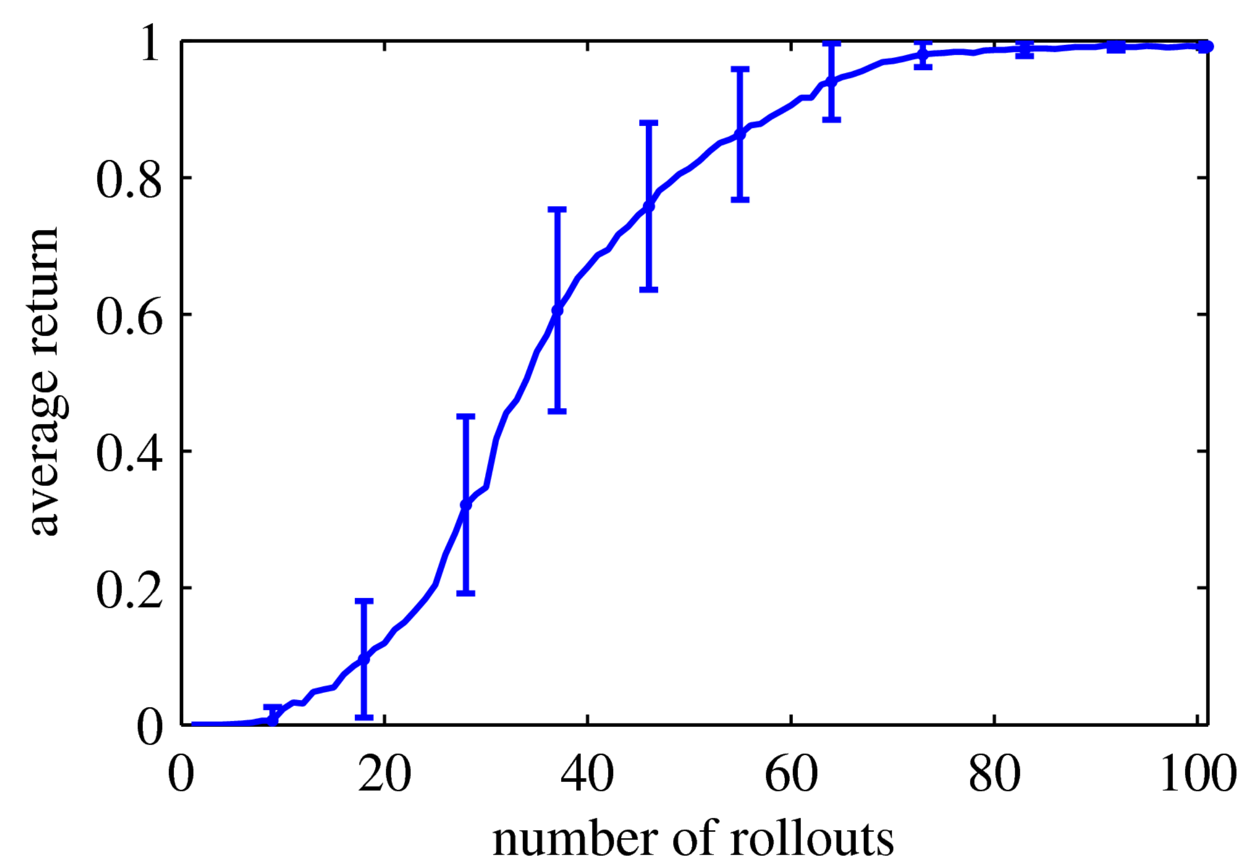
Ball-in-a-Cup

Reward function:

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

Policy: DMPs





Outline of the Lecture

1. Introduction

2. Policy Updates by Weighted Maximum Likelihood

3. Relative Entropy Policy Search (REPS)

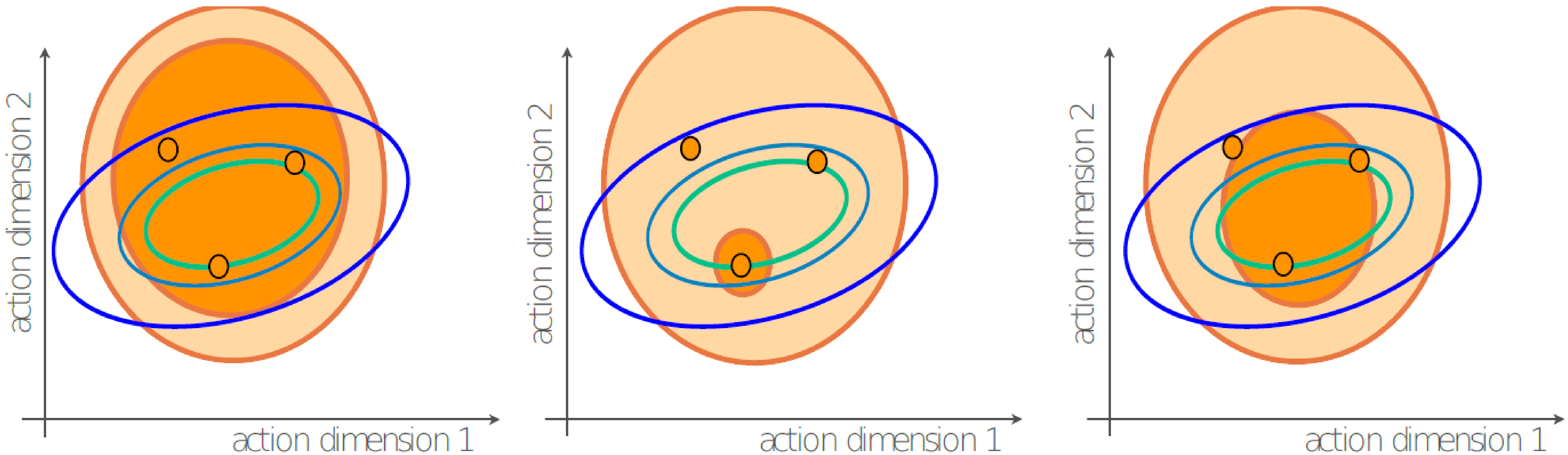
4. REPS for Contextual Policy Search

5. Learning Versatile Solutions

6. Sequencing Movement Primitives

Policy Search: Choosing the step size

What is a good desired distribution for the policy update?



Small Beta High Beta Moderate Beta

How can we choose the **exploration-exploitation tradeoff?**

Again use a metric to control the step-size of the update

Relative Entropy Policy Search

Relative entropy as metric between two policies

$$\mathrm{KL}(\pi(\boldsymbol{\theta})||q(\boldsymbol{\theta})) \leq \epsilon$$

We get the following optimization problem:

$$\max_\pi \sum_i \pi(\boldsymbol{\theta}^{[i]}) R(\boldsymbol{\theta}^{[i]})$$ Maximize Reward

$$\text{s.t:} \quad \sum_i \pi(\boldsymbol{\theta}^{[i]}) = 1$$ It‘s a distribution

$$\mathrm{KL}(\pi(\boldsymbol{\theta})||q(\boldsymbol{\theta})) \leq \epsilon$$ Stay close to the old policy $q(\boldsymbol{\theta})$

Policy Update is formulated as constrained optimization problem

Relative Entropy Policy Search

We get the following optimization problem:

$$\max_\pi \sum_i \pi(\boldsymbol{\theta}^{[i]}) R(\boldsymbol{\theta}^{[i]})$$ Maximize Reward

s.t: $$\sum_i \pi(\boldsymbol{\theta}^{[i]}) = 1$$ It's a distribution

$$\mathrm{KL}(\pi(\boldsymbol{\theta}) || q(\boldsymbol{\theta})) \leq \epsilon$$ Stay close to the old policy $q(\boldsymbol{\theta})$

Which has the following analytic solution:

$$\pi(\boldsymbol{\theta}) \propto q(\boldsymbol{\theta}) \exp\left(\frac{\mathcal{R}_{\boldsymbol{\theta}}}{\eta}\right)$$

Thats exactly sucess matching with exponential transformation!

Scalingfactor η :

- Automatically chosen from optimization (Lagrange Multiplier)
- Specified by KL-bound ϵ

Getting the Lagrangian multipliers

How to get η :

Solve **dual optimization** problem:

Dual function: $$h(\eta) = \eta\epsilon + \eta \log \int q(\boldsymbol{\theta}) \exp\left(\frac{\mathcal{R}_{\boldsymbol{\theta}}}{\eta}\right) d\boldsymbol{\omega}$$

$$= \eta\epsilon + \eta \log \sum_{i=1}^{N} \frac{1}{N} \exp\left(\frac{\mathcal{R}^{[i]}}{\eta}\right)$$

Minimize: $\eta^* = \mathrm{argmin}_\eta h(\eta) \quad \text{s.t: } \eta > 0$

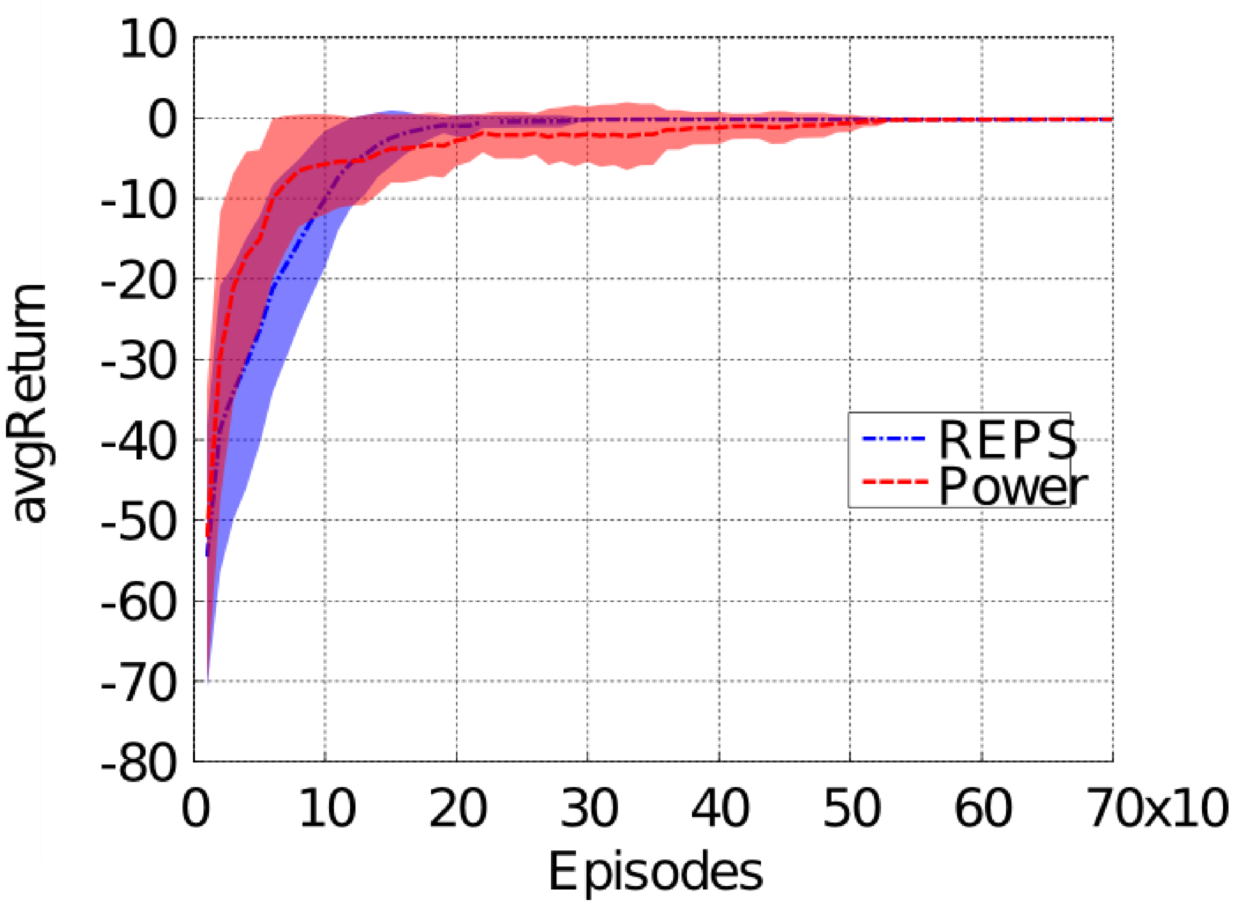
Log-sum-exp softmax structure

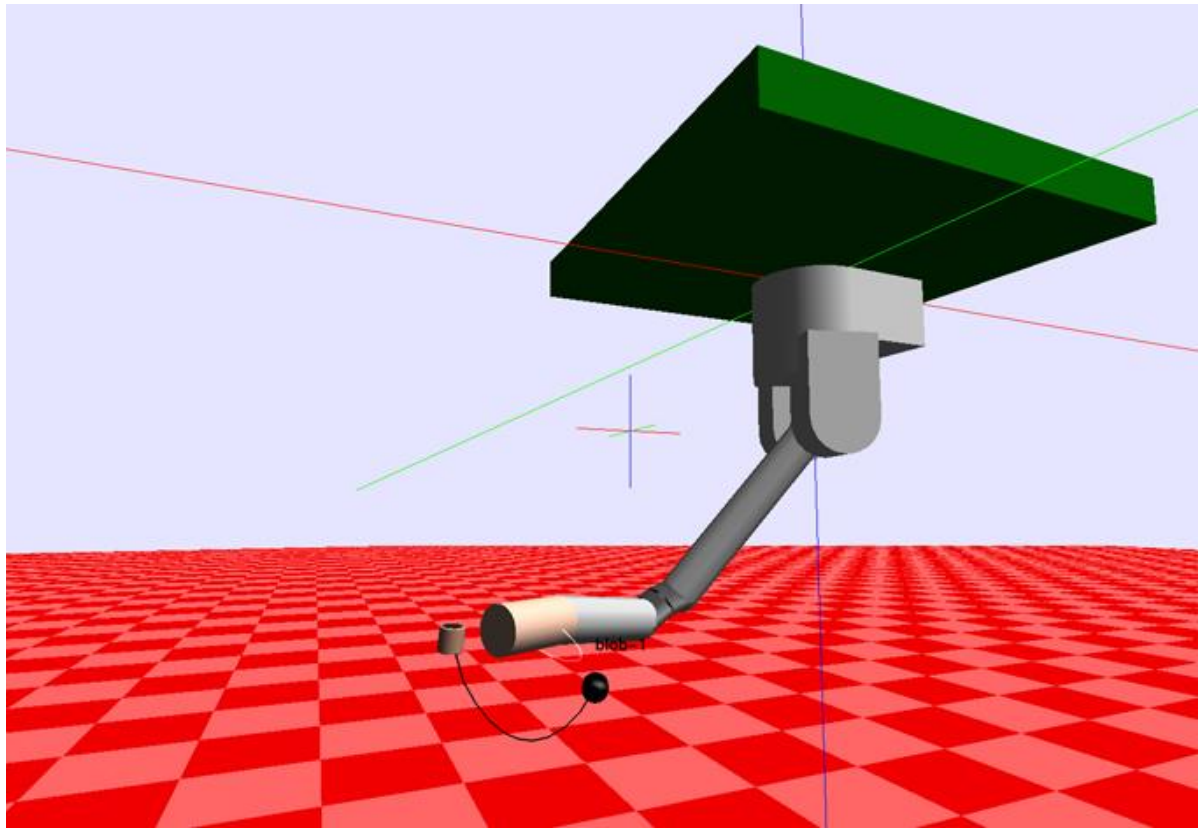
Optimized by standard optimization tools

(e.g. trust region algorithms)

Results

Comparison on simulated Ball In The Cup





Outline of the Lecture

1. Introduction

2. Policy Updates by Weighted Maximum Likelihood

3. Relative Entropy Policy Search (REPS)

4. **REPS for Contextual Policy Search**

5. Learning Versatile Solutions

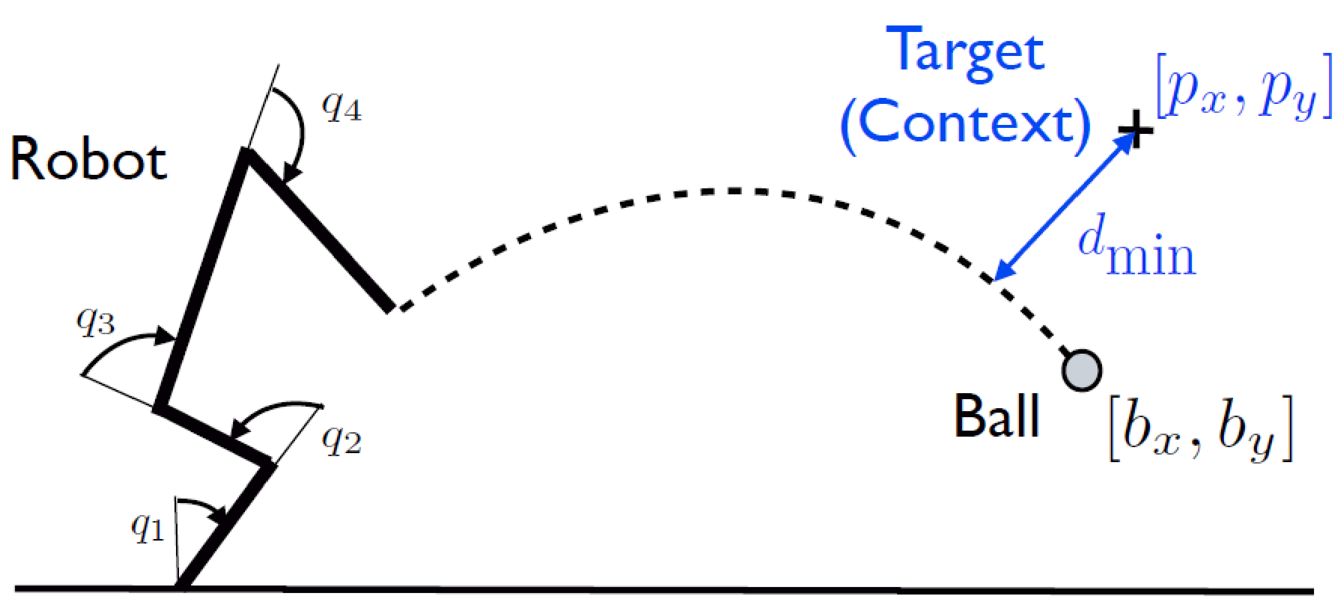
6. Sequencing Movement Primitives

Contextual Policy Search

Contextual Policy Search

Context $\boldsymbol{x}$ describes objectives of the task (fixed before task execution)

E.g.: Target location to throw a ball



Contextual Policy Search

Contextual Policy Search

Context $\boldsymbol{x}$ describes objectives of the task (fixed before task execution)

E.g.: Target location to throw a ball

We now want to learn an upper level policy $\pi(\boldsymbol{\theta}|\boldsymbol{x};\boldsymbol{\omega})$ that adapts $\boldsymbol{\theta}$ to the context

Data-set used for policy update

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

Goal: maximize expected reward

$$J_\pi = \iint \mu_0(\boldsymbol{x})\pi(\boldsymbol{\theta}|\boldsymbol{x})\mathcal{R}_{\boldsymbol{x}\boldsymbol{\theta}} d\boldsymbol{x} d\boldsymbol{\theta}$$

Contextual Policy Search

Optimize over the joint distribution: $p(\boldsymbol{x}, \boldsymbol{\theta}) = \mu(\boldsymbol{x})\pi(\boldsymbol{\theta}|\boldsymbol{x})$

$$\max_p \sum_{\boldsymbol{x},\boldsymbol{\theta}} p(\boldsymbol{x}, \boldsymbol{\theta}) R(\boldsymbol{x}, \boldsymbol{\theta})$$ Maximize Reward

$$\sum_{\boldsymbol{x},\boldsymbol{\theta}} p(\boldsymbol{x}, \boldsymbol{\theta}) = 1$$ It‘s a distribution

$$\mathrm{KL}(p(\boldsymbol{x}, \boldsymbol{\theta}) || q(\boldsymbol{x}, \boldsymbol{\theta})) \leq \epsilon$$ Stay close to the data

$$\forall \boldsymbol{x} \quad p(\boldsymbol{x}) = \sum_\theta p(\boldsymbol{x}, \boldsymbol{\theta}) = \mu_0(\boldsymbol{x})$$ Reproduce given context distribution $\boldsymbol{\mu}_0(\boldsymbol{x})$

Problems:

- Context distribution can not be freely chosen by the algorithm
- Infinite amount of contraints
- For each context, we need many parameter vector samples

Matching Feature Averages

$$\max_p \sum_{\boldsymbol{x},\boldsymbol{\theta}} p(\boldsymbol{x},\boldsymbol{\theta}) R(\boldsymbol{x},\boldsymbol{\theta})$$

Maximize Reward

$$\sum_{\boldsymbol{x},\boldsymbol{\theta}} p(\boldsymbol{x},\boldsymbol{\theta}) = 1$$

It‘s a distribution

$$\mathrm{KL}(\pi(\boldsymbol{\theta}|\boldsymbol{x})\mu(\boldsymbol{x})||q(\boldsymbol{x},\boldsymbol{\theta})) \leq \epsilon$$

Stay close to the data

$$\sum_{\boldsymbol{x},\boldsymbol{\theta}} p(\boldsymbol{x},\boldsymbol{\theta}) \boldsymbol{\phi}(\boldsymbol{x}) = \hat{\boldsymbol{\phi}}$$

Reproduce given context feature averages

Instead of matching the context distribution exactly,
we can match only certain feature averages (moments) of the distribution

Matching Feature Averages

$$\sum_{\boldsymbol{x},\boldsymbol{\theta}} p(\boldsymbol{x},\boldsymbol{\theta})\boldsymbol{\phi}(\boldsymbol{x}) = \hat{\boldsymbol{\phi}}$$

Reproduce given context feature averages

What does that mean? Example:

$$\phi(x) = \left[\begin{array}{c} x \\ x^2 \end{array} \right]$$

- Match first and second order moment
- Equivalent to matching mean and variance
- Exact for Gaussian distributions

Matching Feature Averages

$$\max_p \sum_{\boldsymbol{x},\boldsymbol{\theta}} p(\boldsymbol{x},\boldsymbol{\theta}) R(\boldsymbol{x},\boldsymbol{\theta})$$

Maximize Reward

$$\sum_{\boldsymbol{x},\boldsymbol{\theta}} p(\boldsymbol{x},\boldsymbol{\theta}) = 1$$

It‘s a distribution

$$\mathrm{KL}(\pi(\boldsymbol{\theta}|\boldsymbol{x})\mu(\boldsymbol{x})||q(\boldsymbol{x},\boldsymbol{\theta})) \leq \epsilon$$

Stay close to the data

$$\textstyle\sum_{\boldsymbol{x},\boldsymbol{\theta}} p(\boldsymbol{x},\boldsymbol{\theta})\boldsymbol{\phi}(\boldsymbol{x}) = \hat{\boldsymbol{\phi}}$$

Reproduce given context feature averages

Closed form solution:

$$\mu(\boldsymbol{x})\pi(\boldsymbol{\theta}|\boldsymbol{x}) \propto q(\boldsymbol{x},\boldsymbol{\theta}) \exp\left(\frac{R_{\boldsymbol{x}\boldsymbol{\theta}} - V(\boldsymbol{x})}{\eta}\right)$$

We automatically get a baseline for the returns

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

Again given by Lagrangian multipliers $\boldsymbol{v}$

Matching Feature Averages

$$\max_p \sum_i p(\boldsymbol{x}^{[i]}, \boldsymbol{\theta}^{[i]}) R(\boldsymbol{x}^{[i]}, \boldsymbol{\theta}^{[i]})$$ Maximize Reward

$$\sum_i p(\boldsymbol{x}^{[i]}, \boldsymbol{\theta}^{[i]}) = 1$$ It‘s a distribution

$$\mathrm{KL}(\pi(\boldsymbol{\theta}|\boldsymbol{x})\mu(\boldsymbol{x}) || q(\boldsymbol{x}, \boldsymbol{\theta})) \leq \epsilon$$ Stay close to the data

Reproduce given context feature averages

e.g., $\phi(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}$ Match Mean and Variance

$$\boxed{\mu(\boldsymbol{x})\pi(\boldsymbol{\theta}|\boldsymbol{x}) \propto q(\boldsymbol{x}, \boldsymbol{\theta}) \exp\left(\frac{R_{\boldsymbol{x}\boldsymbol{\theta}} - V(\boldsymbol{x})}{\eta}\right)}$$

Matching Feature Averages

$$\max_p \sum_i p(\boldsymbol{x}^{[i]}, \boldsymbol{\theta}^{[i]}) R(\boldsymbol{x}^{[i]}, \boldsymbol{\theta}^{[i]})$$ Maximize Reward

$$\textstyle\sum_i p(\boldsymbol{x}^{[i]}, \boldsymbol{\theta}^{[i]}) = 1$$ It‘s a distribution

$$\mathrm{KL}(\pi(\boldsymbol{\theta}|\boldsymbol{x})\mu(\boldsymbol{x})||q(\boldsymbol{x}, \boldsymbol{\theta})) \leq \epsilon$$ Stay close to the data

$$\textstyle\sum_{\boldsymbol{x}} p(\boldsymbol{x})\phi(\boldsymbol{x}) = \hat{\phi}$$ Reproduce given context feature averages

e.g., $\boldsymbol{\phi}(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}$ Match Mean and Variance

$$\boxed{\mu(\boldsymbol{x})\pi(\boldsymbol{\theta}|\boldsymbol{x}) \propto q(\boldsymbol{x}, \boldsymbol{\theta}) \exp\left(\frac{R_{\boldsymbol{x\theta}} - V(\boldsymbol{x})}{\eta}\right)}$$

Getting the Lagrangian multipliers

How to get $\eta, \boldsymbol{v}$

Solve **dual optimization** problem:

Dual function:

$$h(\eta, \boldsymbol{v}) = \eta\epsilon + \hat{\boldsymbol{\phi}}^T\boldsymbol{v} + \eta \log \sum_i \frac{1}{N} \exp\left(\frac{\mathcal{R}^{[i]} - \boldsymbol{\phi}^T(\boldsymbol{x}^{[i]})\boldsymbol{v}}{\eta}\right)$$

Minimize: $[\eta^*, \boldsymbol{v}^*] = \mathrm{argmin}_\eta h(\eta, \boldsymbol{v}) \quad \text{s.t: } \eta > 0$

Integral is over the context-parameter space

We can use $(\boldsymbol{x}^{[i]}, \boldsymbol{\theta}^{[i]})$ samples instead of many samples $\boldsymbol{\theta}^{[ij]}$ per context $\boldsymbol{x}^{[i]}$

Contextual Policies with weighted ML

Estimate parametric policy $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{x})$:

If $\pi_{\boldsymbol{\omega}}(\boldsymbol{\theta}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{K}\boldsymbol{x} + \boldsymbol{k}, \boldsymbol{\Sigma}_{\boldsymbol{\omega}})$ is Gaussian:

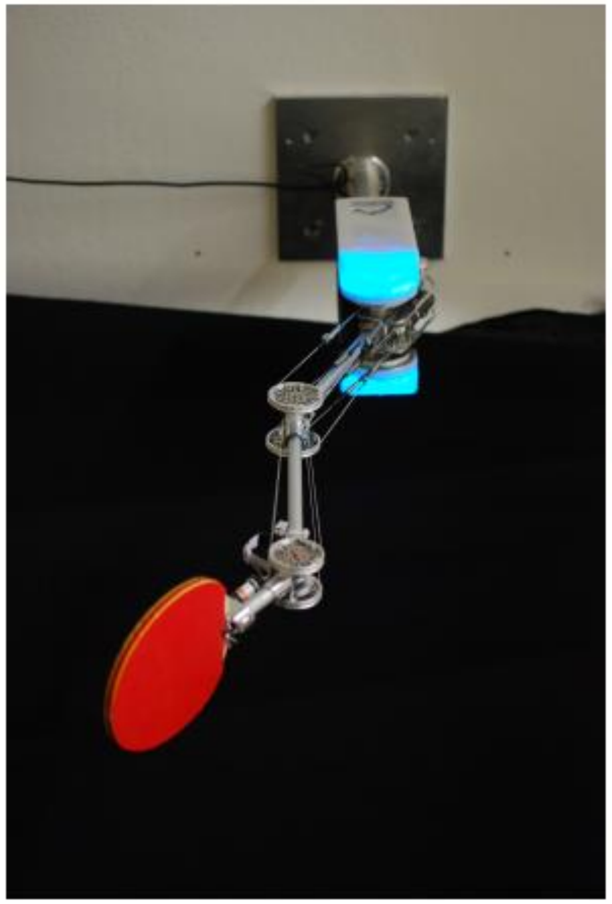
$$\begin{bmatrix} \boldsymbol{k}^T \\ \boldsymbol{K}^T \end{bmatrix} = (\boldsymbol{X}^T\boldsymbol{D}\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{D}\boldsymbol{A}$$

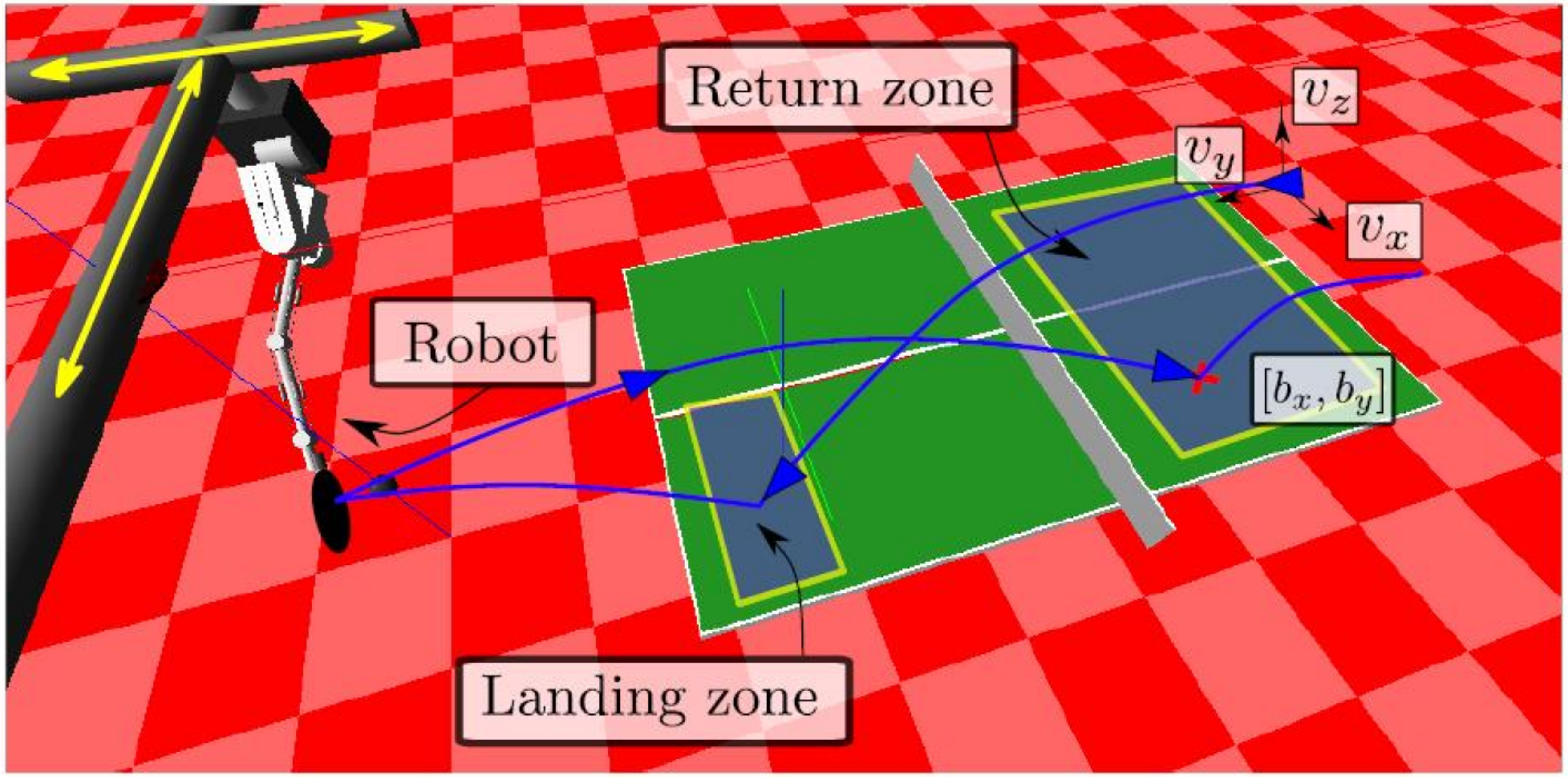
$$\boldsymbol{\mu}_i = \boldsymbol{k} + \boldsymbol{K}\boldsymbol{x}_i \qquad \boldsymbol{\Sigma} = \frac{\sum_i p_i(\boldsymbol{\theta}^{[i]} - \boldsymbol{\mu}_i)(\boldsymbol{\theta}^{[i]} - \boldsymbol{\mu}_i)^T}{\sum_i p_i}$$

- X … input data matrix (including 1 for the bias)
- D … diagonal weighting matrix
- A …. Parameter matrix

Just standard **weighted linear regression**…

Table tennis experiments



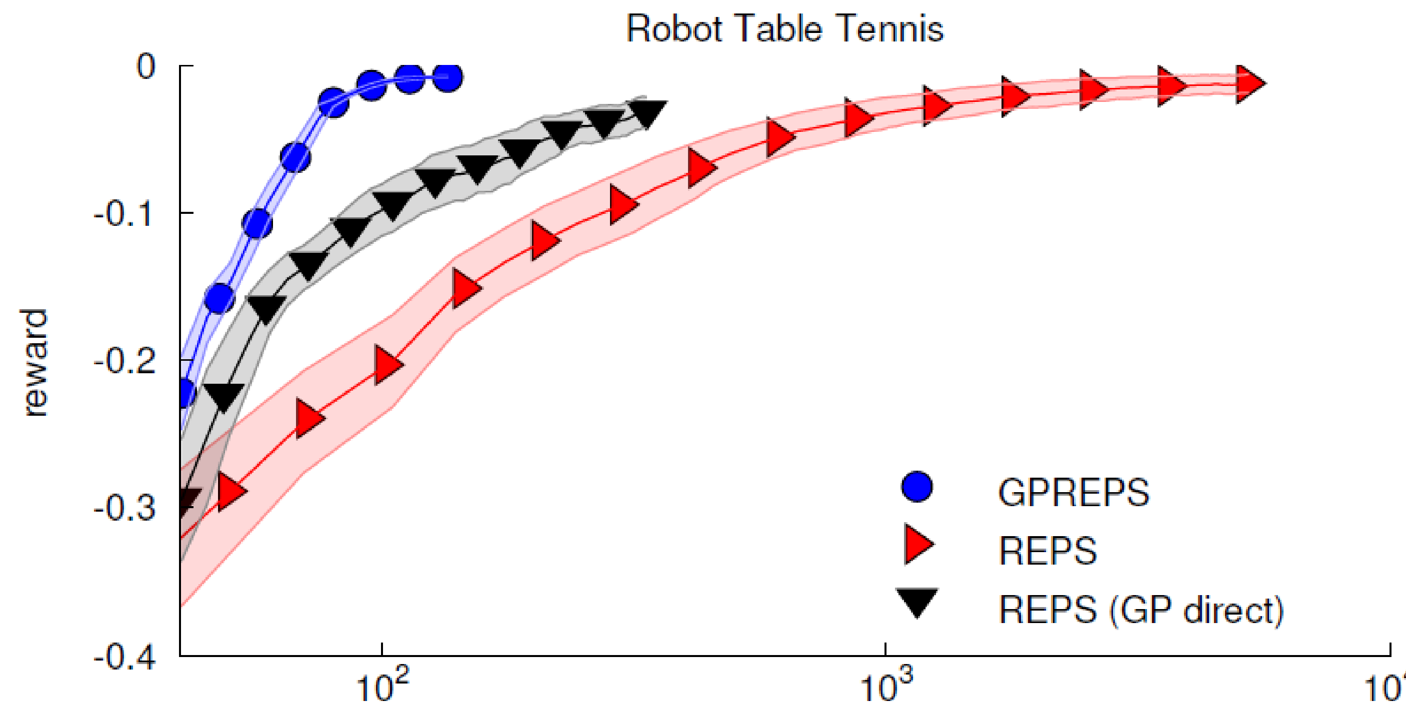


[Kupscik, Neumann et al, submitted, 2013]

Table tennis experiments

REPS with learned forward models

- Complex behavior can be learned within 100 episodes



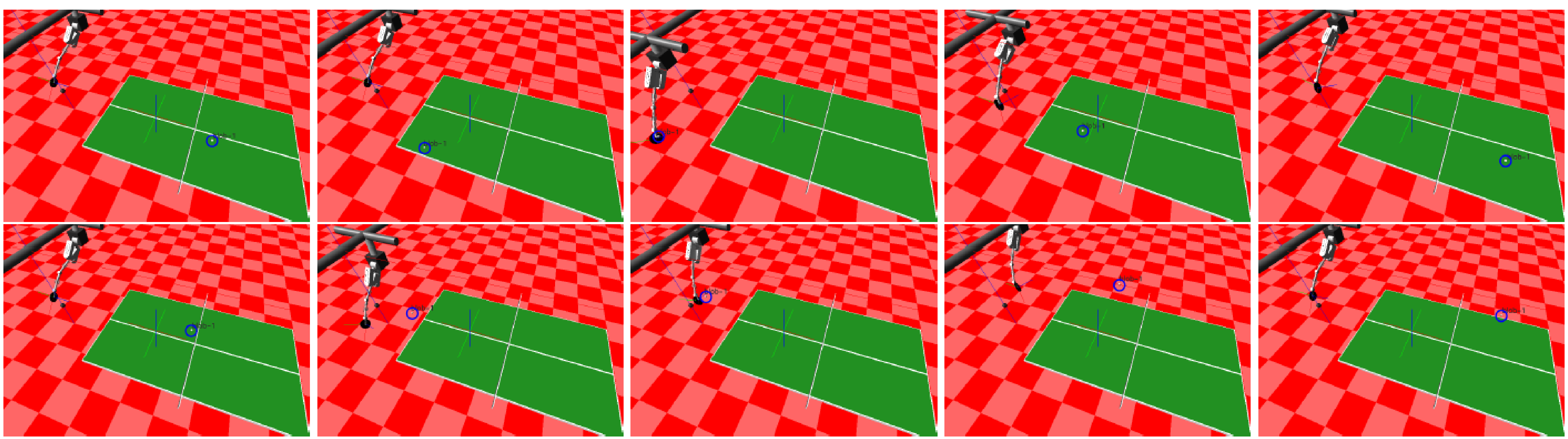
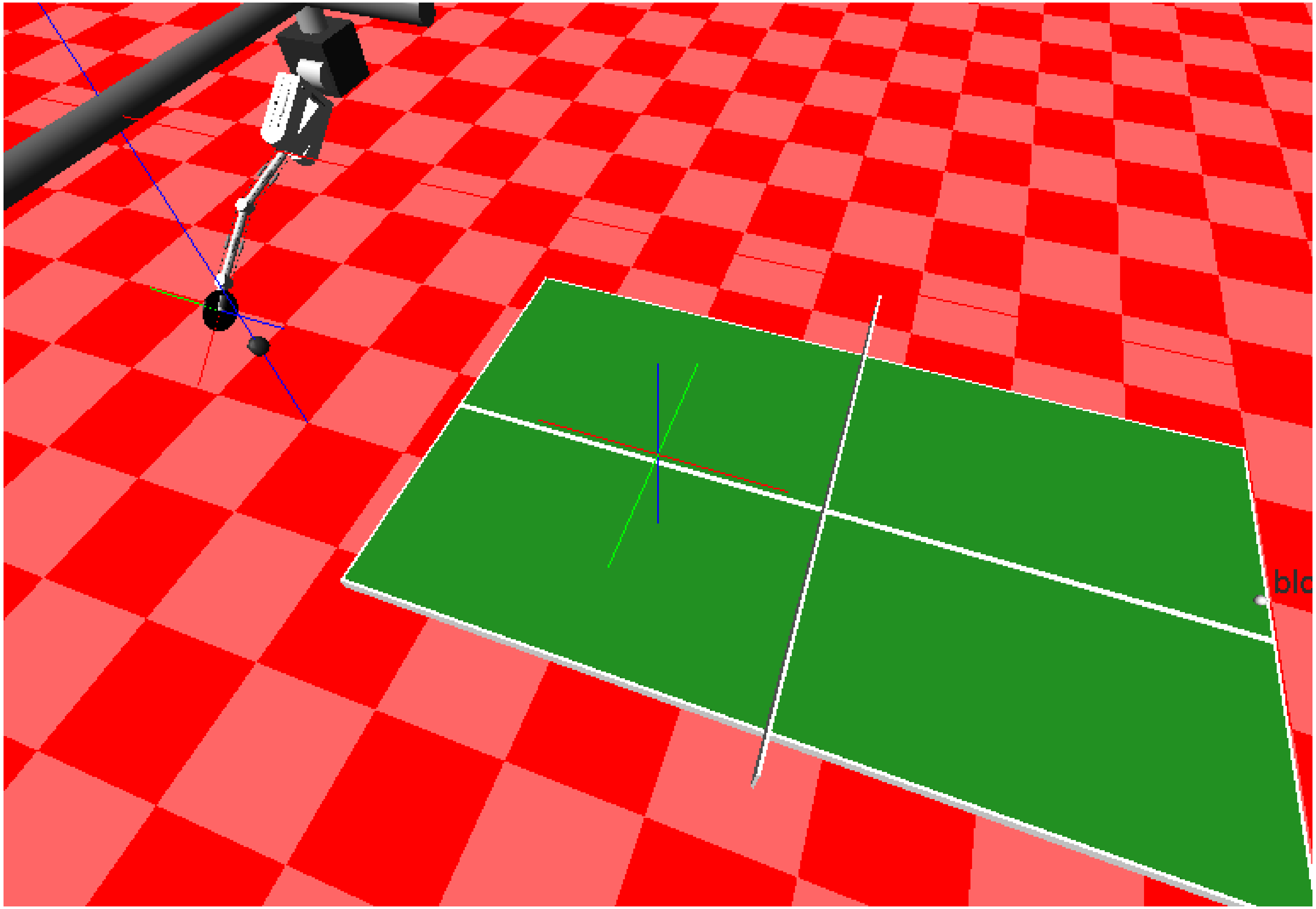


Table tennis experiments



Outline of the Lecture

1. Introduction

2. Policy Updates by Weighted Maximum Likelihood

3. Relative Entropy Policy Search (REPS)

4. REPS for Contextual Policy Search

5. **Learning Versatile Solutions**

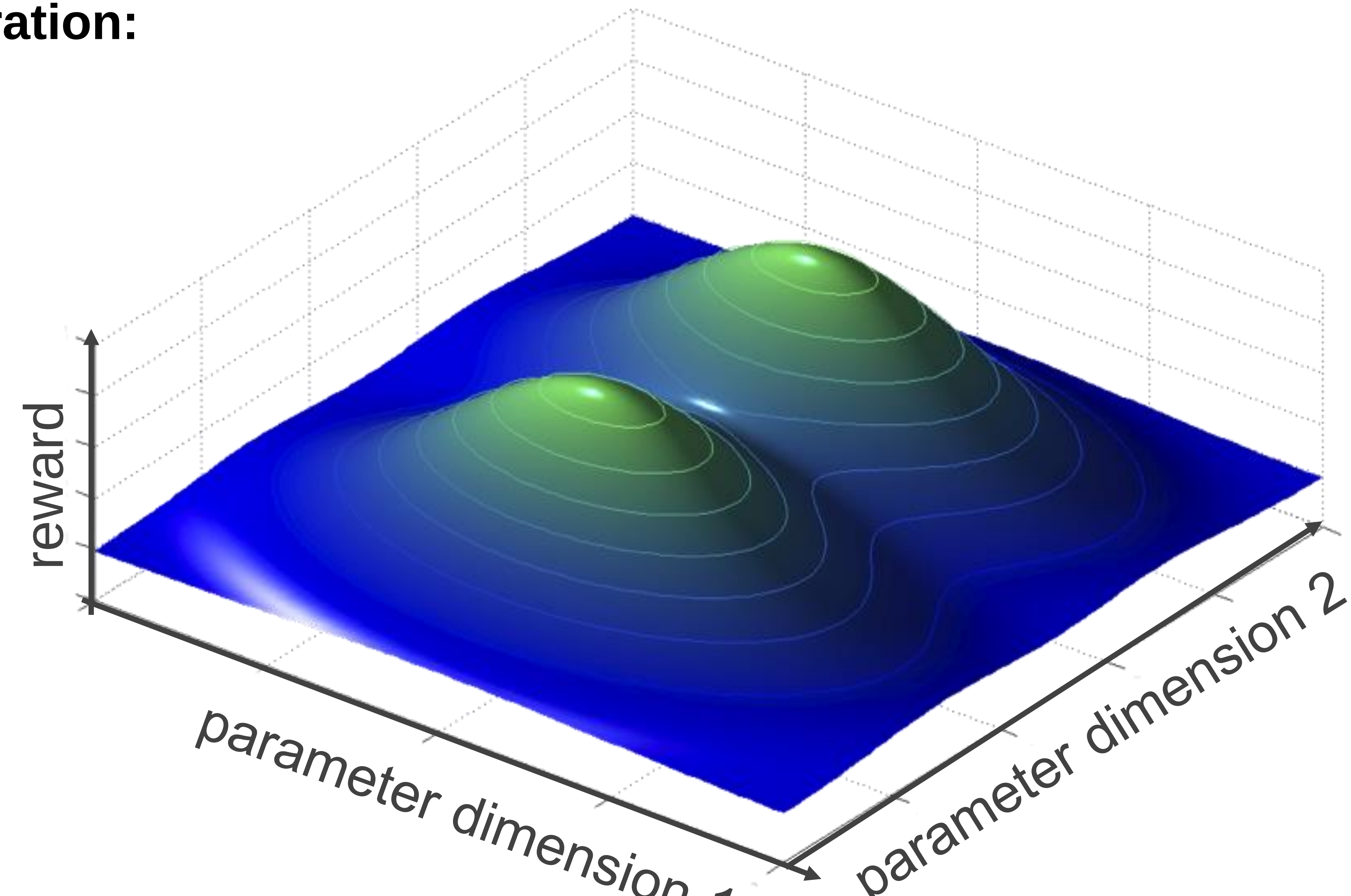
6. Sequencing Movement Primitives

Versatile Solutions: Illustration

Many motor-tasks have multiple solutions:

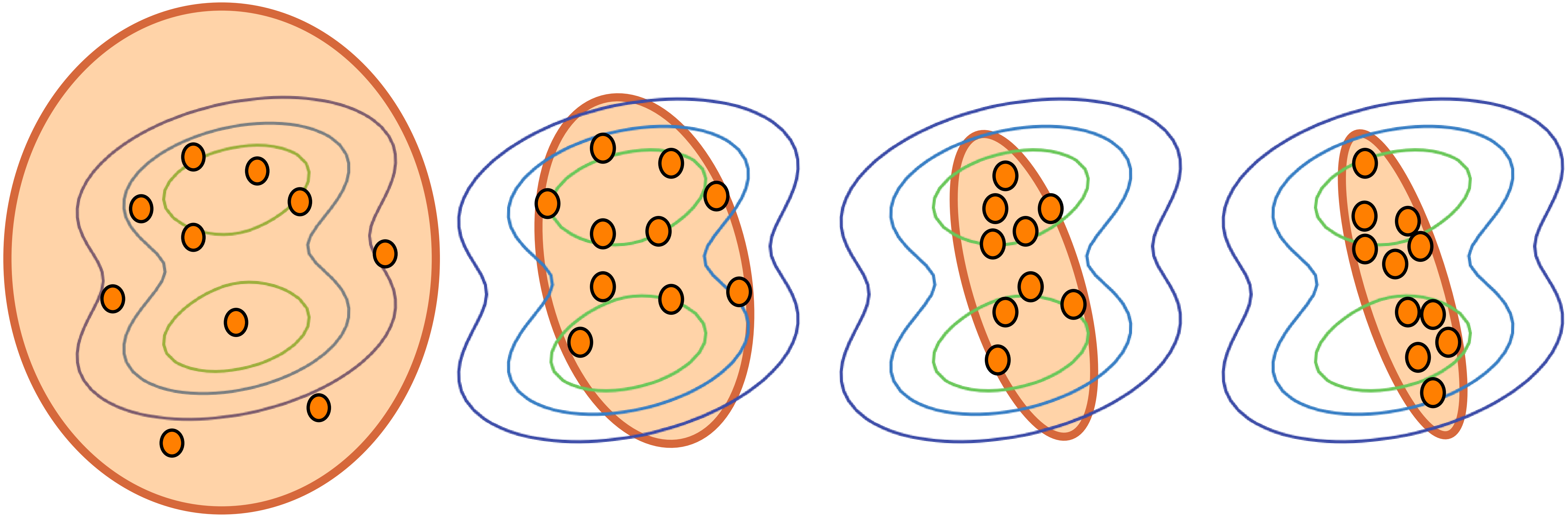
- ➡ More difficult policy search problem
- ➡ We want to find all these solutions

Illustration:



Illustration

Current Formulation:

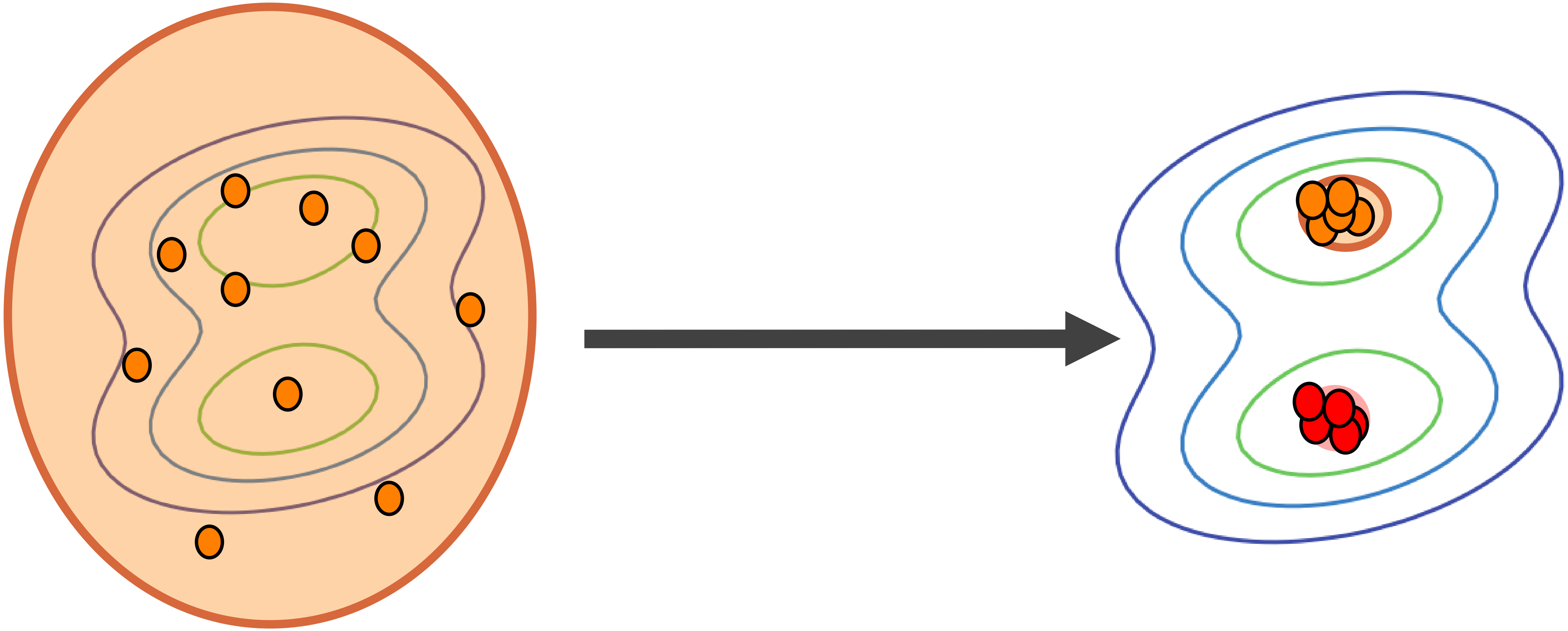


Iteration 0 Iteration 3 Iteration 6 Iteration 9

➡ Policy averages over several modes.

Illustration

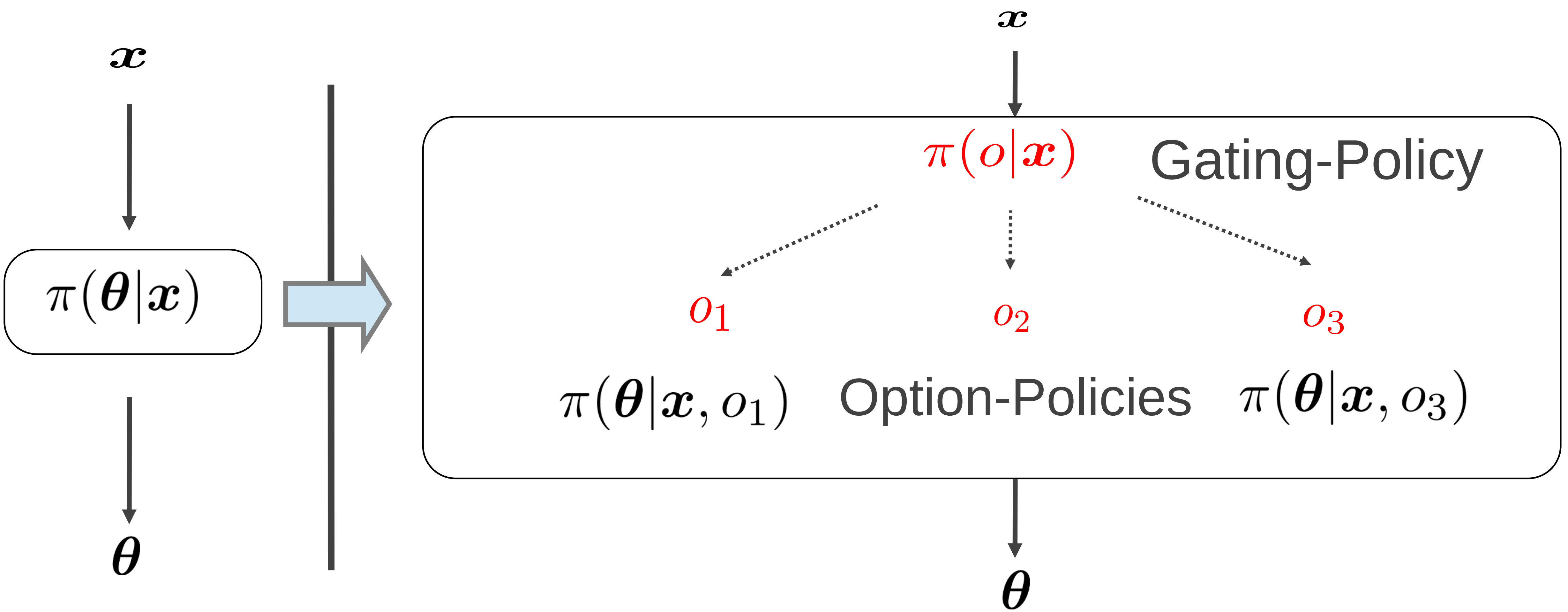
We want to find both solutions!



Introduce Hierarchy

Upper-level policy as combination of options

- Selection of the option: Gating-policy
- Selection of the parameters: Option-policy



“Naive” Hierarchical Approach

$$\max_{\pi,\mu} \sum_{\boldsymbol{x},\omega,o} \mu(\boldsymbol{x})\pi(\boldsymbol{\omega}|\boldsymbol{x},o)\pi(o|\boldsymbol{x})R_{\boldsymbol{x}\boldsymbol{\omega}}$$ Maximize reward

$$\sum_{\boldsymbol{x},\boldsymbol{\omega},o} \mu(\boldsymbol{x})\pi(\boldsymbol{\omega}|\boldsymbol{x},o)\pi(o|\boldsymbol{x}) = 1$$ Distribution

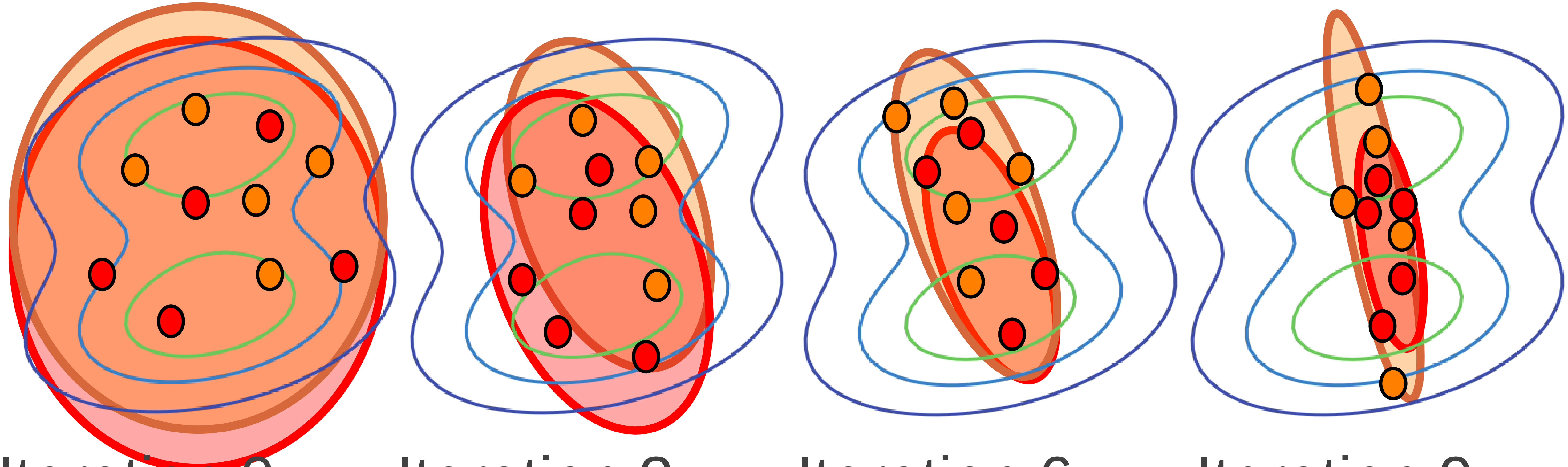
$$\sum_{\boldsymbol{x}} \mu(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x}) = \hat{\boldsymbol{\phi}}$$ Reproduce Context-Features

$$\epsilon \geq \sum_{\boldsymbol{x},\boldsymbol{\omega},o} \mu(\boldsymbol{x})\pi(\boldsymbol{\omega},o|\boldsymbol{x}) \log \frac{\mu(\boldsymbol{x})\pi(\boldsymbol{\omega},o|\boldsymbol{x})}{q(\boldsymbol{x},\boldsymbol{\omega},o)}$$ Stay close to the “data”

? Versatile Solutions

Illustration

“Naive” Approach:



Iteration 0 Iteration 3 Iteration 6 Iteration 9

Multiple Options, BUT no separation

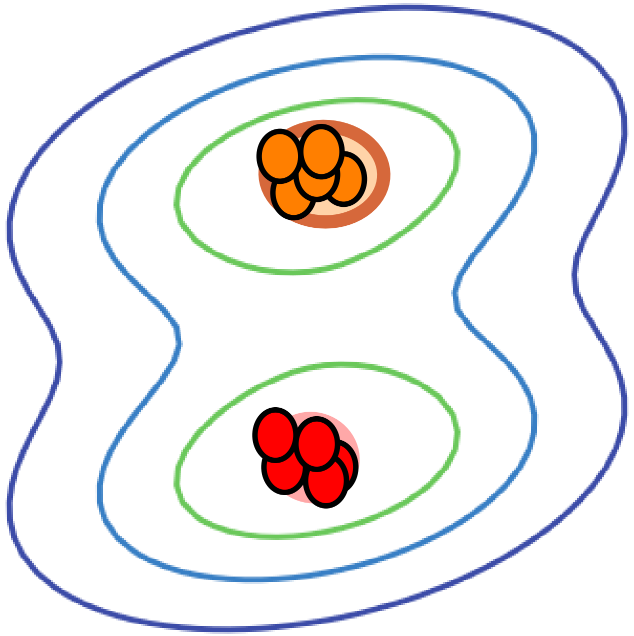
Learning versatile Options

Options should represent distinct solutions.

➡ Limit the overlap of the options

High entropy of $p(o|\boldsymbol{x}, \boldsymbol{\theta})$ ➡ high overlap

Limit the entropy ➡ less overlap



$$\kappa \geq \mathbb{E}\left[\underbrace{-\sum_{o} p(o|\boldsymbol{x}, \boldsymbol{\theta}) \log p(o|\boldsymbol{x}, \boldsymbol{\theta})}_{\text{Entropy}}\right]$$

Hierarchical REPS (HiREPS)

$$\max_{\pi,\mu} \sum_{\boldsymbol{x},\omega,o} \mu(\boldsymbol{x})\pi(\boldsymbol{\omega}|\boldsymbol{x}, o)\pi(o|\boldsymbol{x})R_{\boldsymbol{x}\boldsymbol{\omega}}$$

Maximize reward

$$\sum_{\boldsymbol{x},\boldsymbol{\omega},o} \mu(\boldsymbol{x})\pi(\boldsymbol{\omega}|\boldsymbol{x}, o)\pi(o|\boldsymbol{x}) = 1$$

Distribution

$$\sum_{\boldsymbol{x}} \mu(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x}) = \hat{\boldsymbol{\phi}}$$

Reproduce Context-Features

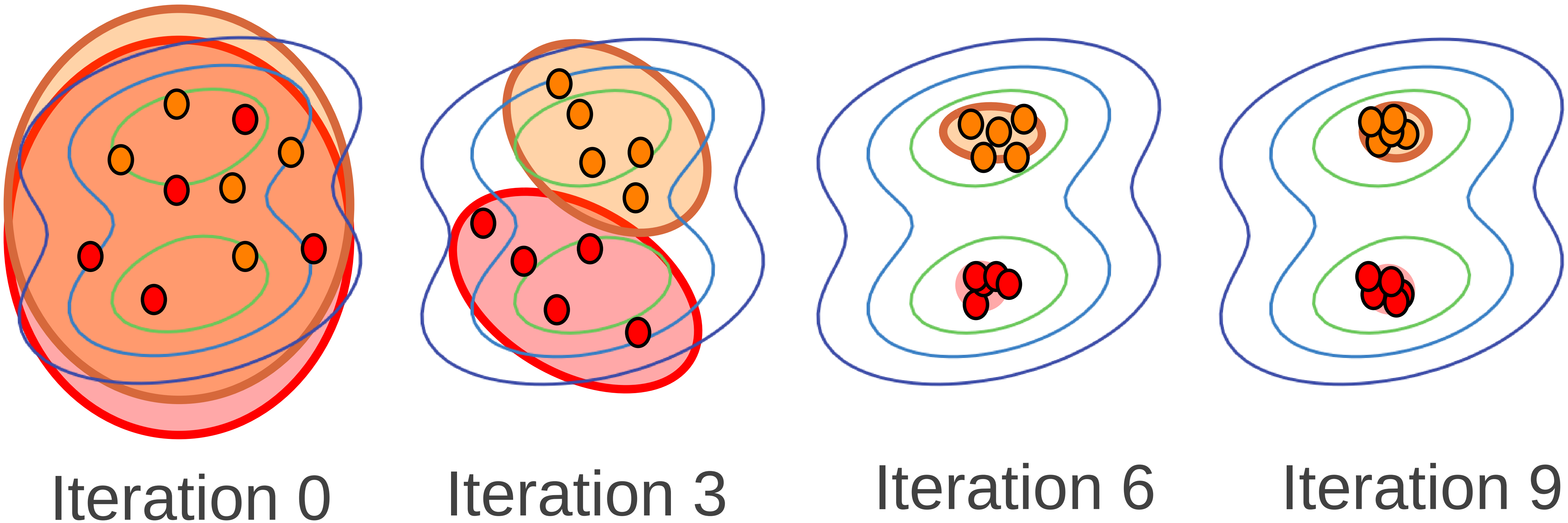
$$\epsilon \geq \sum_{\boldsymbol{x},\boldsymbol{\omega},o} \mu(\boldsymbol{x})\pi(\boldsymbol{\omega}, o|\boldsymbol{x}) \log \frac{\mu(\boldsymbol{x})\pi(\boldsymbol{\omega}, o|\boldsymbol{x})}{q(\boldsymbol{x}, \boldsymbol{\omega}, o)}$$

Stay close to the “data”, no wild exploration

$$\boxed{\kappa \geq \mathbb{E}\left[-p(o|\boldsymbol{x}, \boldsymbol{\omega}) \log p(o|\boldsymbol{x}, \boldsymbol{\omega})\right]}$$

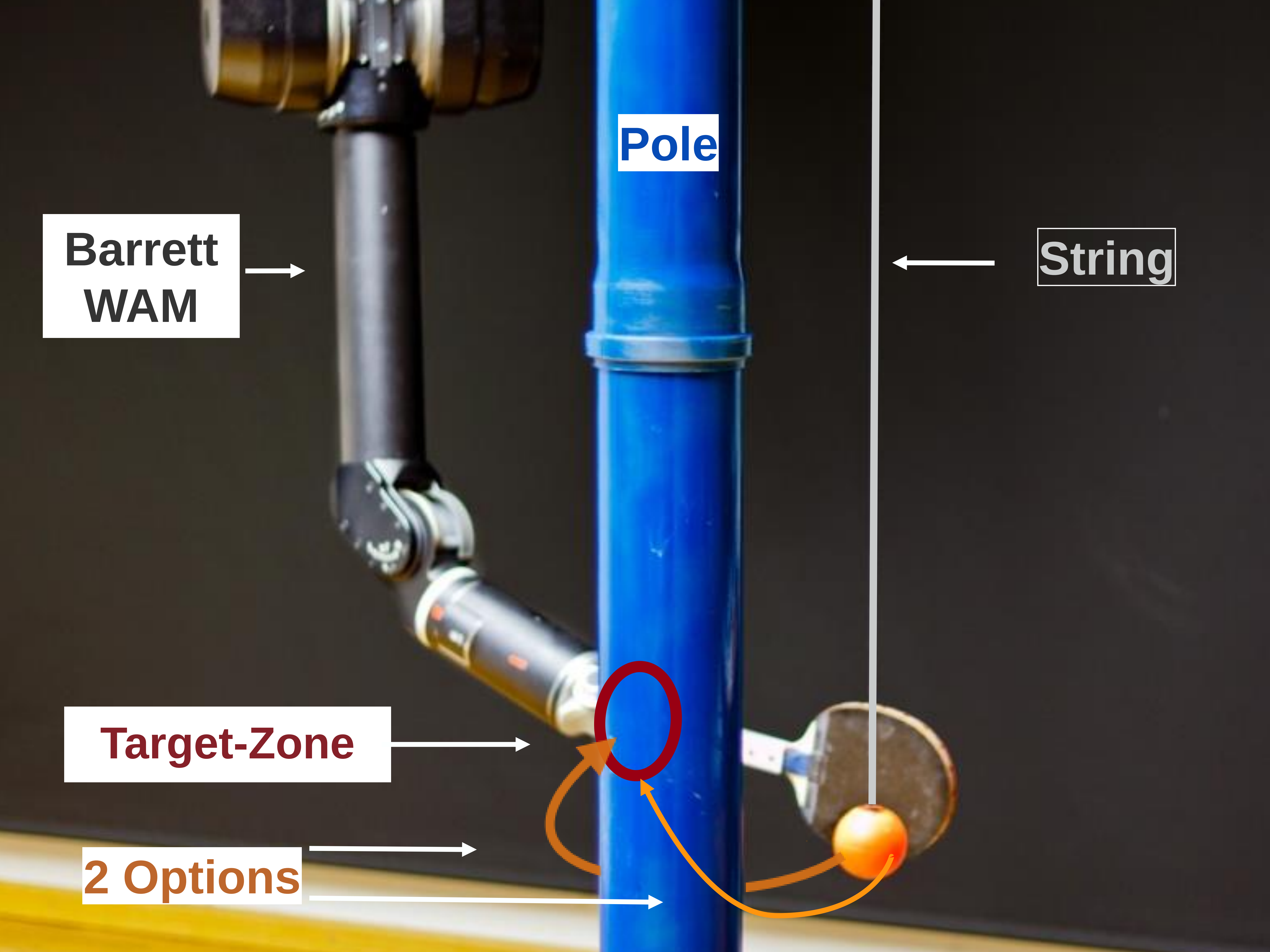
Versatile Solutions

HiREPS

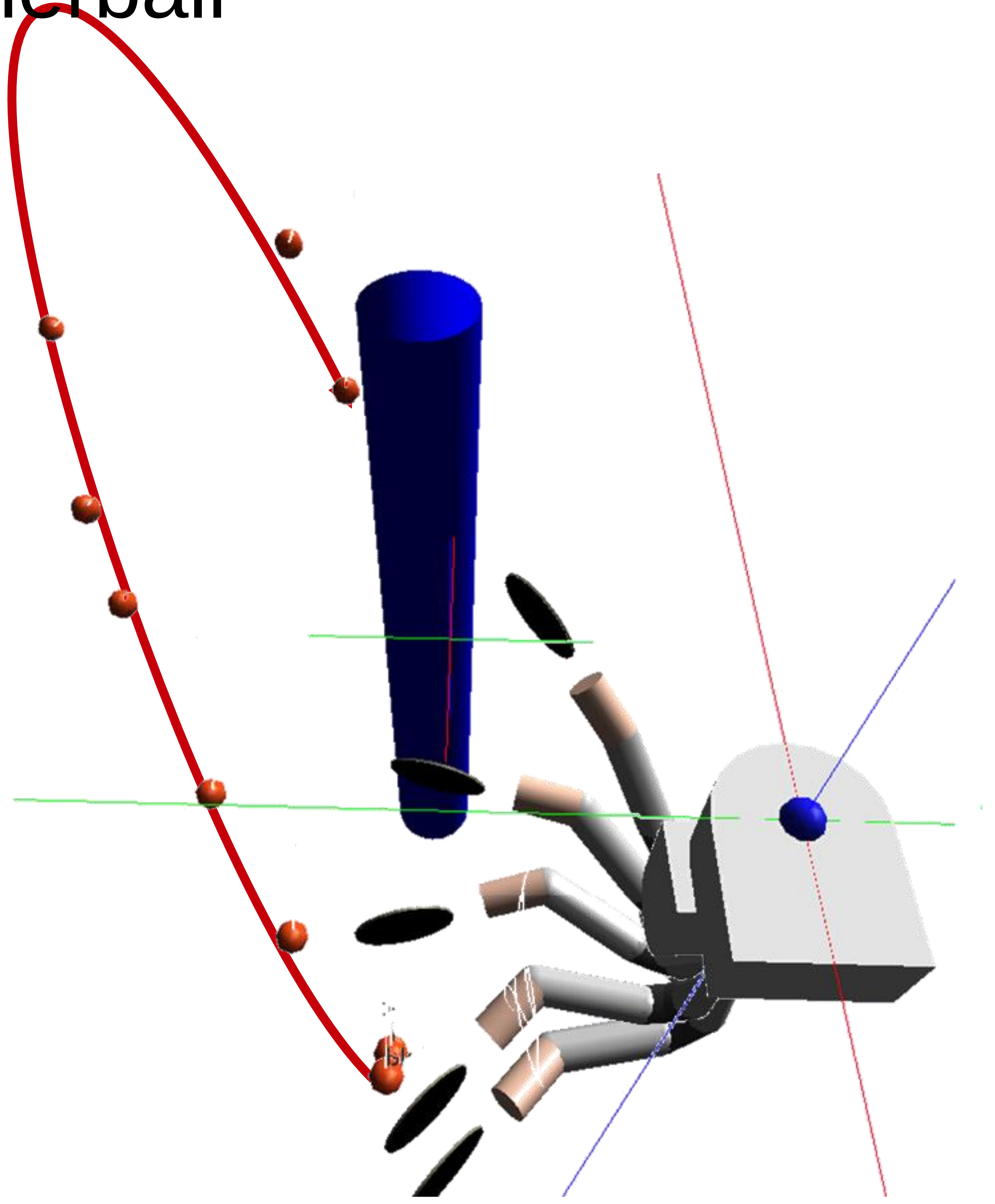


Learning of versatile, distinct solutions due to separation of options.

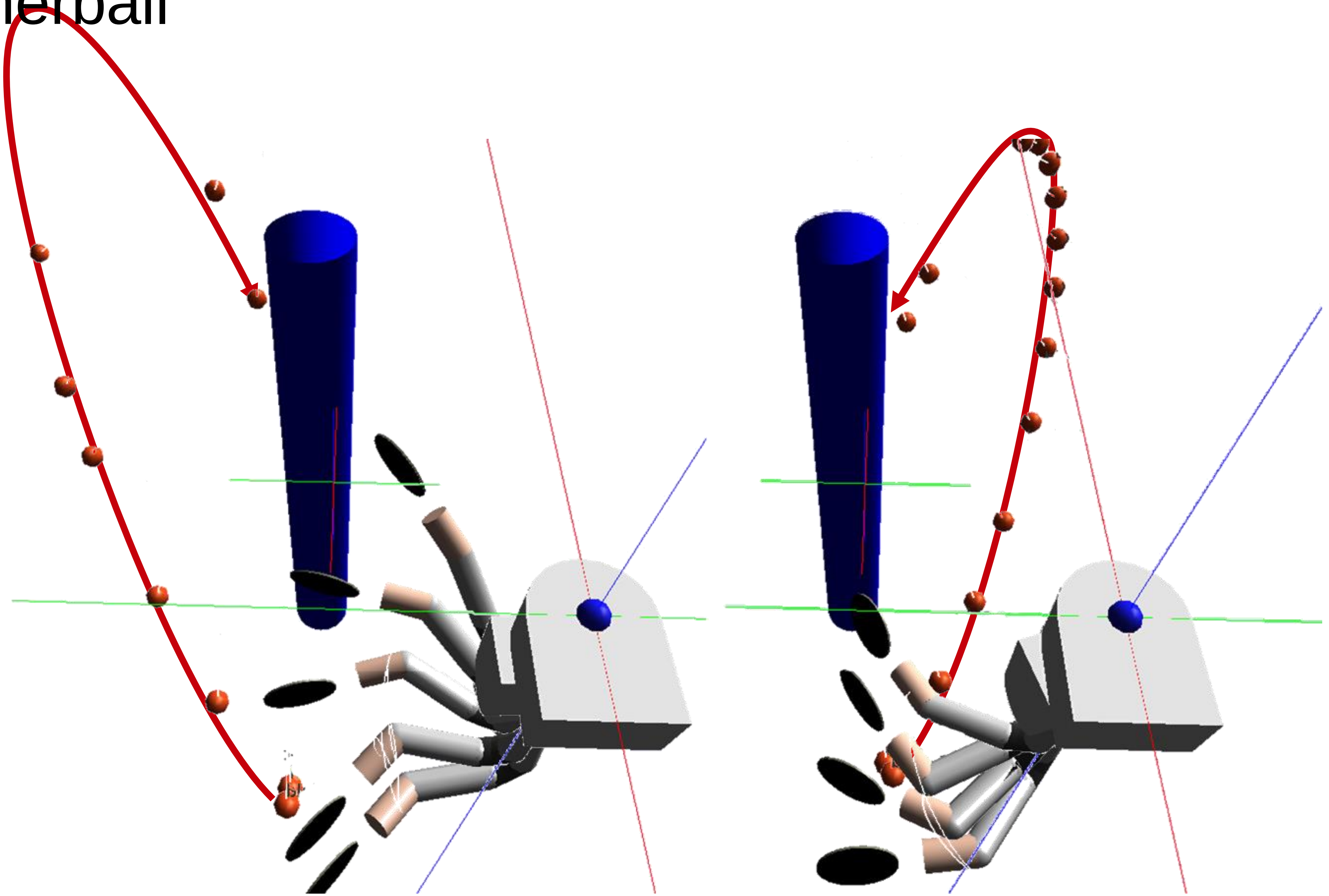
Tetherball


Pole
Barrett WAM
String
Target-Zone
2 Options

Tetherball

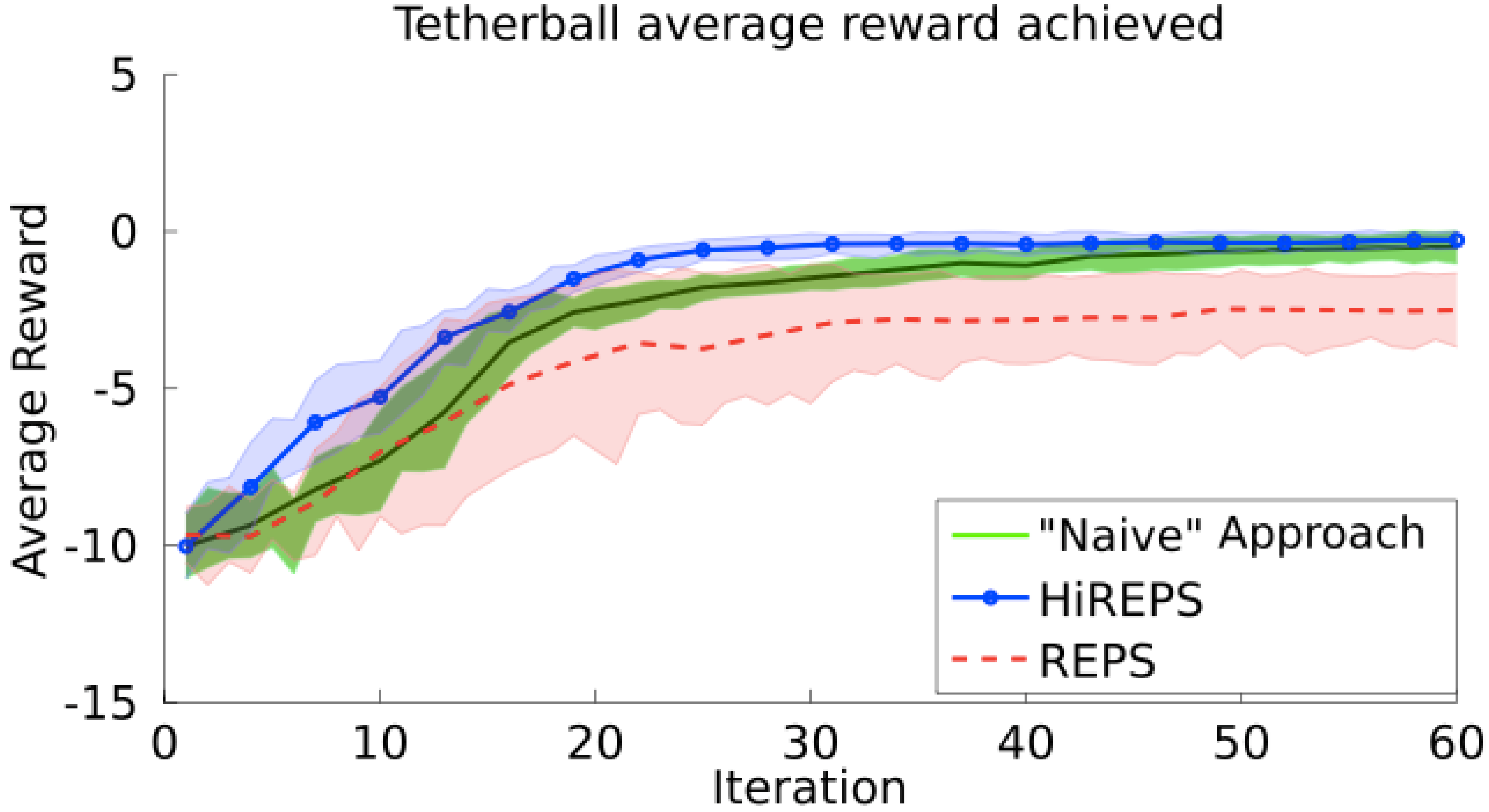


Tetherball



HiREPS learns distinct solutions.

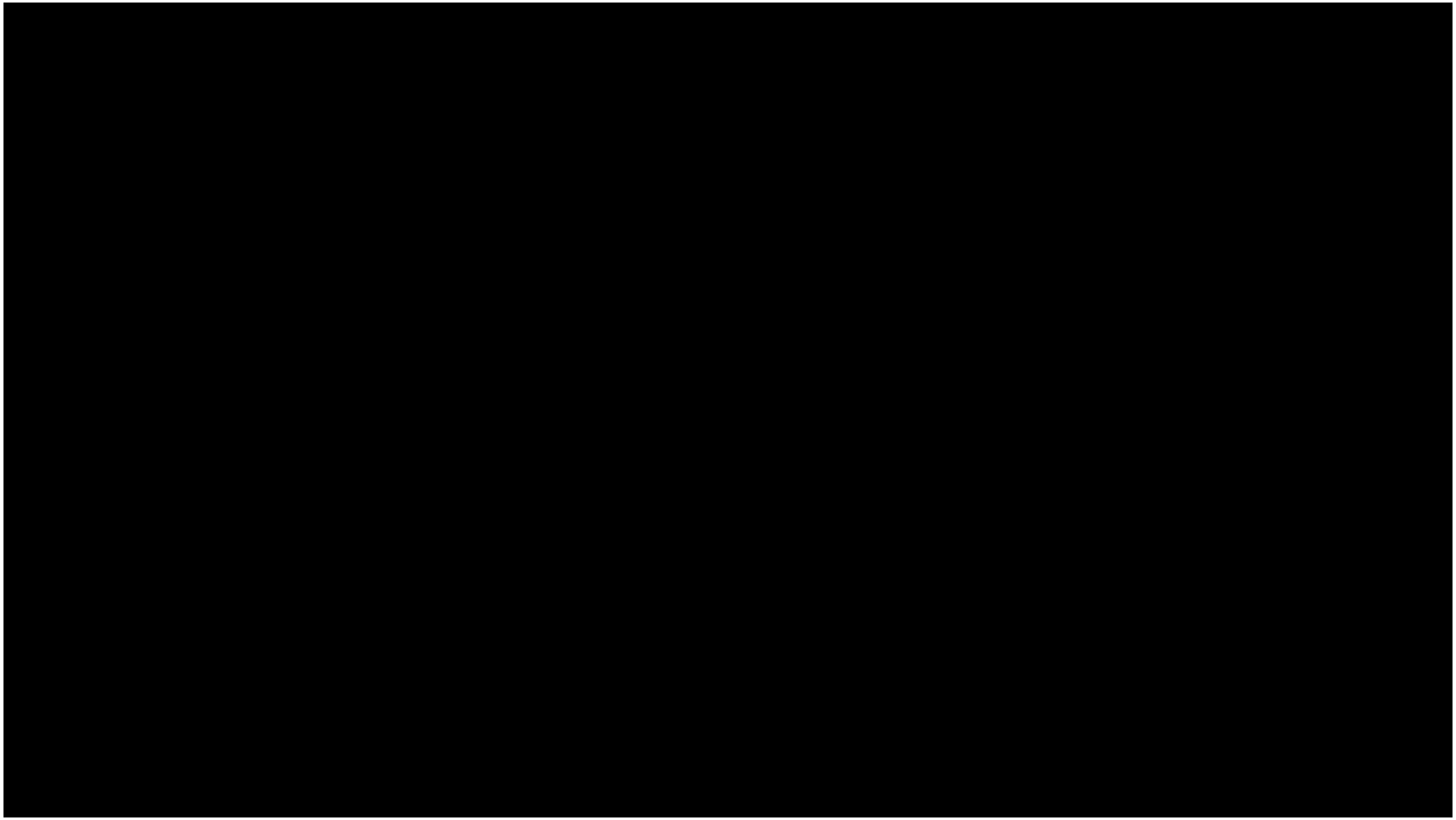
Results:



Finds several solutions

Improved convergence, no averaging over different solutions

Video



Outline of the Lecture

1. Introduction

2. Policy Updates by Weighted Maximum Likelihood

3. Relative Entropy Policy Search (REPS)

4. REPS for Contextual Policy Search

5. Learning Versatile Solutions

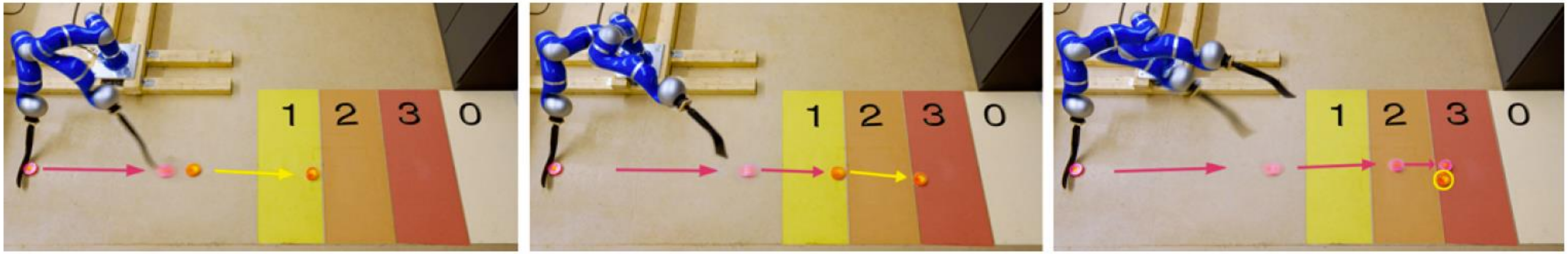
6. **Sequencing Movement Primitives**

Sequencing of Building Blocks

Many motor tasks require a sequence of elemental building blocks to fullfill the task

- ➡ The context of later building blocks depends on the execution of previous ones
- ➡ We need to learn the long-term effects of the building blocks

Sequential Robot-Hockey Task: place target-puck in reward zone ‚3' after three shoots



Sequencing of Building Blocks

Goal: Sequence several building blocks k with parameters $\boldsymbol{\theta}_k$,

React to the outcome $\boldsymbol{x}_k$ of the previous action $\boldsymbol{\theta}_{k-1}$

Introduce K decision steps

For each decision step, learn individual upper-level policy

Maximize the **expected return over all decision steps**

$$J_\pi = \sum_{k=1}^{K} \iint \mu_k(\boldsymbol{x}) \pi_k(\boldsymbol{\theta}|\boldsymbol{x}) R^k_{\boldsymbol{x}\boldsymbol{\theta}} d\boldsymbol{\theta} d\boldsymbol{x}$$

Context distributions: $\mu_k(\boldsymbol{s})$ is specified by the previous policies $\pi_{l<k}(\boldsymbol{\theta}_l|\boldsymbol{x}_l)$

$$\mu_k(\boldsymbol{x}') = \iint \mu_{k-1}(\boldsymbol{x}) \pi_{k-1}(\boldsymbol{\theta}|\boldsymbol{x}) p(\boldsymbol{x}'|\boldsymbol{x}, \boldsymbol{\theta}) d\boldsymbol{x} d\boldsymbol{\theta}$$

Sequential REPS

How to compute the policy $\pi_k(\boldsymbol{\theta}|\boldsymbol{x})$ **?**

Exploit: Maximize reward

$$\mathrm{argmax}_{p(\boldsymbol{x},\boldsymbol{\theta})} \sum_k \sum_{\boldsymbol{\theta},\boldsymbol{x}} p_k(\boldsymbol{x},\boldsymbol{\theta}) R_{\boldsymbol{x}\boldsymbol{\theta},k}$$

Explore: Stay close to old exploration policy $q_k(\boldsymbol{x},\boldsymbol{\theta})$

$$\text{s.t.: } \mathrm{KL}\left(p_k(\boldsymbol{x},\boldsymbol{\theta}) || q_k(\boldsymbol{x},\boldsymbol{\theta})\right) \leq \epsilon, \ \forall k$$

Estimate a distribution

$$\sum_{\boldsymbol{x},\boldsymbol{\theta}} p_k(\boldsymbol{x},\boldsymbol{\theta}) = 1, \ \forall k$$

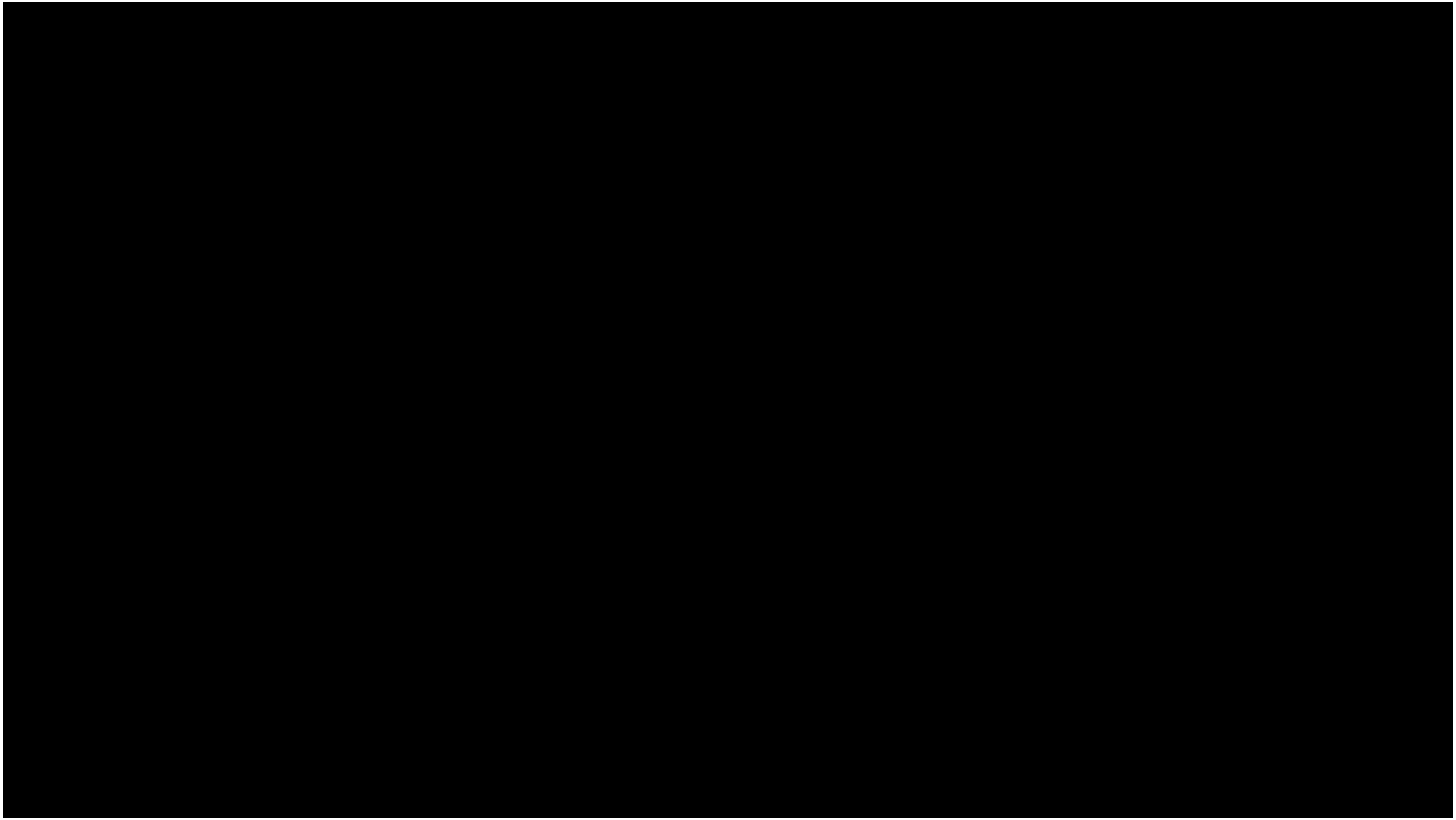
Reproduce context distribution

$$p_k(\boldsymbol{x}') = \sum_{\boldsymbol{x},\boldsymbol{\theta}} p_{k-1}(\boldsymbol{x},\boldsymbol{\theta}) p(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{\theta})$$

Solution: $p_k(\boldsymbol{x},\boldsymbol{\theta}) \propto q_k(\boldsymbol{x},\boldsymbol{\theta}) \exp\left(\frac{R_{\boldsymbol{x}\boldsymbol{\theta},k} + \mathbb{E}_{p(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{\theta})}[V_{k+1}(\boldsymbol{x}')] - V_k(\boldsymbol{x})}{\boldsymbol{\eta}_k}\right)$

$\mathbb{E}_{p(\boldsymbol{x}'|\boldsymbol{x},\boldsymbol{\theta})}[V_{k+1}(\boldsymbol{x}')]$ ... Encodes long-term reward

Video



Conclusion

Probabilisitic Policy Search Methods:

➡Policy update reduces to **weighted maximum likelihood estimates** of the parameters

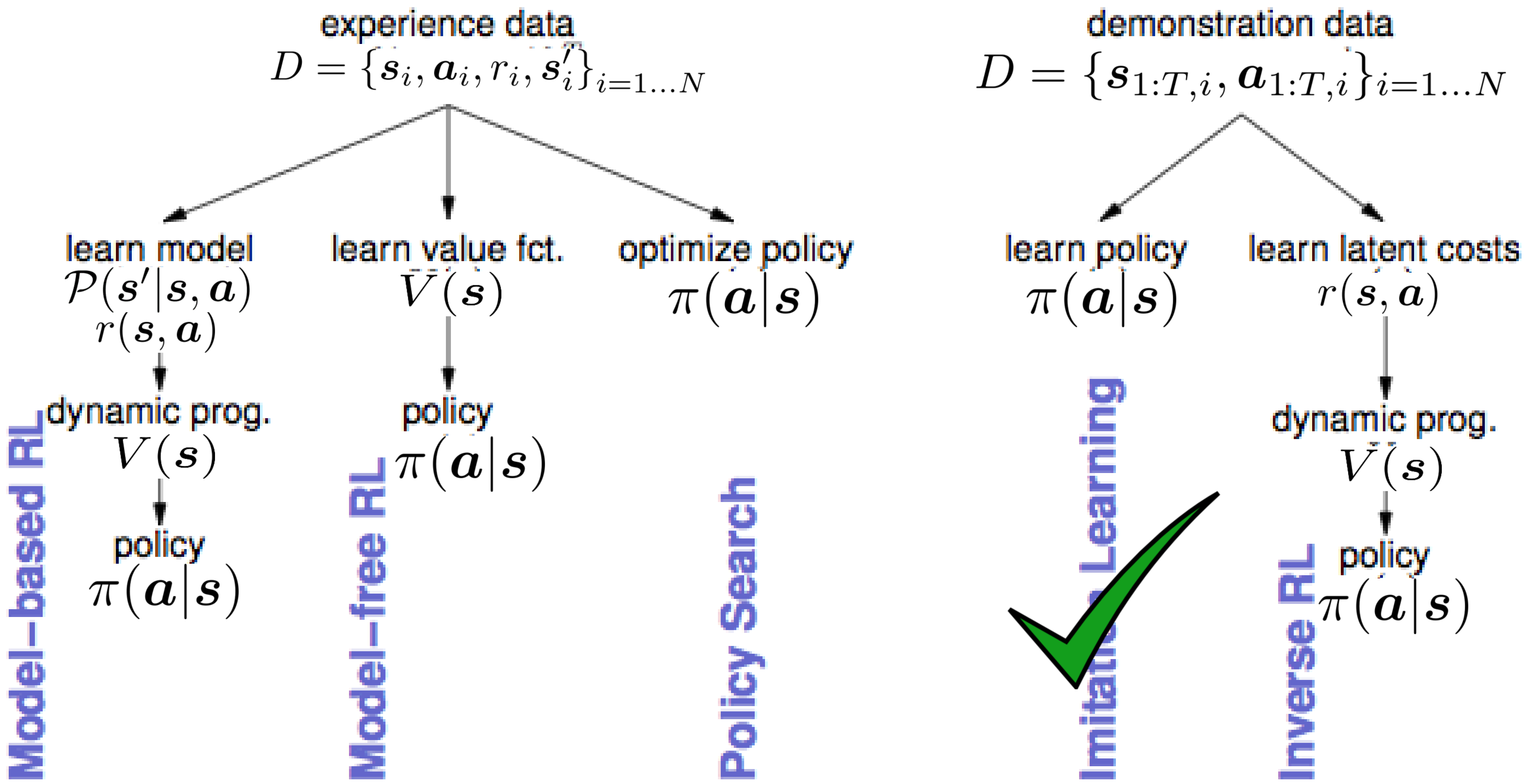
➡Any type of **structured policy** can be used (e.g. mixture model)

➡Weights are specified by **exponential transformation** of the returns

➡REPS **optimizes the temperature** of this transformation to match a desired Kullback-Leibler divergence

➡**Contextual policy search** can be used for for multi-task learning

Bigger Picture



Wrap-Up: Model-Based

Model Complexity: Very High

Learn forward model $f : (\mathcal{R}^{|S|+|A|}) \rightarrow \mathcal{R}^{|S|}$

Need to be able to do dynamic programming (e.g. LQR)

Small modelling error can have a big effect on the policy

Scalability: Poor (with some positive exceptions)

Learning high-dimensional (or discontinous) models is very hard

Data-Efficiency: Excellent

Use every transition to learn model

Model can be reused for different tasks

Other Limitations:

Distance between two policies is hard to control

Huge computation times

Wrap-Up: Value Based

Model Complexity: OK

- Learn Q-Function $Q : (\mathcal{R}^{|S|+|A|}) \to \mathcal{R}$
- Small function approximation error can have a big effect on the policy

Scalability: Poor (with some positive exceptions)

- Function approximation in high-dimensional state spaces is difficult
- Policy is hard to obtain in high-dimensional action spaces

Data-Efficiency: OK (online TD learning) to good (batch methods)

- Batch: Reuse every transition
- Online: Every transition is just used once

Other Limitations:

- Policy update is again unbounded, might lead to oscillations

Wrap-Up: Step-Based Policy Search

Model Complexity: None (no approximation errors)

Need to evaluate reward to come $Q_t^{[i]}$

Scalability: Good

Parametrized polices are a compact representation that allow learning also for high-D robots

Only works for a medium amount of parameters (a few hundred)

Data-Efficiency: Poor

Use every state action pair with reward to come

High variance in reward to come due to exploration in action space

Other Limitations:

Mainly used for learning single trajectories (e.g. DMPs)

Wrap-Up: Episode-Based Policy Search

Model Complexity: None (no approximation errors)

Need to evaluate return for each trajectory $R^{[i]}$

Scalability: Good

Parametrized policies

Only works for a small amount of parameters (around hundred)

Data-Efficiency: Poor

Each rollout is just one sample

High variance in returns in case of stochastic environments

Other Limitations:

Mainly used for learning single trajectories (e.g. DMPs)