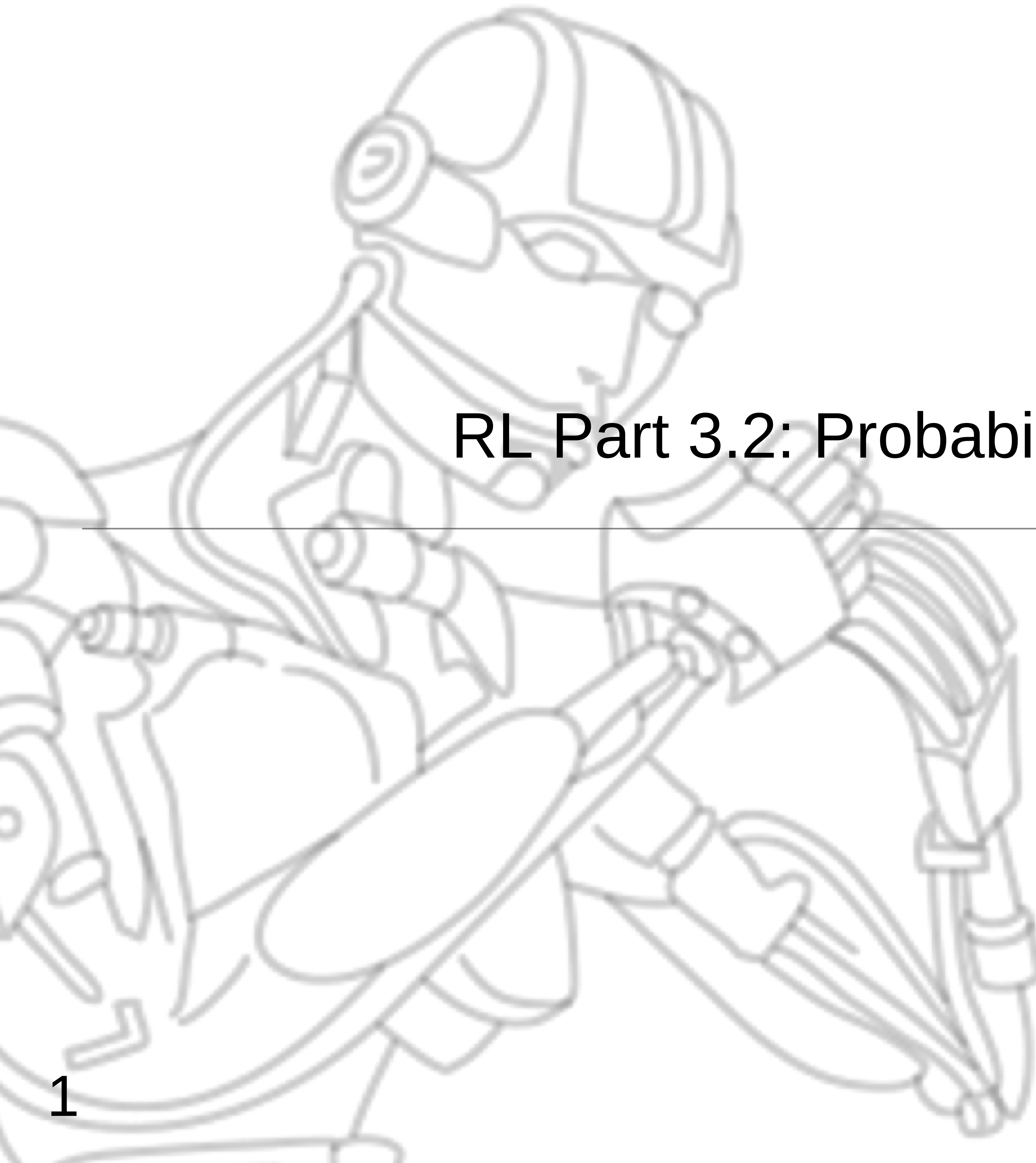


RL Part 3.2: Probabilistic Policy Search

Gerhard Neumann
Jan Peters

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.
- Say still need a lot of samples and **we need to tune the learning rate**
- Learning the **exploration rate** is still an open problem

Outline of the Lecture

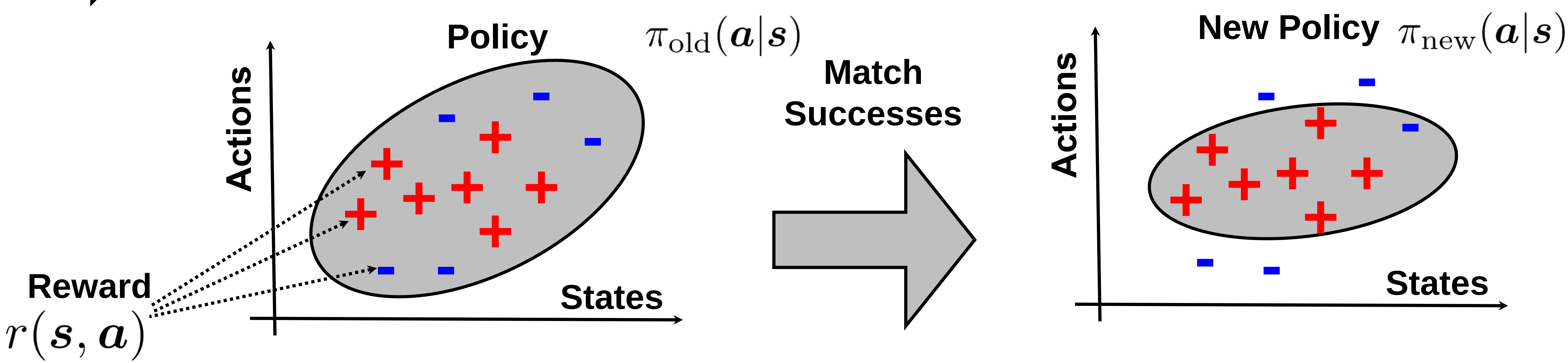
1. Introduction
2. Policy Updates by Weighted Maximum Likelihood
3. Relative Entropy Policy Search (REPS)
4. REPS for Contextual Policy Search
5. Conclusion

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).

- Why? We still need to explore!
- Create policies such that $\pi_{\text{new}}(\boldsymbol{a}|\boldsymbol{s}) \propto \pi_{\text{old}}(\boldsymbol{a}|\boldsymbol{s})r(\boldsymbol{s},\boldsymbol{a})$

⇨ Only possible for non-negative reward functions $r(\boldsymbol{s},\boldsymbol{a})$



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

Quick Recap: Episode-based Policy Search

For now, we will consider the episode-based setting

The policy evaluation strategy is needed to assess the quality of the samples

Episode-Based:

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

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

Data-set used for policy update

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

One data-point per trajectory

Episode-based policy evaluation strategy

The policy evaluation strategy is needed to assess the quality of the samples

Upper-level Policy :

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

$\pi(\boldsymbol{\theta}; \boldsymbol{\omega})$ is called **upper-level policy, e.g.** $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$

$\boldsymbol{\omega}$ … 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.,

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

Works for a **moderate number of parameters (e.g. DMPs)**

Policy Update by Sucess Matching

Iterate:

Sample state-actions with current policy $\boldsymbol{\theta}^{[i]} \sim \pi(\boldsymbol{\theta}; \boldsymbol{\omega}_k)$

Compute Target Distribution: „Success" weighted policy on the samples

$$\tilde{\pi}(\boldsymbol{\theta}^{[i]}) \propto w^{[i]} \pi_k(\boldsymbol{\theta}^{[i]}; \boldsymbol{\omega}_k) \quad w^{[i]} = f(R^{[i]})$$

We need to transform the reward with f in **a non-negative weight** (improper probability distribution)

Fit new parametric policy $\pi(\boldsymbol{\theta}^{[i]}; \boldsymbol{\omega}_{k+1})$ to target distribution

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

⇨ **Weighted Maximum Likelihood Estimate**

Policy Updates by Weighted ML

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

Why is it cool?

No learning rate involved

Can be **computed efficiently** for many distributions

We can directly „jump“ to the desired distribution

Why can we do a **weighted ML estimate** with $w^{[i]}$ as weights?

For now, we will assume that the weights $w^{[i]}$ are given

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 Updates by Weighted ML

Why can we do a **weighted ML estimate** with $w^{[i]}$ as weights?

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

We can do that by minimizing the expected KL between $\tilde{\pi}(\boldsymbol{\theta}^{[i]})$ and $\pi(\boldsymbol{\theta};\boldsymbol{\omega})$

Minimizing the KL is the same as maximizing the likelihood!!

$$\boldsymbol{\omega}_{k+1} = \mathrm{argmin}_{\boldsymbol{\omega}} \quad \mathrm{KL}(\tilde{\pi}(\boldsymbol{\theta}^{[i]})||\pi(\boldsymbol{\theta}^{[i]};\boldsymbol{\omega}))$$

$$= \mathrm{argmin}_{\boldsymbol{\omega}} \quad \int \tilde{\pi}(\boldsymbol{\theta}) \log \frac{\tilde{\pi}(\boldsymbol{\theta})}{\pi(\boldsymbol{\theta};\boldsymbol{\omega})} d\boldsymbol{\theta}$$

$$\approx \mathrm{argmax}_{\boldsymbol{\omega}} \quad \sum_i \frac{\tilde{\pi}(\boldsymbol{\theta}^{[i]})}{\pi(\boldsymbol{\theta}^{[i]};\boldsymbol{\omega}_k)} \log \pi(\boldsymbol{\theta}^{[i]};\boldsymbol{\omega})$$

We sampled from the old policy

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

Weighted Maximum Likelihood Solutions…

If the upper-level policy $\pi(\boldsymbol{\theta};\boldsymbol{\omega})$ is **Gaussian** $\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu},\boldsymbol{\Sigma})$ then mean and covariance are given by:

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

Weighted mean

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

Weighted covariance

Full covariance matrix → **correlated exploration** in parameter space

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

Weighted Maximum Likelihood Solutions…

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

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

Expectation-Maximization Based Algorithms

One way of derivating the weighted ML updates

EM algorithms introduce a reward event C

$$w^{[i]} = p(C = 1 | \boldsymbol{\tau}^{[i]}) \propto \exp(\beta R^{[i]})$$

Hence, the weight is given by an **exponential transformation of the return**

Expectation Maximization

Some notes on Expectation-Maximization in this context

EM is a method for Max. Likelihood in the case of latent (unobserved) variables

Observed variable: Reward Event $C = 1$ (we want to get reward)

Unobserved variable: Trajectory $\boldsymbol{\tau}$ (or parameters) that created the reward event

E-Step: Estimate new desired distribution

M-Step: Estimate new policy parameters from the weighted samples

Step-based Policy Search Versions of EM: PoWER (Kober 2008), Reward-Weighted Regression (Peters 2007)

Expectation Maximization

Some notes on Expectation-Maximization in this context

Formally, the reward transformation is hard to motivate

$$w^{[i]} = p(C = 1|\boldsymbol{\tau}^{[i]}) \propto \exp(\beta R^{[i]})$$

β … temperature parameter. Needs to be hand-tuned (task specific)

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

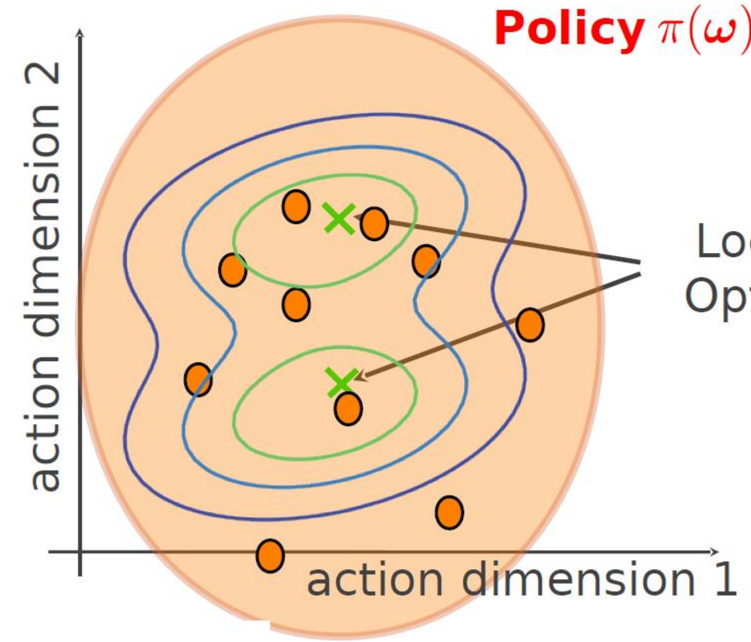
$$\mathbb{E}_{p(\boldsymbol{\tau})}\left[\exp(R(\tau))\right] \neq \exp(\mathbb{E}_{p(\boldsymbol{\tau})}\left[R(\tau)\right])$$

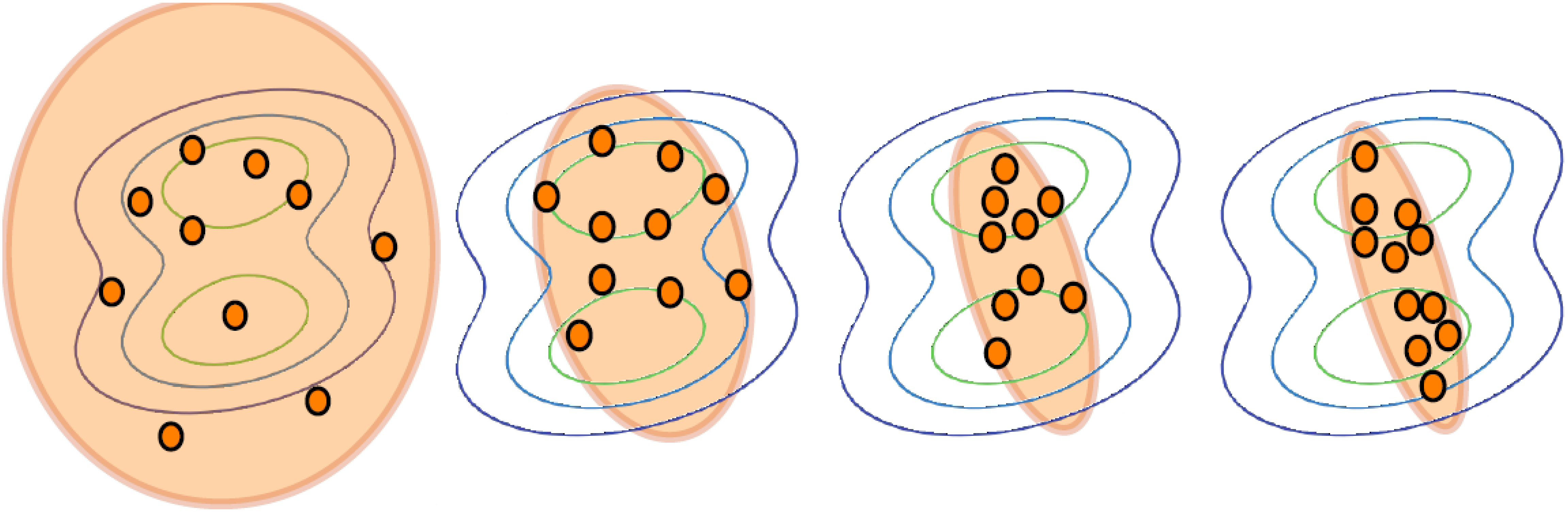
The objective gets „risk attracted“

For moderately stochastic environments it still works well

Illustration on weighted ML

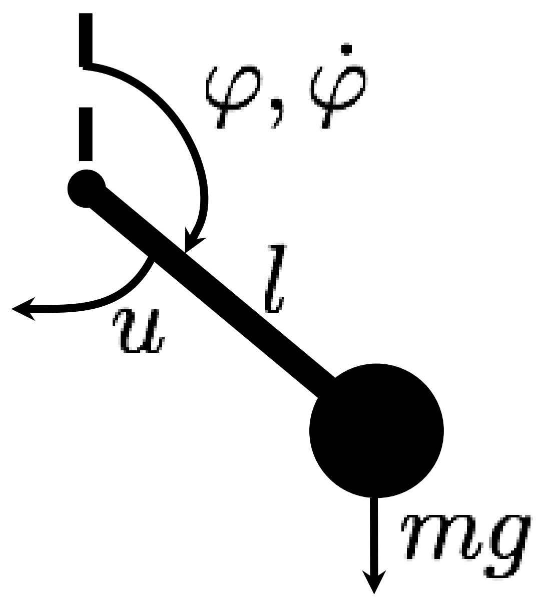
Example for a 2D parameter space:





Underactuated Swing-Up

- swing heavy pendulum up



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

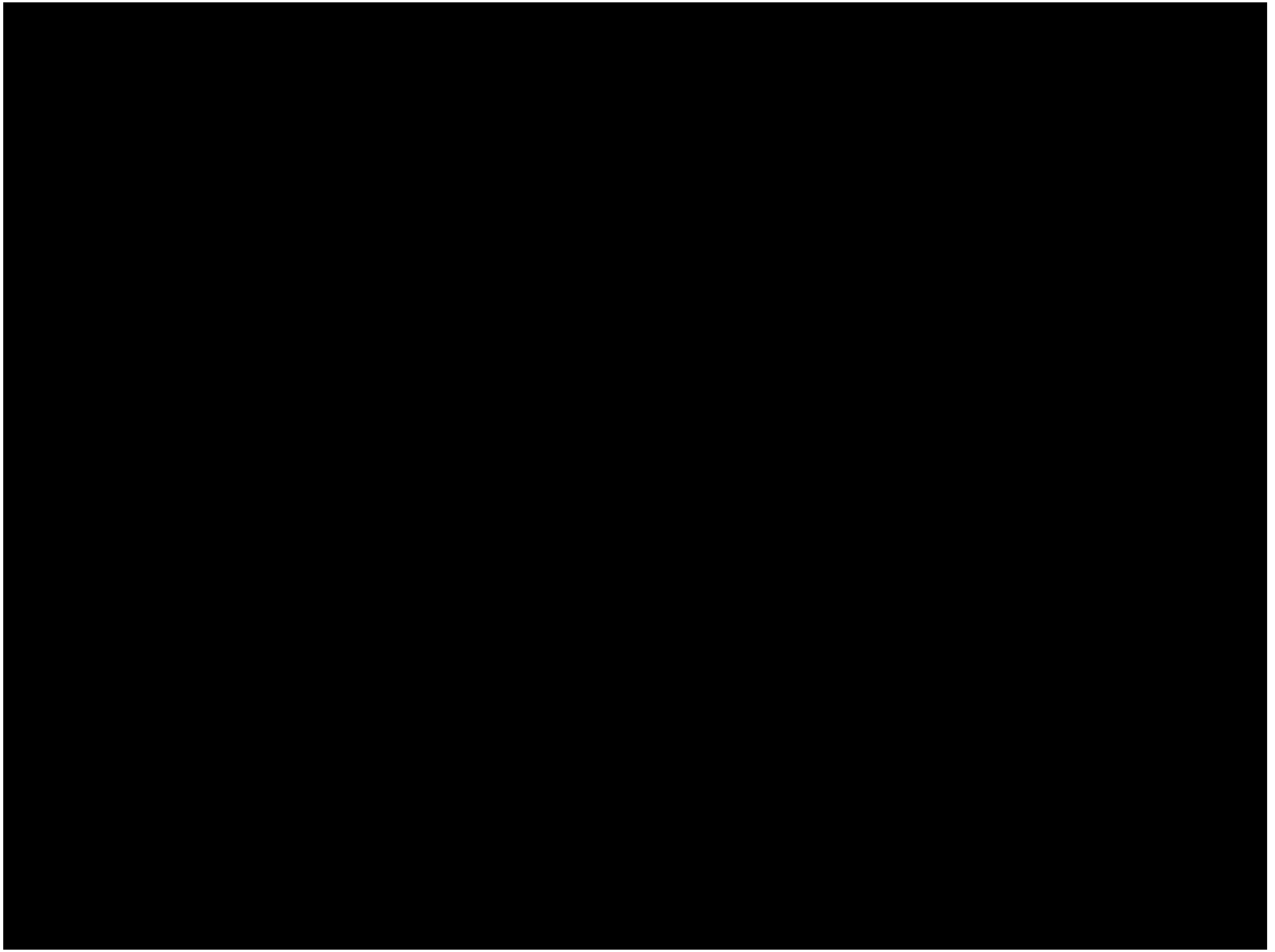
- motor torques limited, Policy: DMPs

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

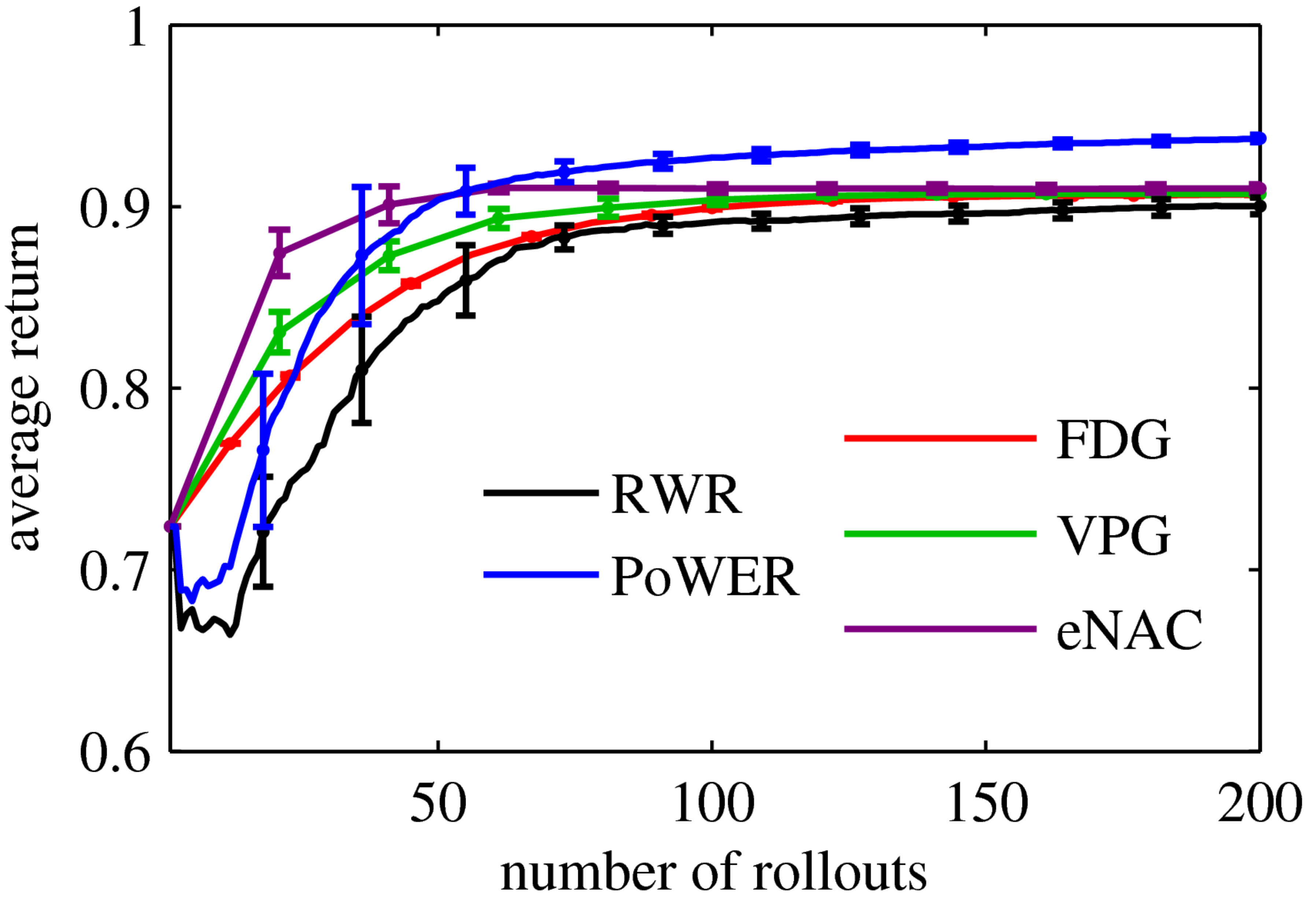
- reward function

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

(Schaal, NIPS 1997; Atkeson, ICML 1997)



Underactuated Swing-Up



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

Ball in the Cup

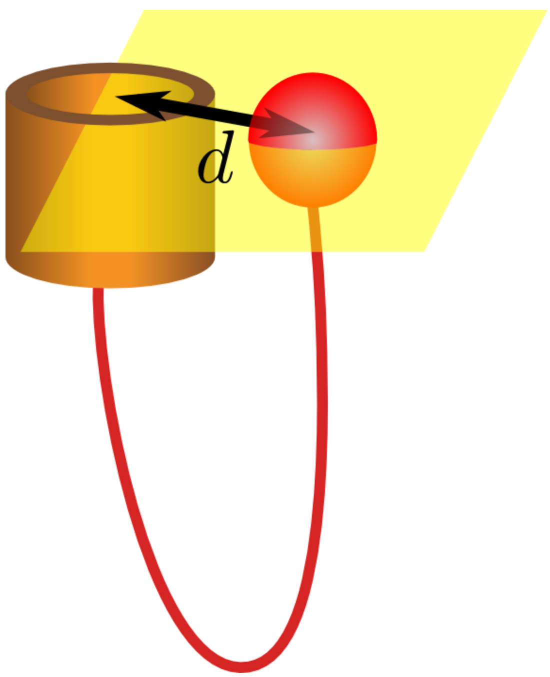
Video with Ball In the Cup

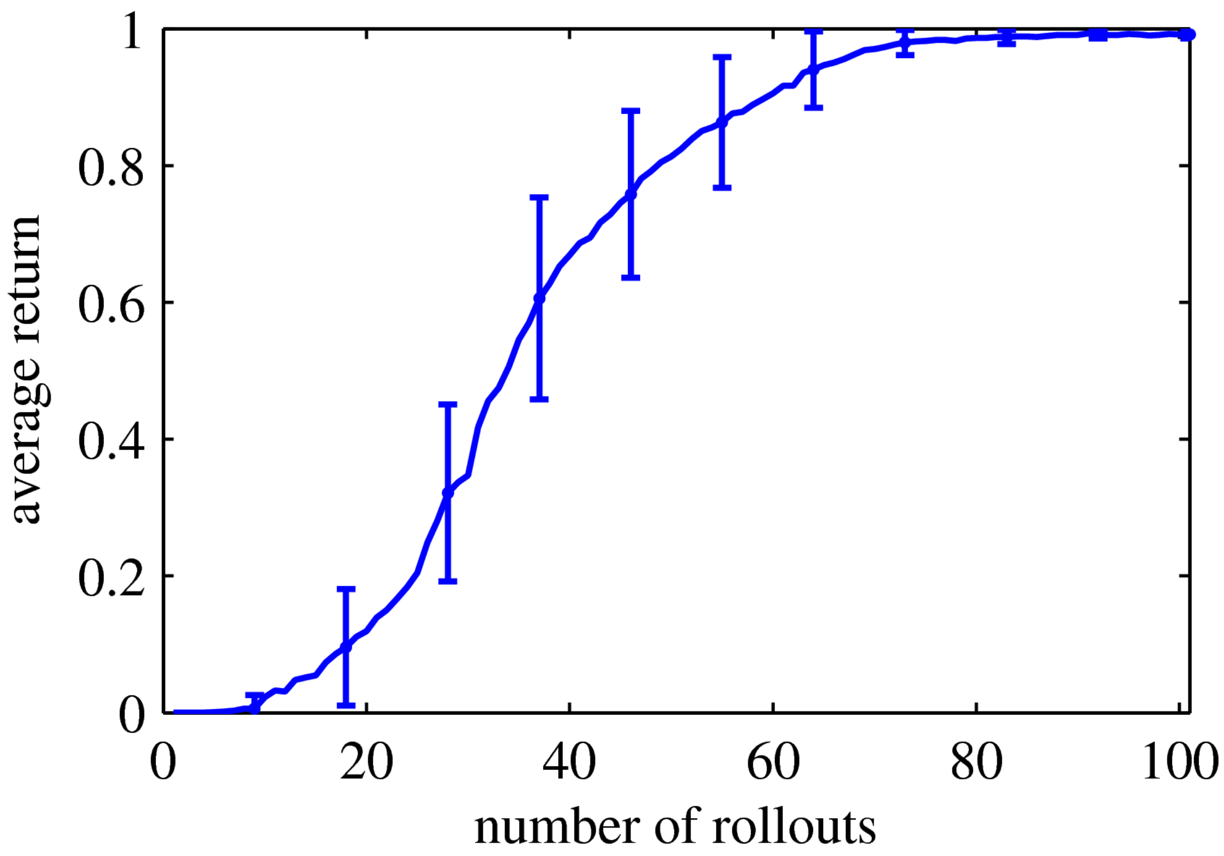
Ball-in-a-Cup

Reward function:

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

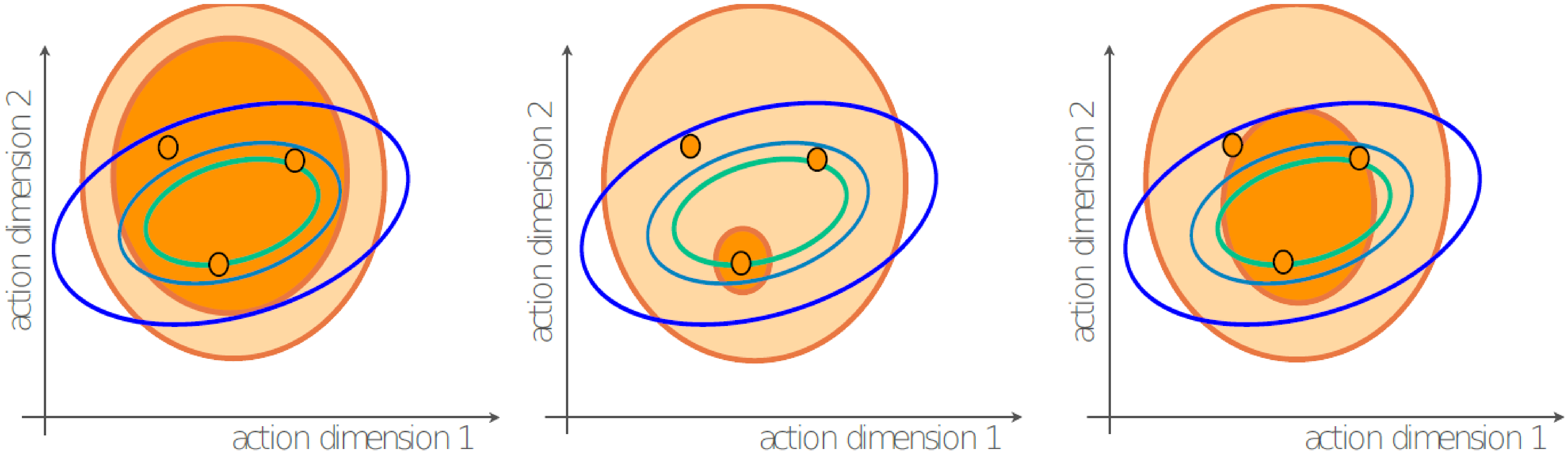
Policy: DMPs





Policy Search: Choosing the metric/step width

What is a good desired distribution for the policy update?



Too conservative

Too greedy

Moderate tradeoff between exploration and exploitation

We want to have invariance to:

Transformations of the reward

Transformations of the parameter space

For the **weighted ML** methods: How to choose the reward transformation? How to choose the temperature β ?

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

Preserve locality

Coming back to the question: What is a good metric for the policy update?

Goal: Maximize the expected long-term reward

$$J_{\boldsymbol{\theta}} = \mathbb{E}_{\mu_0,\mathcal{P},\pi}\left[\sum_{t=1}^{T-1} r_t(\boldsymbol{s}_t, \boldsymbol{a}_t) + r_T(\boldsymbol{s}_T)\right]$$

But: We want to preserve locality to achieve a „safe" policy update

$$\text{s.t.:} M(\pi_{k+1}, \pi_k) \leq \epsilon$$

M... metric on the policy update

Used metrics in practice

Euclidian distance in parameter space

$$M(\boldsymbol{\omega}, \boldsymbol{\omega}_{\text{old}}) = ||\boldsymbol{\omega} - \boldsymbol{\omega}_{\text{old}}||^2$$

Used implicitely by all standard policy gradient approaches

Information-theoretic distance in probability distribution space

$$M(\boldsymbol{\omega}, \boldsymbol{\omega}_{\text{old}}) = \text{KL}(\pi_{\boldsymbol{\omega}} || \pi_{\boldsymbol{\omega}_{\text{old}}})$$

Measures the ‚distance‘ between old and new policy in probability space

Policy update: $\text{KL}(\pi_{\boldsymbol{\omega}} || \pi_{\boldsymbol{\omega}_{\text{old}}}) \leq \epsilon$

Invariant to transformations of parameter vector or reward

Stay close to the data!!

Used metrics in practice

Two different methods have been used:

Natural Policy Gradient (Peters & Schaal, 2008)**:**

$$\mathrm{KL}(\pi_{\boldsymbol{\omega}} || \pi_{\boldsymbol{\omega}_{\mathrm{old}}}) \approx \Delta\boldsymbol{\omega}^T \boldsymbol{F} \Delta\boldsymbol{\omega}$$

$\boldsymbol{F}$ … Fisher information matrix

Second order Taylor approximation of the KL

➡ Update is still **done in parameter space**

Relative Entropy Policy Search

Directly **optimize probabilities of the samples** such that

$$\mathrm{KL}(\pi_{\boldsymbol{\omega}} || \pi_{\boldsymbol{\omega}_{\mathrm{old}}}) \leq \epsilon$$

Subsequently**, fit policy to weighted samples** to get the new policy

Relative Entropy Policy Search Peters 2010

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

Old Policy

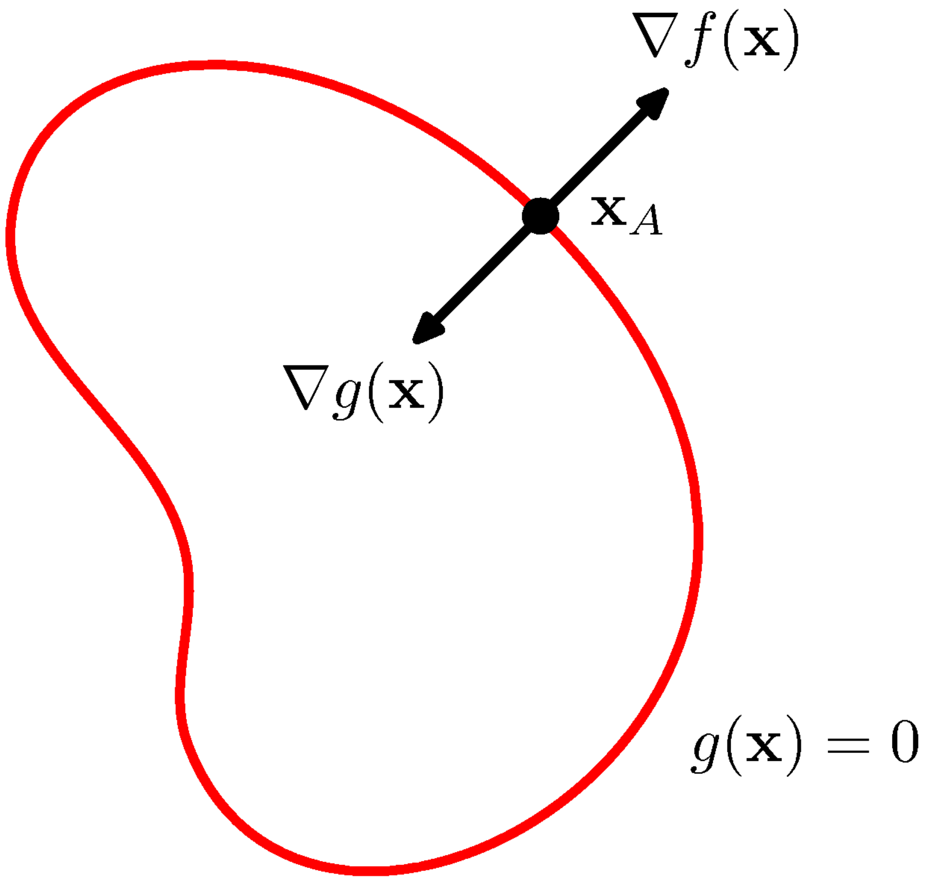
Policy Update is formulated as constraint optimization problem

Cookbook for Constraint Optimization Problems

Given: **Constraint Optimization Problem**

$$\max_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{s.t: } g_i(\boldsymbol{x}) = 0, \quad \forall i$$

Suppose $\boldsymbol{x}$ is on the valid constraint surface, i.e., $g(\boldsymbol{x}) = 0$



For a taylor expansion around $\boldsymbol{x}$

$$g(\boldsymbol{x} + \boldsymbol{\epsilon}) \approx g(\boldsymbol{x}) + \boldsymbol{\epsilon}^T \nabla g(\boldsymbol{x})$$

From $g(\boldsymbol{x} + \boldsymbol{\epsilon}) = 0$, it follows that

$$\boldsymbol{\epsilon}^T \nabla g(\boldsymbol{x}) = 0$$

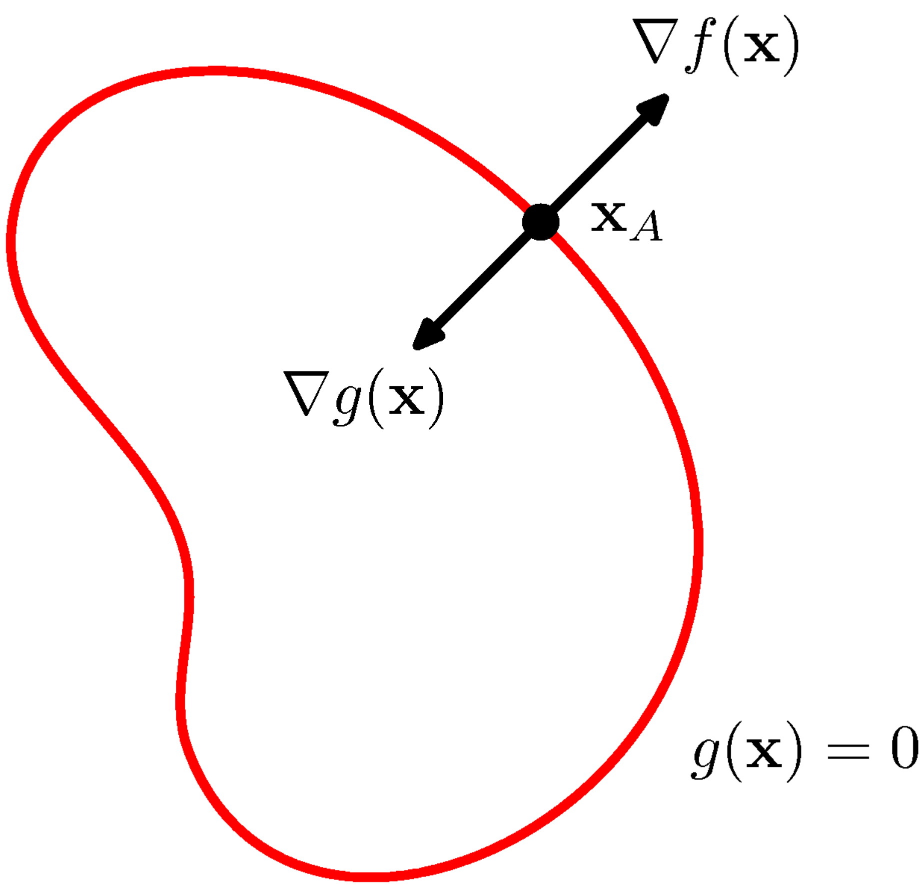
i.e., $\nabla g(\boldsymbol{x})$ is normal to the constraint surface

Cookbook for Constraint Optimization Problems

Given: **Constraint Optimization Problem**

$$\max_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{s.t: } g_i(\boldsymbol{x}) = 0, \quad \forall i$$

Now we look for a point $\boldsymbol{x}^*$on the constraint surface, that maximizes f



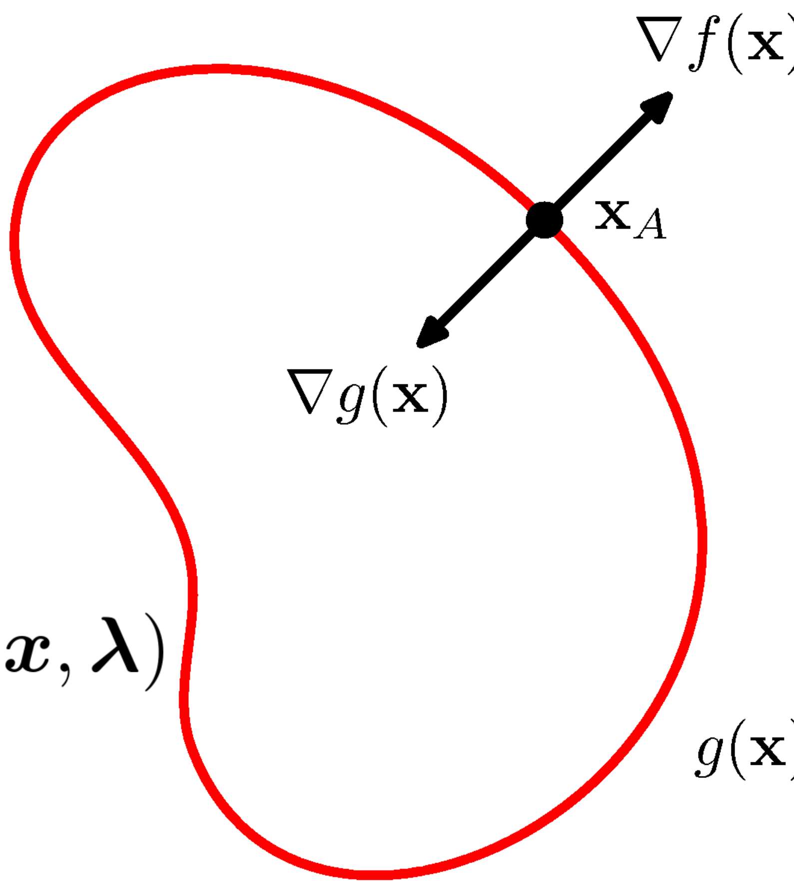
$\nabla f(\boldsymbol{x}^*)$ needs to be orthagonal to the constraint surface (otherwise we could increase f)

$\Rightarrow$ $\nabla f(\boldsymbol{x}^*)$ and $\nabla g(\boldsymbol{x})$ are (anti-)parallel

There must be a parameter λ such that

$$\nabla f + \lambda \nabla g = 0$$

Cookbook for Constraint Optimization Problems



Now we can introduce the **Lagrangian function**

$$L(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \lambda g(\boldsymbol{x})$$

and we want to find a stationary point of $L(\boldsymbol{x}, \lambda)$

$$\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \lambda) = 0 \qquad \frac{\partial \mathcal{L}(\boldsymbol{x}, \lambda)}{\partial \lambda} = 0$$

The lagrangian dual function is $g(\boldsymbol{\lambda}) = \max_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$

and, hence, $g(\boldsymbol{\lambda}) \geq f(\boldsymbol{x}) + \lambda g(\boldsymbol{x})$

For an optimal point $\boldsymbol{x}^*$ that solves the primal problem

$$g(\boldsymbol{\lambda}) \geq f(\boldsymbol{x}^*) + \lambda g(\boldsymbol{x}^*) = f(\boldsymbol{x}^*)$$

Since $g(\boldsymbol{\lambda}) \geq f(\boldsymbol{x}^*)$, the optimal λ is obtained by minimizing the dual

$$\lambda^* = \min g(\lambda)$$

Cookbook for Constraint Optimization Problems

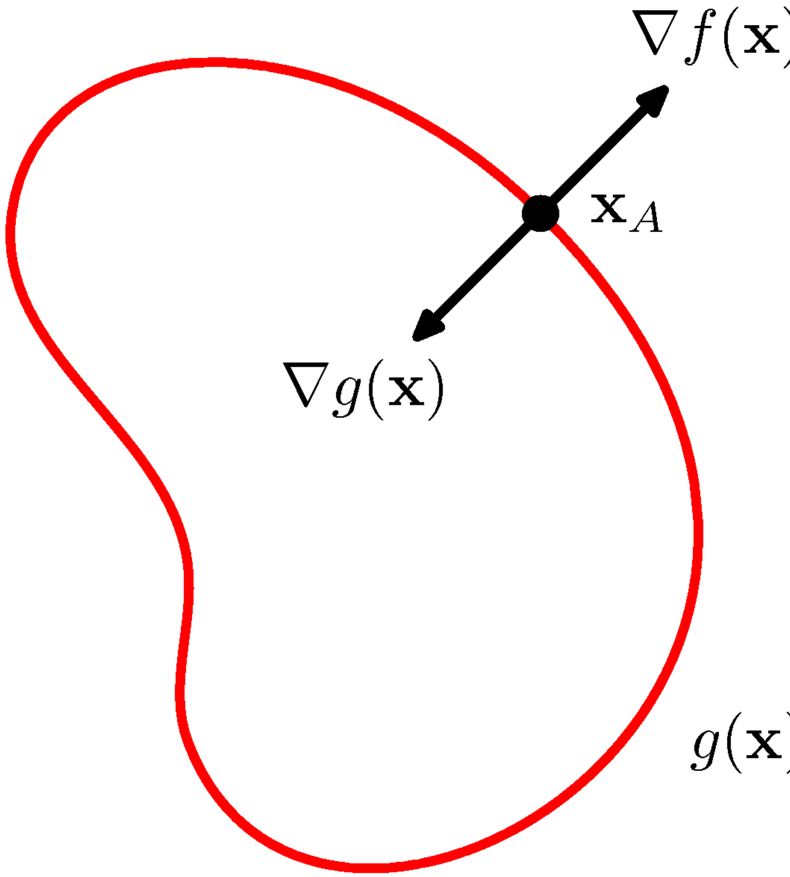
Why should we solve the dual problem?

$$\lambda^* = \min_\lambda g(\lambda)$$

It is often easier to solve (**less variables, unconstrained**)

It is **convex**, even if the original problem is not

However, the solution of the dual can be used to solve the **primal only under certain conditions**



Cookbook for Constraint Optimization Problems

Given: **Constraint Optimization Problem**

$$\max_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{s.t: } g_i(\boldsymbol{x}) = 0, \quad \forall i$$

1. Write down **Lagrangian with Lagrangian Multipliers**

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_i \lambda_i \boldsymbol{g}(\boldsymbol{x})$$

2. **Solve for optimal x** for a given $\boldsymbol{\lambda}$

$$\boldsymbol{x}^* = \text{argmax}_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$$

3. Set back in Lagrangian to obtain **dual function**

$$g(\boldsymbol{\lambda}) = \max_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$$

4. **Solve for optimal** $\boldsymbol{\lambda}^*$: Optimize the **(unconstrained) dual function**

$$\boldsymbol{\lambda}^* = \text{argmin}_{\lambda} g(\boldsymbol{\lambda})$$

5. Use $\boldsymbol{\lambda}^*$ to obtain $\boldsymbol{x}^*$

This is only a sketch, the theory behind it is more complicated!

Cookbook for Constraint Optimization Problems

Given: **Constraint Optimization Problem** with several constraints $g_i(\boldsymbol{x})$

$$\max_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{s.t: } g_i(\boldsymbol{x}) = 0, \quad \forall i$$

1. Write down **Lagrangian with Lagrangian Multipliers**

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_i \lambda_i \boldsymbol{g}(\boldsymbol{x})$$

2. **Solve for optimal x** for a given $\boldsymbol{\lambda}$

$$\boldsymbol{x}^* = \text{argmax}_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$$

To see the structure of the solution it is enough to use it until here…

---

3. Set back in Lagrangian to obtain **dual function**

$$g(\boldsymbol{\lambda}) = \sup_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda})$$

4. **Solve for optimal** $\boldsymbol{\lambda}^*$: Optimize the **(unconstrained) dual function**

$$\boldsymbol{\lambda}^* = \text{argmin}_{\lambda} g(\boldsymbol{\lambda})$$

5. Use $\boldsymbol{\lambda}^*$ to obtain $\boldsymbol{x}^*$

REPS: Policy Search as constraint 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 data

Scalingfactor η

- Automatically chosen from optimization (Lagrange Multiplier)
- Specified by KL-bound ϵ
- We get the exponential transformation (used by EM) for free

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

Analytic Solution

Weight computed by REPS

$$w^{[i]} \propto \exp\left(\frac{\mathcal{R}^{[i]}}{\eta}\right)$$

Getting the Lagrangian multipliers

How to get η :

Solve **dual optimization** problem:

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

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

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

Log-sum-exp softmax structure

Optimized by standard optimization tools

(e.g. trust region algorithms)

Episode-based policy evaluation strategy

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 that adapts $\boldsymbol{\theta}$ to the context $\pi(\boldsymbol{\theta}|\boldsymbol{x};\boldsymbol{\omega})$

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

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

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

Contextual Policy Search as constraint optimization

We now **optimize over the joint distribution** $p(\boldsymbol{x}, \boldsymbol{\theta}) = \mu(\boldsymbol{x})\pi(\boldsymbol{\theta}|\boldsymbol{x})$
in order to be able to use context-parameter pairs instead of many parameters for a single context

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

Maximize Reward

$$\sum_i p(\boldsymbol{x}^{[i]}, \boldsymbol{\theta}^{[i]}) = 1$$

It's a distribution

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

Stay close to the data

$$p(\boldsymbol{x}) = \sum_\theta p(\boldsymbol{x}, \boldsymbol{\theta}) = \mu_0(\boldsymbol{x})$$

Reproduce given context distribution $\boldsymbol{\mu}_0(\boldsymbol{x})$

Continuous Context?

Adding the context constraints

$\max_p \sum_i p(\boldsymbol{x}^{[i]}, \boldsymbol{\theta}^{[i]}) R(\boldsymbol{x}^{[i]}, \boldsymbol{\theta}^{[i]})$	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
$\sum_{\boldsymbol{x}} p(\boldsymbol{x})\phi(\boldsymbol{x}) = \hat{\phi}$	Reproduce given context feature averages
e.g., $\boldsymbol{\phi}(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}$ ⇒	Match Mean and Variance

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

Adding the context constraints

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

$$\sum_i \pi(\boldsymbol{\theta}^{[i]}|\boldsymbol{x}^{[i]})\boldsymbol{\mu}(\boldsymbol{x}^{[i]}) = 1$$ It's a distribution

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

Context-dependent reward baseline, linear in features

$\boldsymbol{v}$ is specified by feature constraint, Langrange Multiplier

e.g., $\phi(x) = \left[\quad x^2 \right]$ Match Mean and Variance

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

Getting the Lagrangian multipliers

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

Solve **dual optimization** problem:

Dual function:

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

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

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 $\boldsymbol{x}^{[i]}$

Policy generalization 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:

Weighting Matrix

$$\left[\begin{array}{c} \boldsymbol{k}^T \\ \boldsymbol{K}^T \end{array}\right] = (\boldsymbol{X}^T\boldsymbol{D}\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{D}\boldsymbol{A}$$

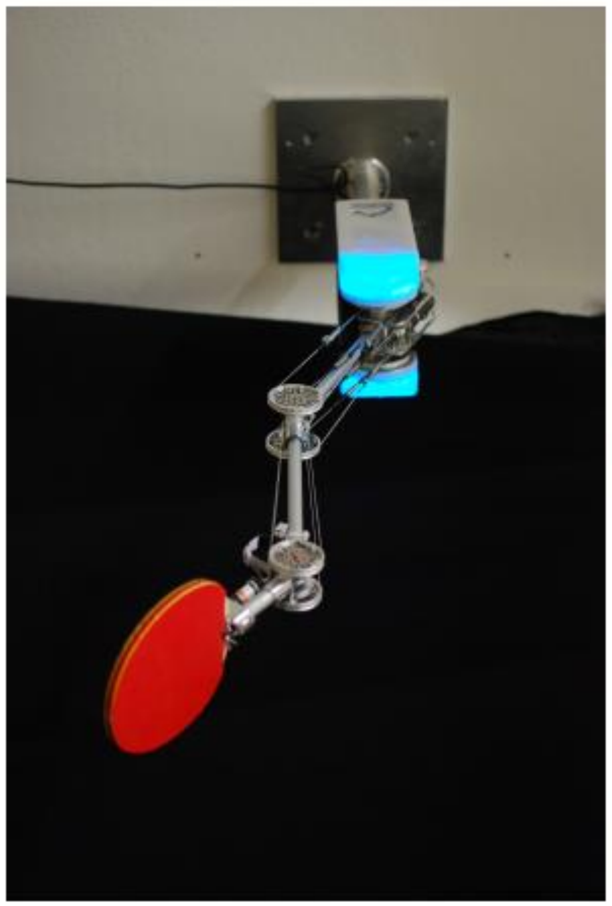
Input Matrix

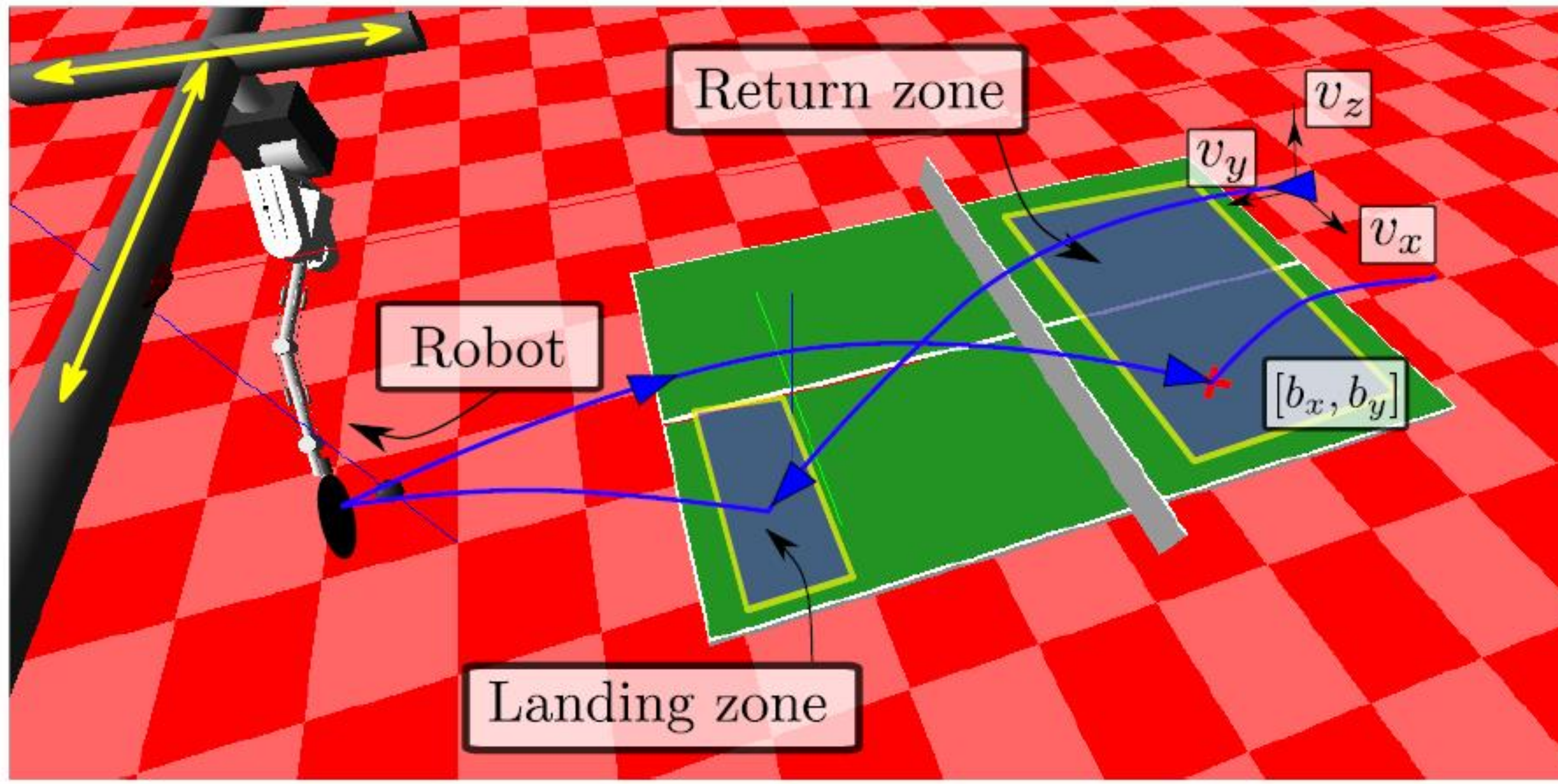
Parameter Matrix

$$\boldsymbol{\mu}_i = \boldsymbol{k} + \boldsymbol{K}\boldsymbol{x}_i \qquad \Sigma = \frac{\sum \ldots \boldsymbol{\mu_i})^T}{\sum_i p_i}$$

Just standard **weighted linear regression**…

Table tennis experiments



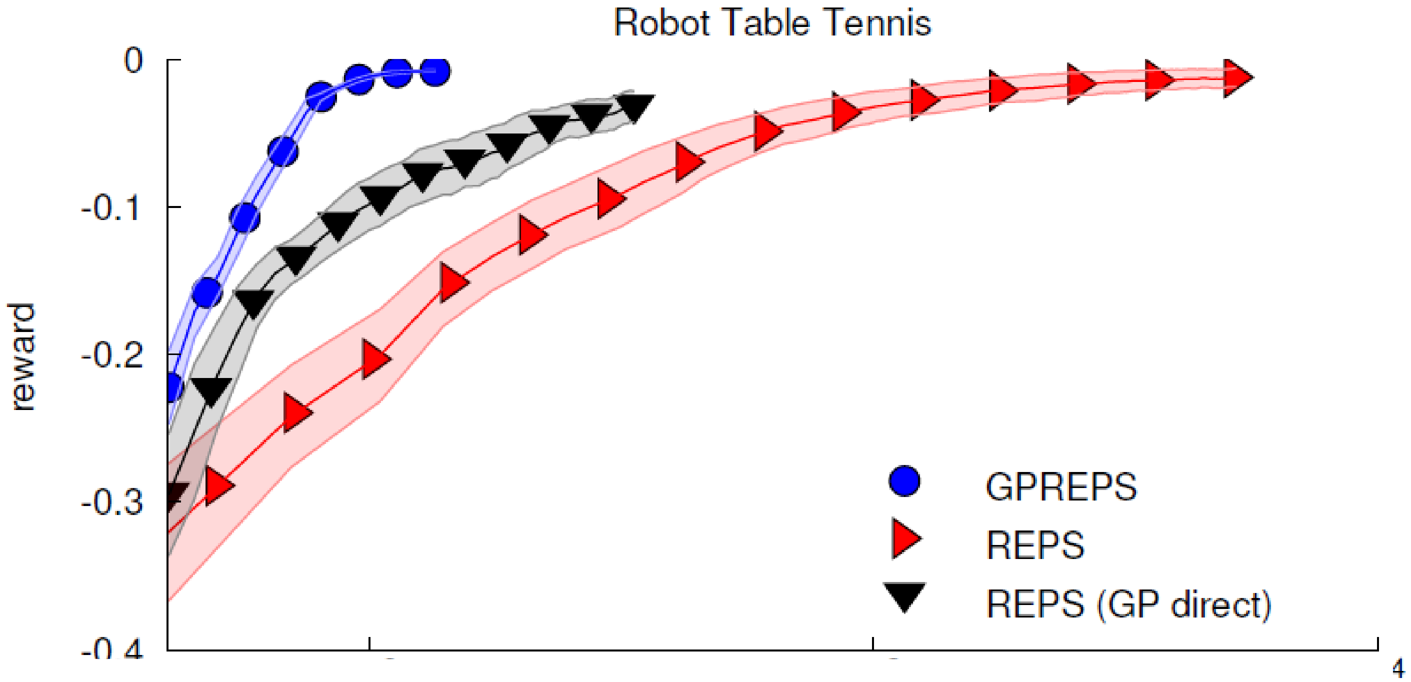


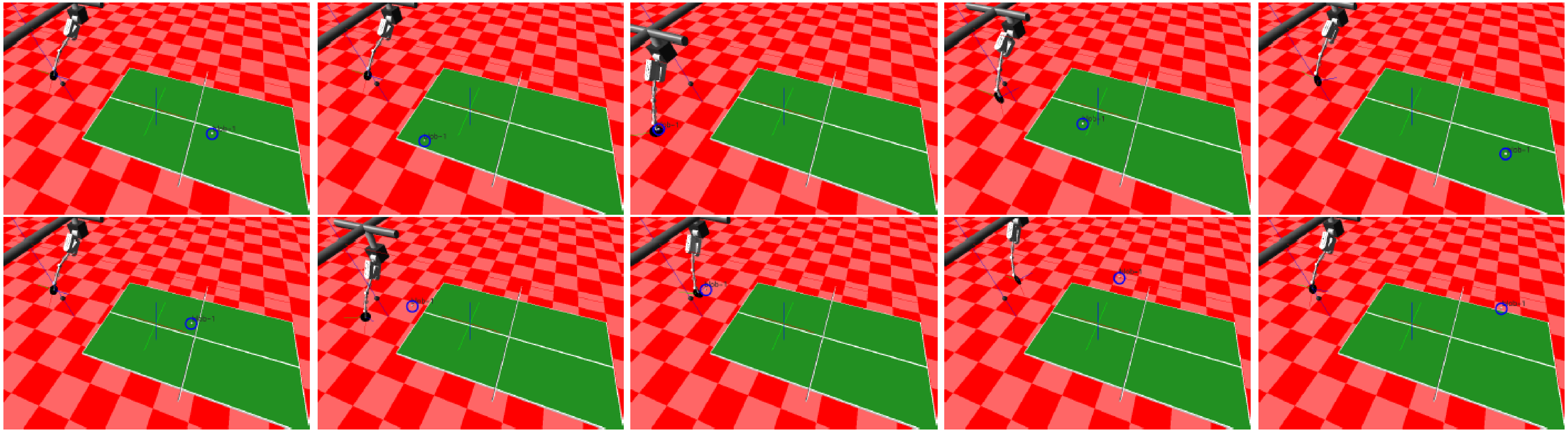
[Kupscik, Neumann et al, submitted, 2013]

Table tennis experiments

REPS with learned forward models

- Complex behavior can be learned within 100 episodes





Outline of the Lecture

1. Introduction

2. Policy Updates by Weighted Maximum Likelihood

3. Relative Entropy Policy Search (REPS)

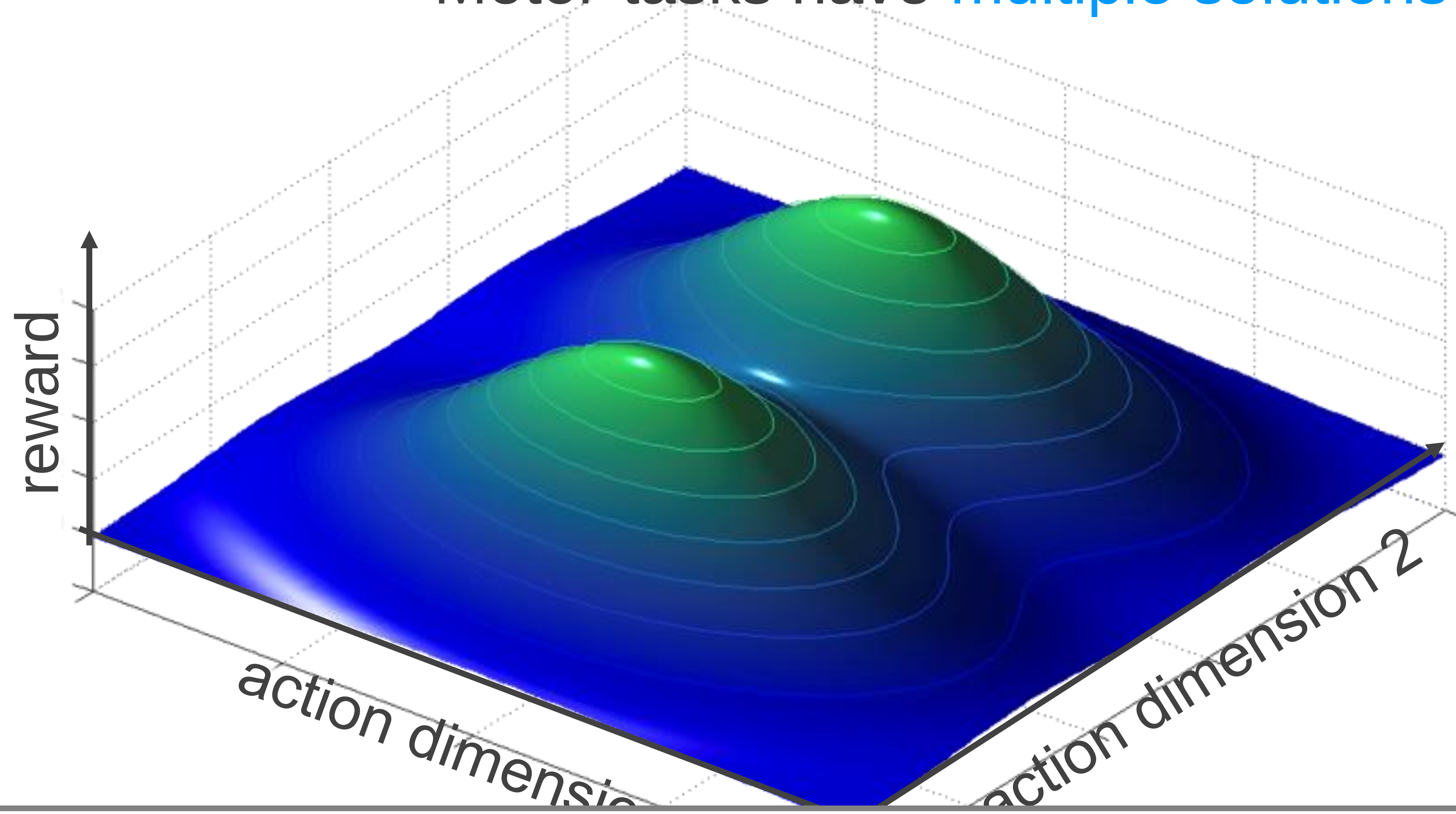
4. REPS for Contextual Policy Search

5. Learning versatile solutions

6. Conclusion

Versatile Solutions: Illustration

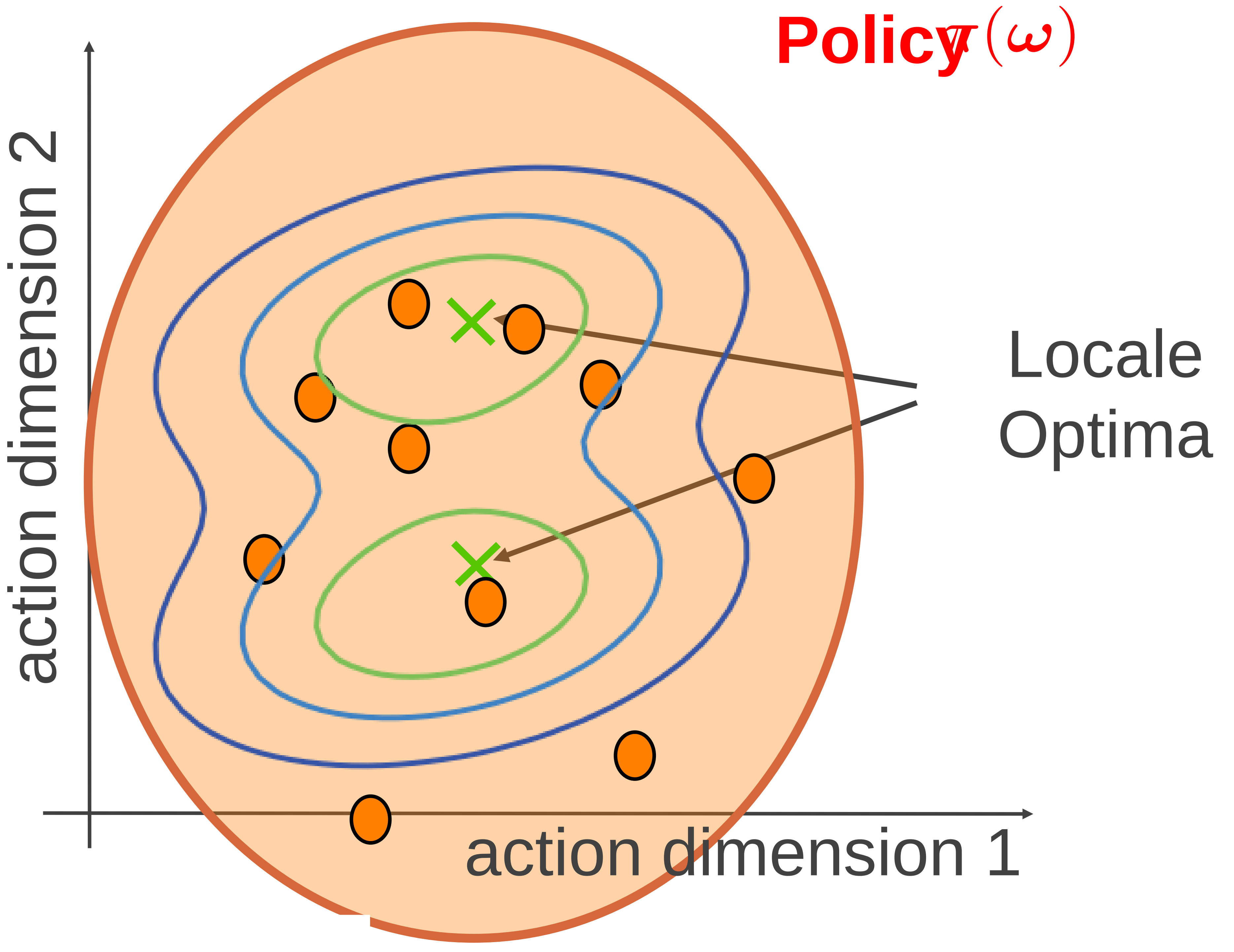
Motor-tasks have multiple solutions



C. Daniel, G. Neumann, J. Peters, *Hierarchical Relative Entropy Policy Search*, AISTATS 2012

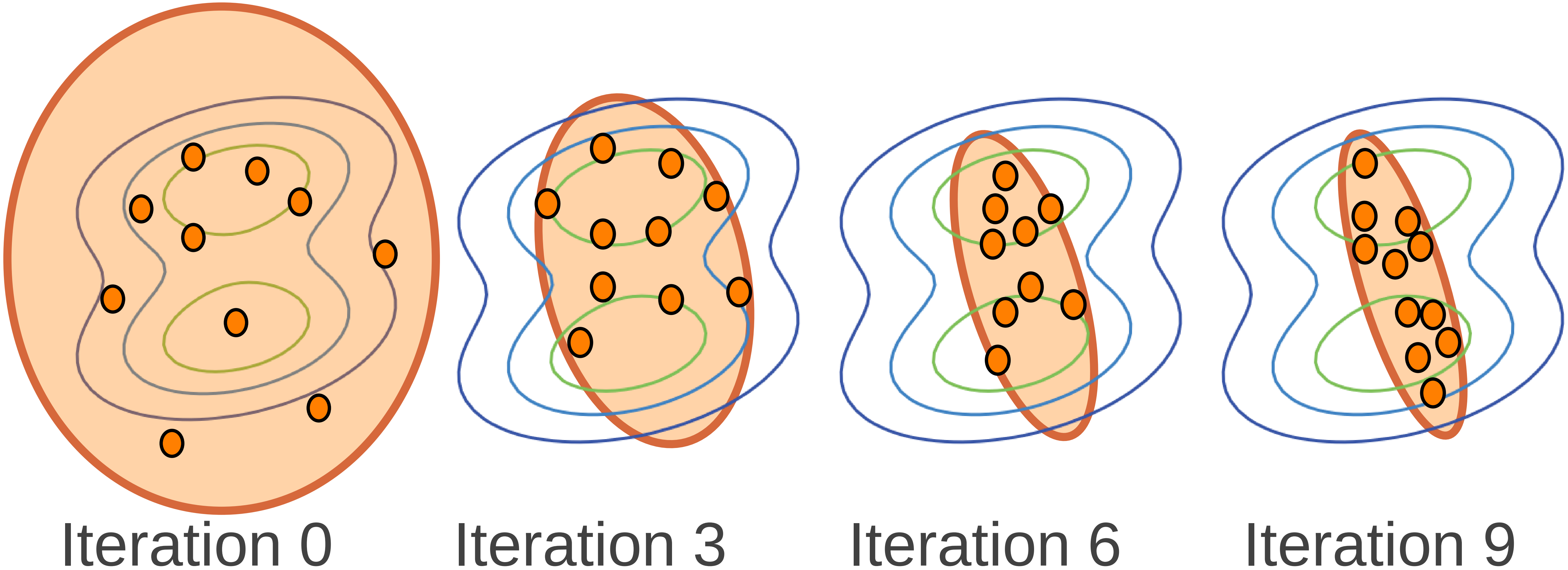
C. Daniel, G. Neumann, J. Peters, *Learning Concurrent Motor Skills in Versatile Solution Spaces*, IROS 2012, Best Cognitive Systems Paper, Best Paper Finalist

Illustration



Illustration

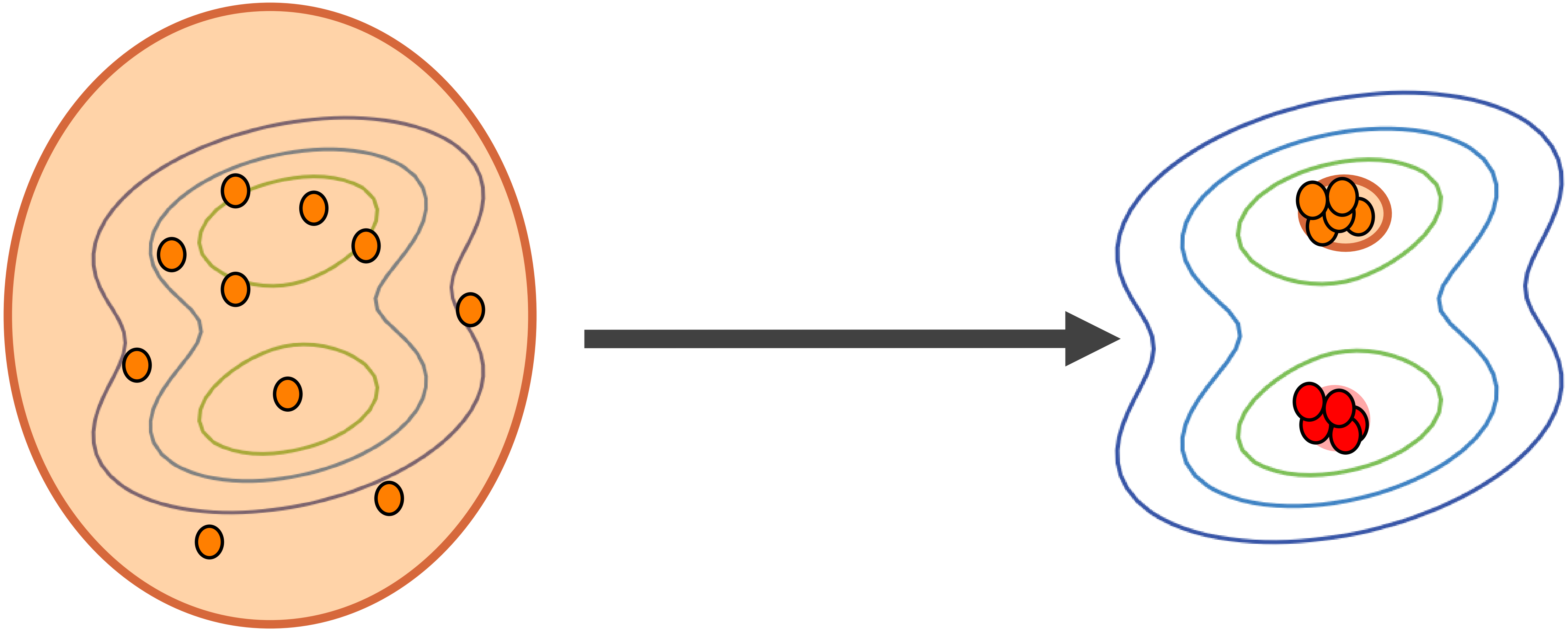
Current Formulation:



Policy averages over several modes.

Illustration

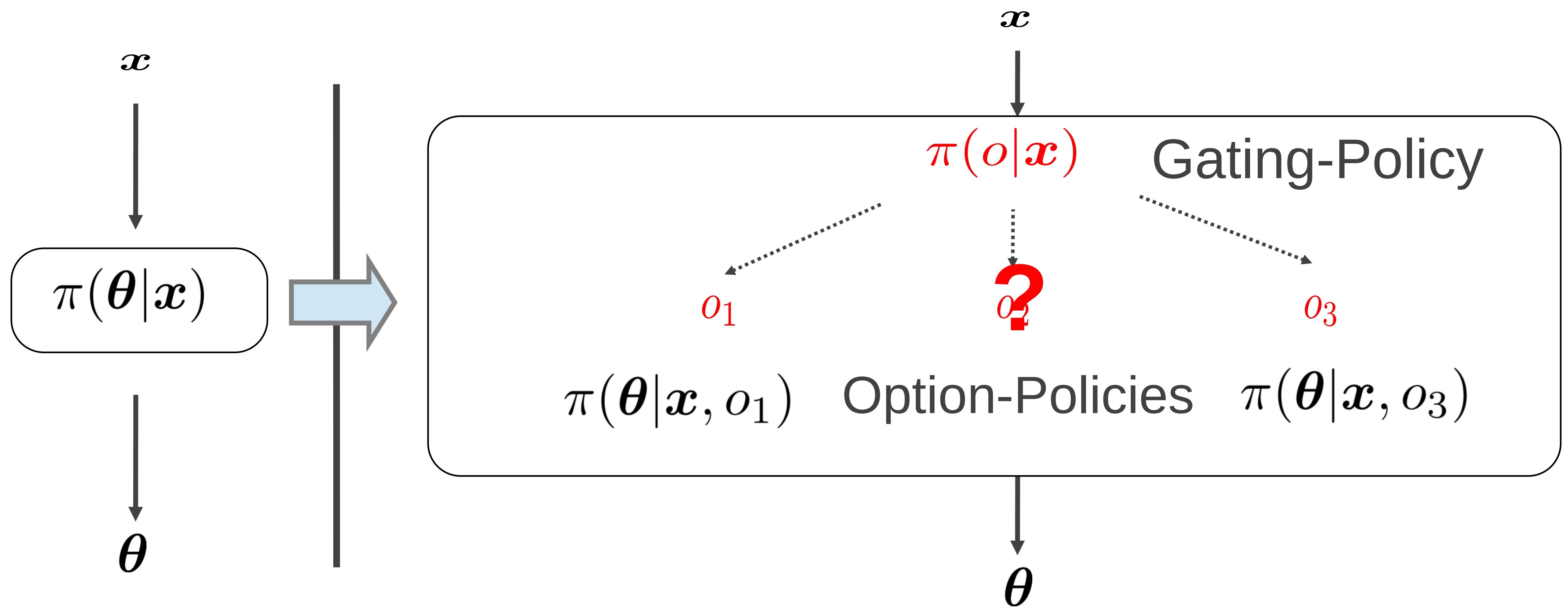
We want to find both solutions!



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

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

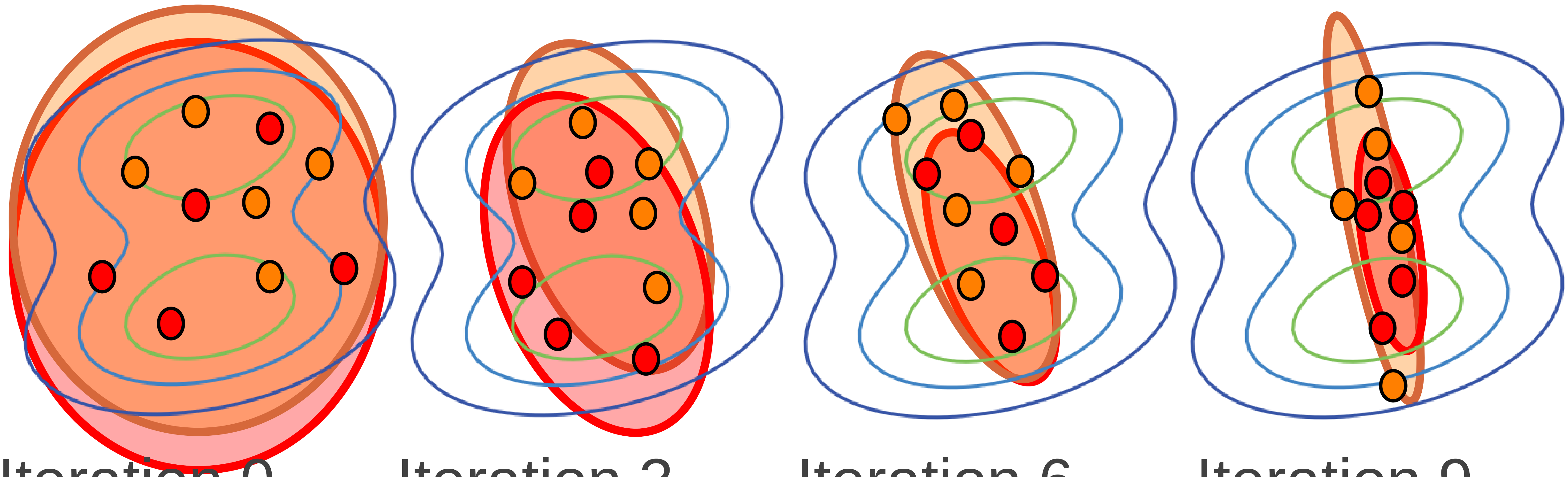
Stay close to the “data”

?

Versatile Solutions

Illustration

"Naive" Approach:

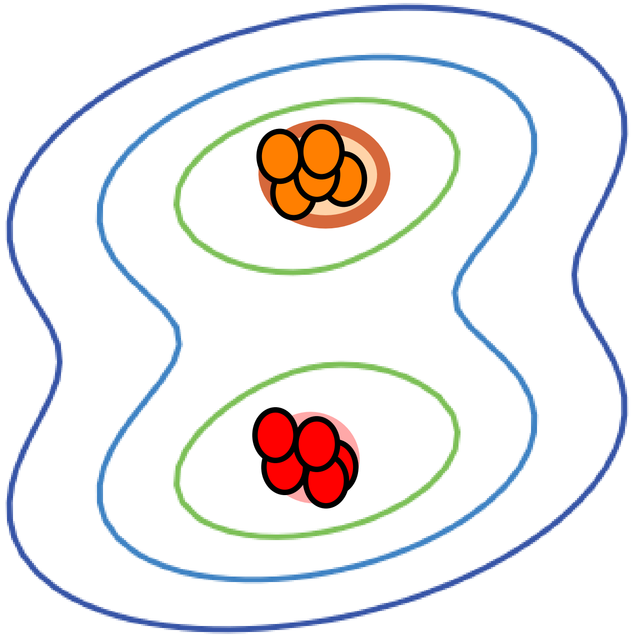


Multiple Options, BUT no separation

Learning versatile Options

Options should represent distinct solutions.

➡ Limit the overlap of the options



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

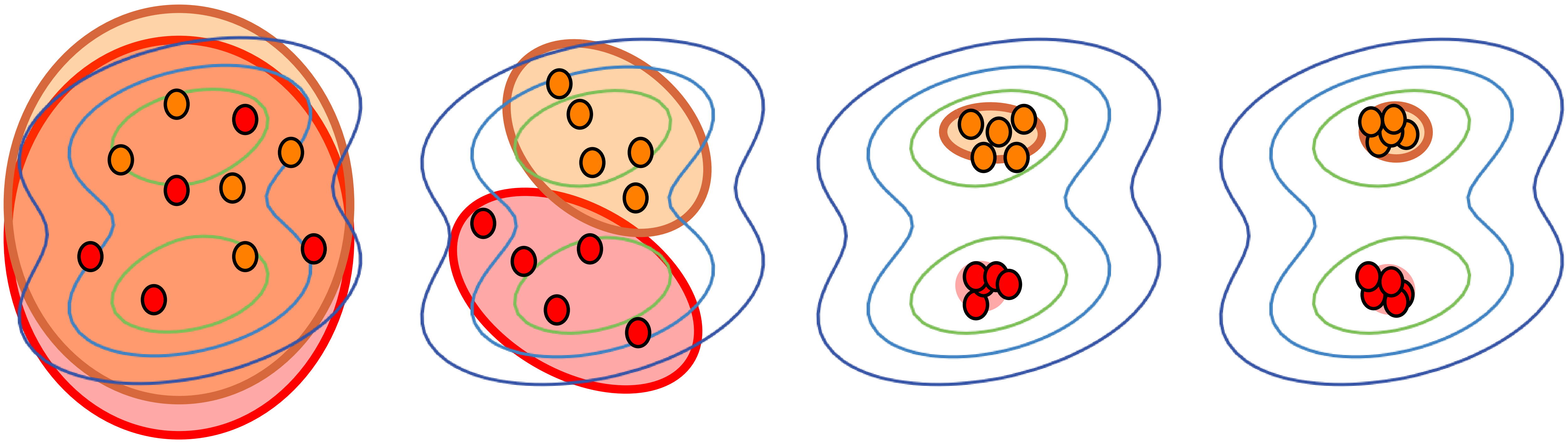
$$\epsilon \geq \sum_{\boldsymbol{x},\boldsymbol{\omega},o} \mu(\boldsymbol{x})\pi(\boldsymbol{\omega},o|\boldsymbol{x}) \log \frac{\mu(\boldsymbol{x})\pi(\boldsymbol{\omega},o|\boldsymbol{x})}{q(\boldsymbol{x},\boldsymbol{\omega},o)}$$

Stay close to the “data”, no wild exploration

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

Versatile Solutions

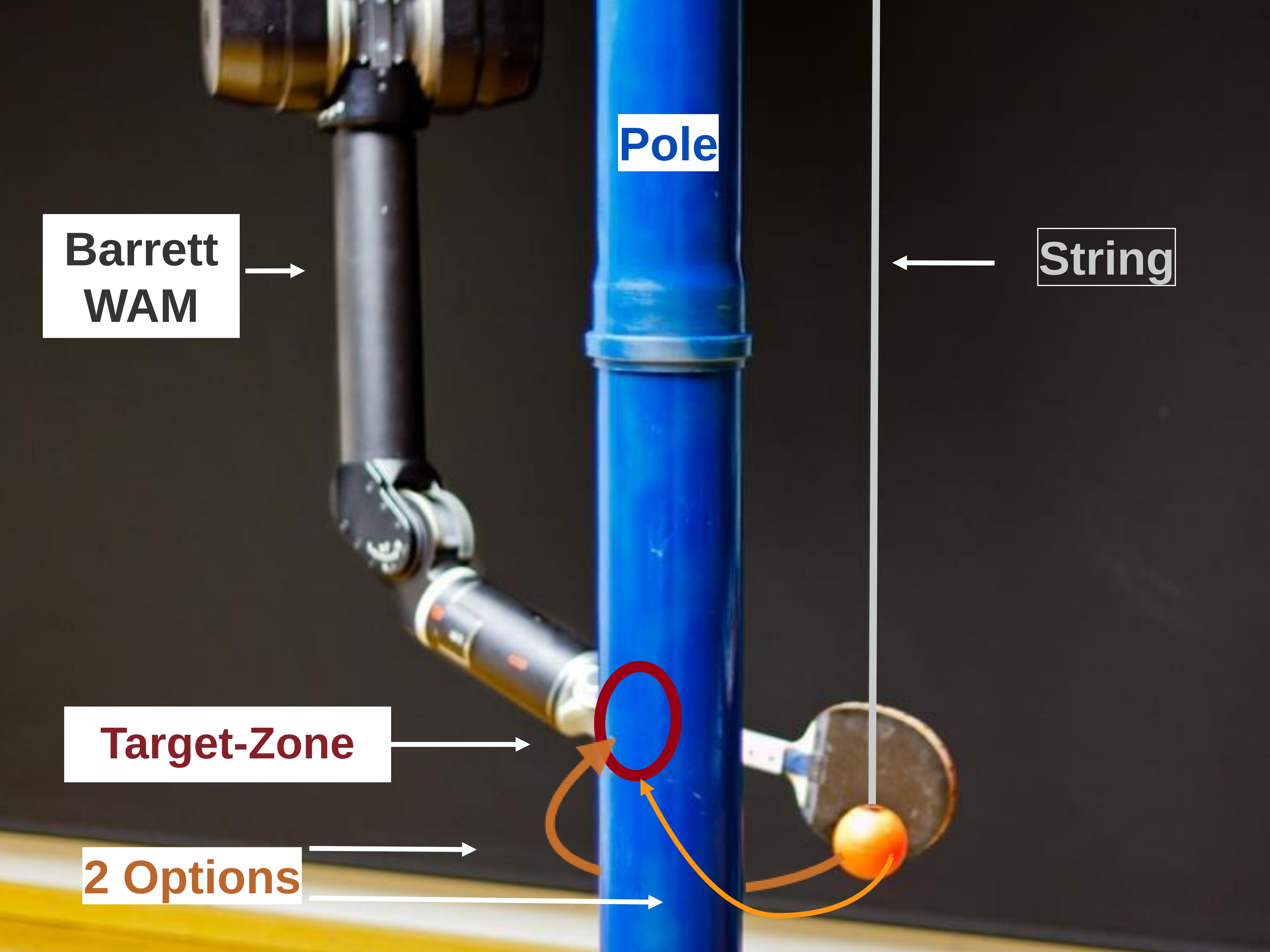
HiREPS



Iteration 0 Iteration 3 Iteration 6 Iteration 9

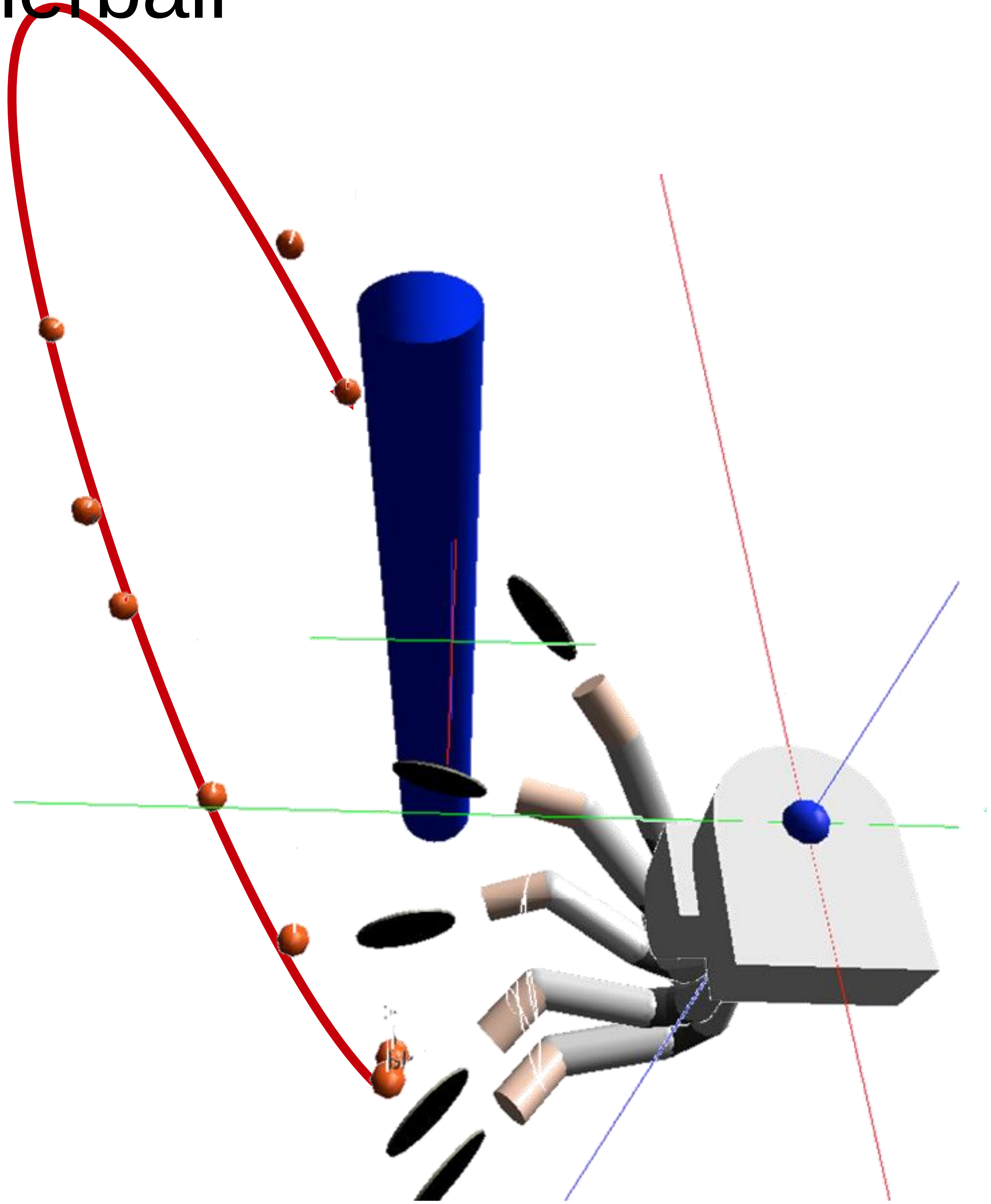
Learning of versatile, distinct solutions due to separation of options.

Tetherball

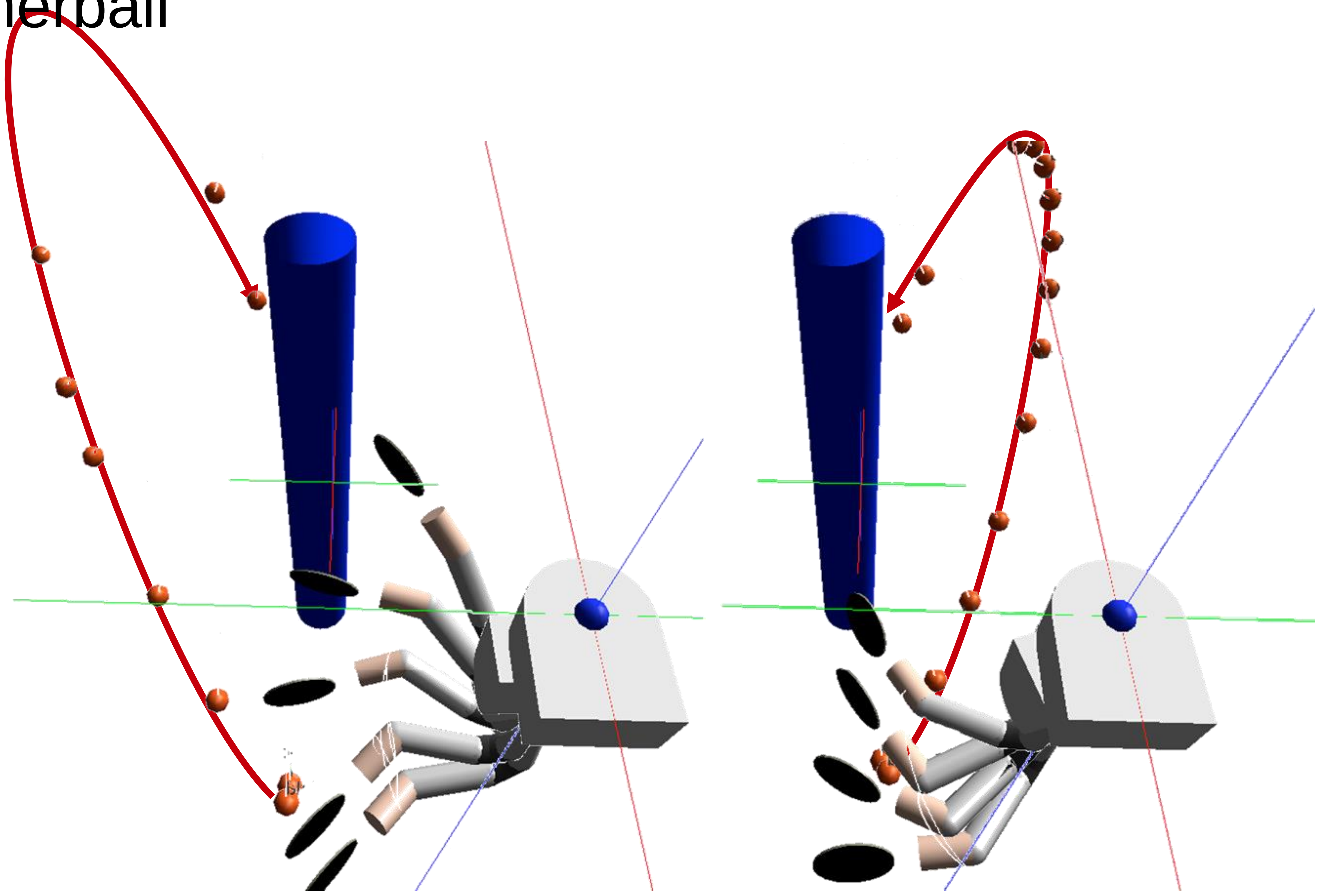


Pole
Barrett WAM
String
Target-Zone
2 Options

Tetherball



Tetherball



HiREPS learns distinct solutions.

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 transformatio**n of the returns
- REPS **optimizes the temperature** of this transformation to match a desired Kullback-Leibler divergence
- **Contextual policy search** is a powerful tool for multi-task learning