Policy Search: Methods and Applications

Jan Peters
Gerhard Neumann

Horizon 2020

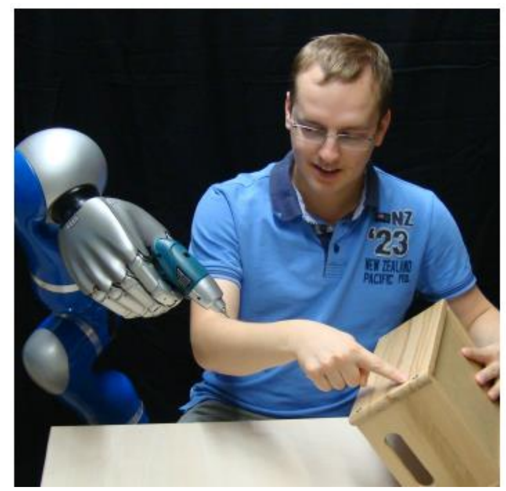
Motivation

In the next few years, we will see a dramatic increase of robot applications

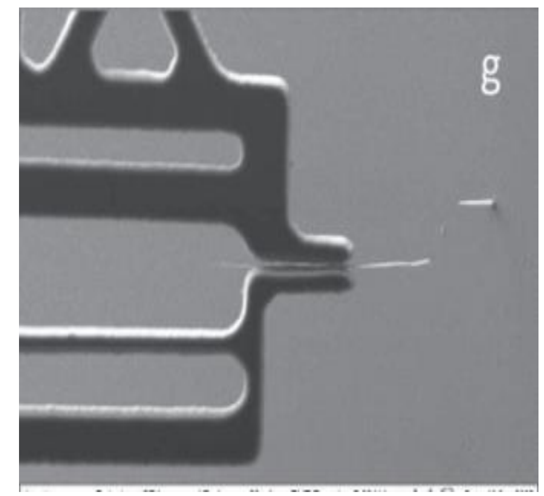
Today:

Industrial Robots

Tomorrow:



Robot Assistants



Nano-Robots

Dangerous Env.

http://www.Wikipedia.de

Household

http://news.softpedia.com/

Household

Robot Athletes

http://zackkanter.com/

Transportation

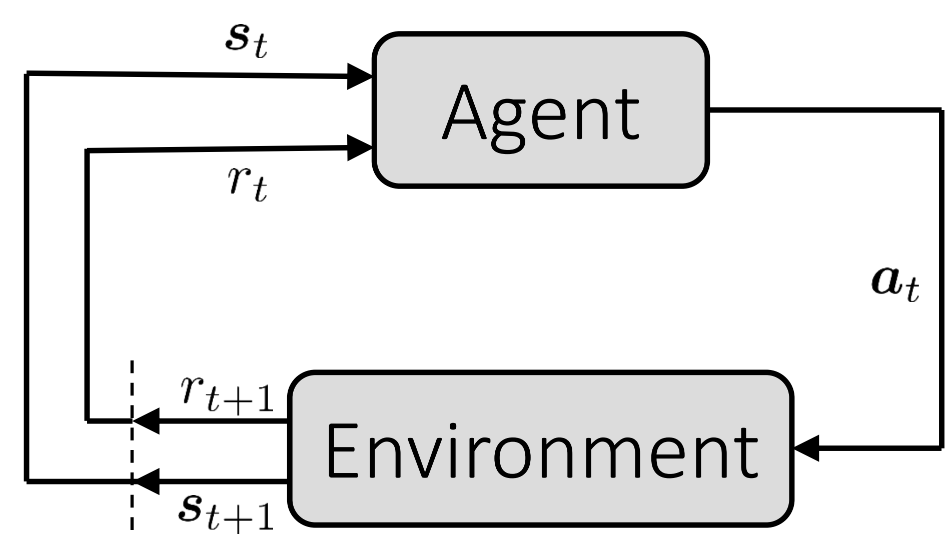
Reinforcement Learning

Most of these tasks can not be programmed by hand

Easier: Specifying a reward function ⇨ Markov Decision Processes

A Markov Decision Process (MDP) is defined by:

- its state space $\boldsymbol{s} \in \mathcal{S}$
- its action space $\boldsymbol{a} \in \mathcal{A}$
- its transition dynamics $\mathcal{P}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$
- its reward function $r(\boldsymbol{s}, \boldsymbol{a})$
- and its initial state probabilities $\mu_0(\boldsymbol{s})$



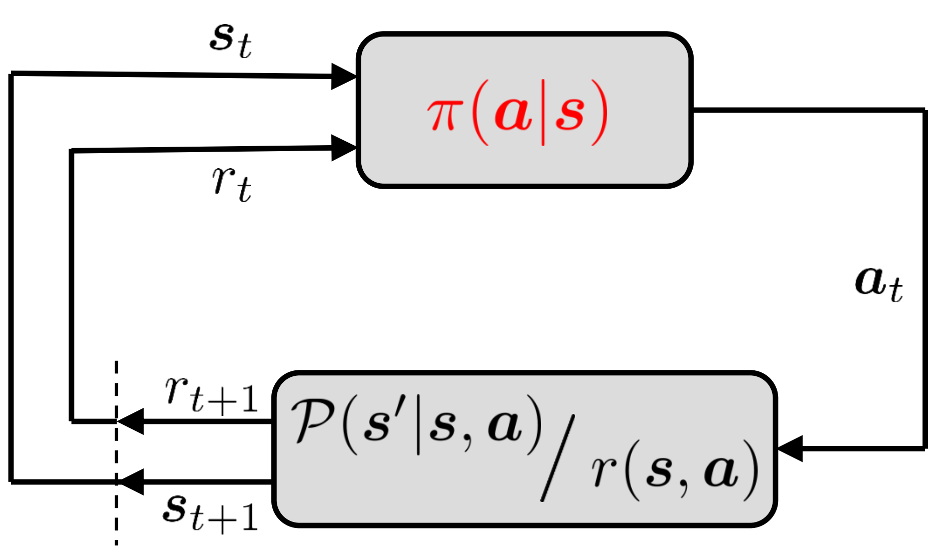
Reinforcement Learning

Most of these tasks can not be programmed by hand

Easier: Specifying a reward function ⇨ Markov Decision Processes

A Markov Decision Process (MDP) is defined by:

- its state space $\boldsymbol{s} \in \mathcal{S}$
- its action space $\boldsymbol{a} \in \mathcal{A}$
- its transition dynamics $\mathcal{P}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$
- its reward function $r(\boldsymbol{s}, \boldsymbol{a})$
- and its initial state probabilities $\mu_0(\boldsymbol{s})$



Learning: Adapting the policy $\pi(\boldsymbol{a}|\boldsymbol{s})$ of the agent

Reinforcement Learning

Objective: Find policy that maximizes long term reward J_π

$$\pi^* = \arg\max_\pi J_\pi$$

Infinite Horizon MDP:

$$J_\pi = \mathbb{E}_{\mu_0, \mathcal{P}, \pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

Tasks:

- Stabilizing movements:
  Balancing, Pendulum Swing-up…
- Rhythmic movements:
  Locomotion [Levine & Koltun., ICML 2014], Ball Padding [Kober et al, 2011], Juggling [Schaal et al., 1994]

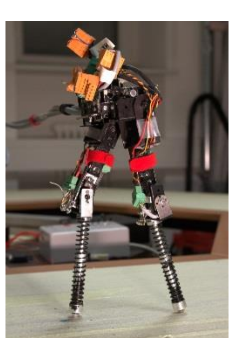
Finite Horizon MDP:

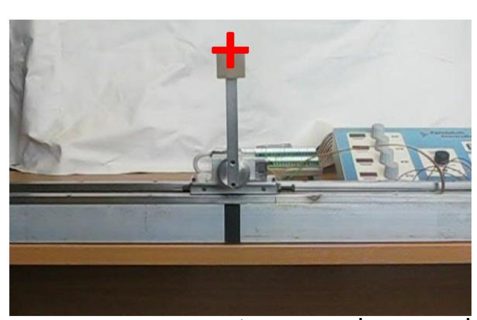
$$J_\pi = \mathbb{E}_{\mu_0, \mathcal{P}, \pi}\left[\sum_{t=0}^{T} r_t\right]$$

Tasks:

- Stroke-based movements:
  Table-tennis [Mülling et al., IJRR 2013], Ball-in-a-Cup [Kober & Peters., NIPS 2008], Pan-Flipping [Kormushev et al., IROS 2010], Object Manipulation [Krömer et al, ICRA 2015]


Stanford


Peters et. al.


Deisenroth et. al.



Peters et. al.


Kormushev et. al.

Robot Reinforcement Learning

Challenges:

Dimensionality:

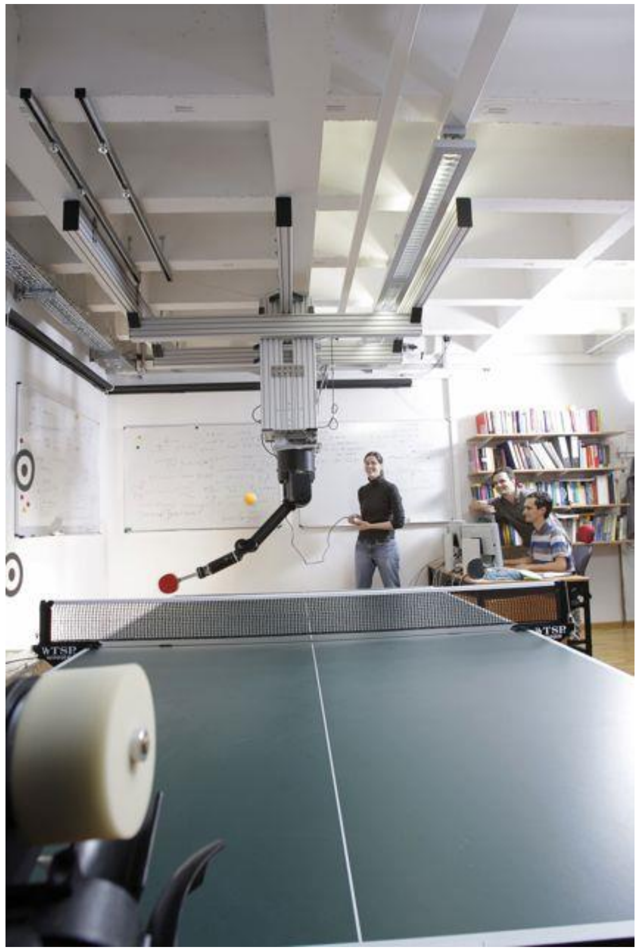
- High-dimensional continuous state and action space
- Huge variety of tasks

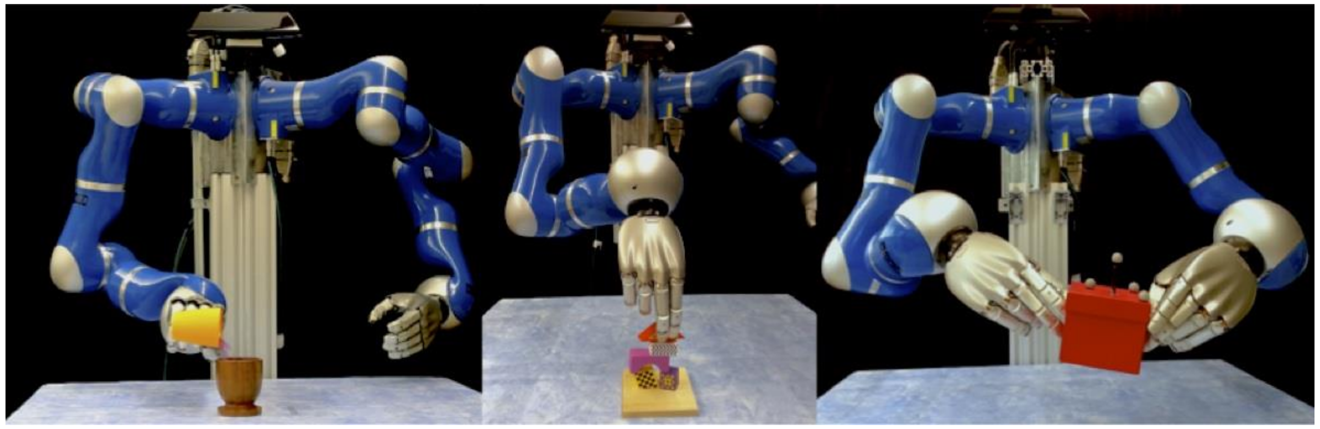
Real world environments:

- High-costs of generating data
- Noisy measurements

Exploration:

- Do not damage the robot
- Need to generate smooth trajectories





Robot Reinforcement Learning

Challenges:

- Dimensionality
- Real world environments
- Exploration

Value-based Reinforcement Learning:

Estimate value function:

e.g.: $Q(\boldsymbol{s},\boldsymbol{a}) = r(\boldsymbol{s},\boldsymbol{a}) + \gamma \mathbb{E}_{\mathcal{P}}\left[V(\boldsymbol{s}')|\boldsymbol{s},\boldsymbol{a}\right]$

- Global estimate for all reachable states
- Hard to scale to high-D
- Approximations might „destroy“ policy

Estimate global policy:

e.g.: $\pi^*(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}} Q(\boldsymbol{s},\boldsymbol{a})$

- Greedy policy update for all states
- Policy update might get unstable

Explore the whole state space:

e.g.: $\pi(\boldsymbol{a}|\boldsymbol{s}) = \dfrac{\exp(Q(\boldsymbol{s},\boldsymbol{a}))}{\sum_{\boldsymbol{a}'}\exp\left(Q(\boldsymbol{s},\boldsymbol{a}')\right)}$

- Uncorrelated exploration in each step
- Might damage the robot

Robot Reinforcement Learning

Challenges:

- Dimensionality
- Real world environments
- Exploration

Value-based Reinforcement Learning:

- Estimate value function
- Estimate global policy
- Explore the whole state space

Policy Search Methods [Deisenroth, Neumann &Peters, A Survey of Policy Search for Robotics, FNT 2013]

Use parametrized policy

$$\boldsymbol{a} \sim \pi(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{\theta}),\ \boldsymbol{\theta} \ldots \text{ parameter vector}$$

- Compact parametrizations for high-D exists
- Encode prior knowledge

Locally optimal solutions

$$\text{e.g.: } \boldsymbol{\theta}_{\text{new}} = \boldsymbol{\theta}_{\text{old}} + \alpha \frac{dJ_{\boldsymbol{\theta}}}{d\boldsymbol{\theta}}$$

- Safe policy updates
- No global value function estimation

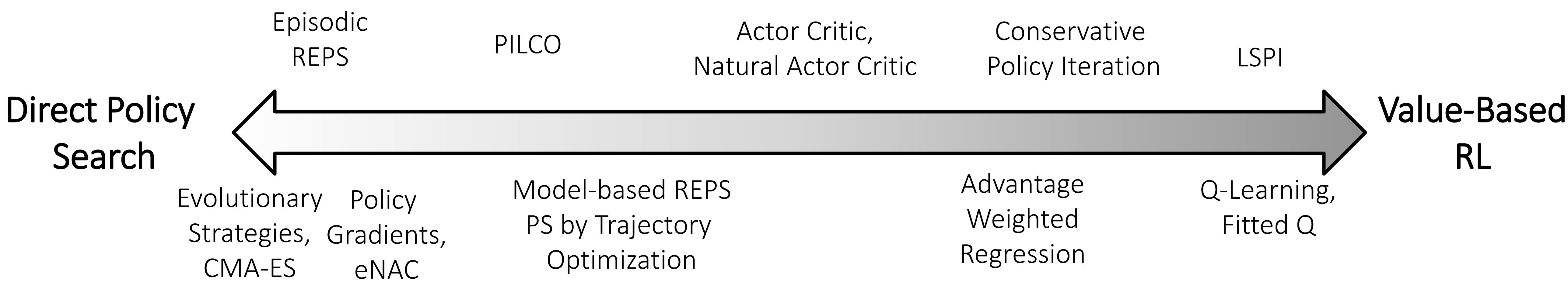
Correlated local exploration

$$\text{e.g.: } \boldsymbol{\theta}_i \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_\theta, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$$

- Explore in parameter space
- Generates smooth trajectories

Policy Search Classification

Yet, it's a grey zone…



Important Extensions:

- Contextual Policy Search [Kupscik, Deisenroth, Peters & Neumann, AAAI 2013], [Silva, Konidaris & Barto, ICML 2012], [Kober & Peters, IJCAI 2011], [Paresi & Peters et al., IROS 2015]
- Hierarchical Policy Search [Daniel, Neumann & Peters., AISTATS 2012], [Wingate et al., IJCAI 2011], [Ghavamzadeh & Mahedevan, ICML 2003]

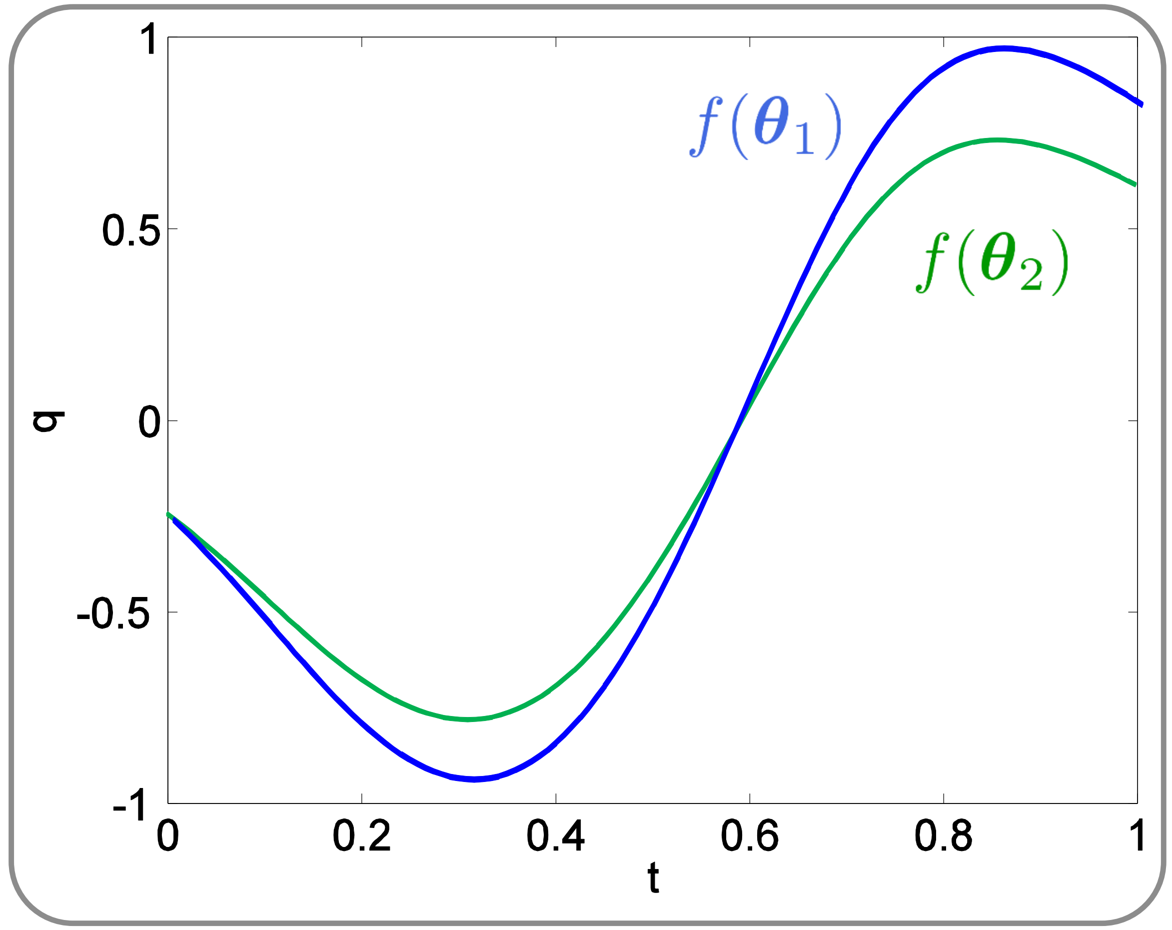
Used policy representations

Parametrized Trajectory Generators

- Returns a desired trajectory $\boldsymbol{\tau}^*$

  $$\boldsymbol{\tau}^* = \boldsymbol{q}^*_{1:T} = f(\boldsymbol{\theta})$$

- Compute controls $\boldsymbol{u}_t$ by the use of trajectory tracking controllers
- Compact representation for high-D state spaces
- Can only represent local solutions



Examples:

- Splines, Bezier Curves [Kohl & Stone., ICRA 2004], ...
- Movement Primitives [Peters & Schaal, IROS 2006], [Kober & Peters., NIPS 2008], [Kormushev et al., IROS 2010], [Kober & Peters, IJCA 2011] [Theodorou, Buchli & Schaal., JMLR 2010]

Other Representations:

- Linear Controllers [Williams et. al., 1992]
- RBF-Networks [Deisenroth & Rasmussen., ICML 2011]
- (Deep) Neural Networks [Levine & Koltun., ICML 2014][Levine & Abbeel, NIPS 2014, ICRA 2015]

Outline

Taxonomy of Policy Search Algorithms

Model-Free Policy Search Methods

- Policy Gradients
  - Likelihood Gradients: REINFORCE [Williams, 1992], PGPE [Rückstiess et al, 2009]
  - Natural Gradients: episodic Natural Actor Critic (eNAC), [Peters & Schaal, 2006]
- Weighted Maximum Likelihood Approaches
  - Success-Matching Principle [Kober & Peters, 2006]
  - Information Theoretic Methods [Daniel, Neumann & Peters, 2012]
- Extensions: Contextual and Hierarchical Policy Search

Model-Based Policy Search Methods

- Greedy Updates: PILCO [Deisenroth & Rasmussen, 2011]
- Bounded Updates: Model-Based REPS [Peters at al., 2010], Guided Policy Search by Trajectory Optimization [Levine & Koltun, 2010]

Taxonomy of Policy Search Algorithms

model-free vs. model-based

Model-Free Policy Search

Use samples

$$\mathcal{D} = \left\{ \left(\boldsymbol{s}_{1:T}^{[i]}, \boldsymbol{a}_{1:T-1}^{[i]}, r_{1:T}^{[i]} \right) \right\}$$

to directly update the policy

Properties:

- No model approximations required
- Applicable in many situations
- Requires a lot of samples

Model-Based Policy Search

Use samples

$$\mathcal{D} = \left\{ \left(\boldsymbol{s}_{1:T}^{[i]}, \boldsymbol{a}_{1:T-1}^{[i]} \right) \right\}$$

to estimate a model

Properties:

- Sample efficient
- Only works if a good model can be learned
- Optimization of inaccurate models might lead to disaster

Taxonomy of Policy Search Algorithms

model-free vs. model-based

Model-Free Policy Search

Use samples

$$\mathcal{D} = \left\{ \left(\boldsymbol{s}_{1:T}^{[i]}, \boldsymbol{a}_{1:T-1}^{[i]}, r_{1:T}^{[i]} \right) \right\}$$

to directly update the policy

Optimization methods:

- Policy Gradients [Williams et al. 992, Peters & Schaal 2006, Rückstiess et al 2008]
- Natural Gradients [Peters & Schaal 2006, Peters & Schaal 2008, Su, Wiestra & Peters 2009]
- Expectation Maximization [Kober & Peters 2008, Vlassis & Toussaint 2009]
- Information-Theoretic Policy Search [Daniel, Neumann & Peters 2012, Daniel, Neumann & Peters, 2013]
- Path Integral Control [Theoudorou, Buchli & Schaal 2010, Stulp & Sigaud 2012]
- Stochastic Search Methods [Hansen 2012, Mannor 2004]

Model-Based Policy Search

Use samples

$$\mathcal{D} = \left\{ \left(\boldsymbol{s}_{1:T}^{[i]}, \boldsymbol{a}_{1:T-1}^{[i]} \right) \right\}$$

to estimate a model

Optimization methods:

- Any model-free method with artificial samples [Kupscik , Deisenroth, Peters & Neumann, 2013]
- Analytic Policy Gradients [Deisenroth & Rasmussen 2011]
- Trajectory Optimization [Levine & Koltun 2014]

Model-free policy search

Pseudo-Algorithm: 3 basic steps

Repeat

1. **Explore:** Generate trajectories $\boldsymbol{\tau}^{[i]}$ following the current policy π_k
2. **Evaluate:** Assess quality of trajectory or actions
3. **Update:** Compute new policy π_{k+1} from trajectories and evaluations

Until convergence

Taxonomy of Model-Free Policy Search Algorithms

episode-based vs. step-based

Episode-based

Explore: in parameter space at the beginning of an episode

$$\boldsymbol{\theta}_i \sim \pi(\boldsymbol{\theta}; \boldsymbol{\omega})$$

- Learn a search distribution $\pi(\boldsymbol{\theta}; \boldsymbol{\omega})$ over the parameter space
- $\boldsymbol{\omega} \ldots$ parameter vector of search distribution
- $\boldsymbol{a} = \pi(\boldsymbol{s}; \boldsymbol{\theta})$... deterministic control policy

Evaluate: quality of parameter vectors $\boldsymbol{\theta}_i$ by the returns $R^{[i]}$

$$R^{[i]} = \sum_{t=1}^{T} r_t, \quad \mathcal{D} = \left\{ \boldsymbol{\theta}^{[i]}, R^{[i]} \right\}$$

Step-Based

Explore: in action-space at each time step

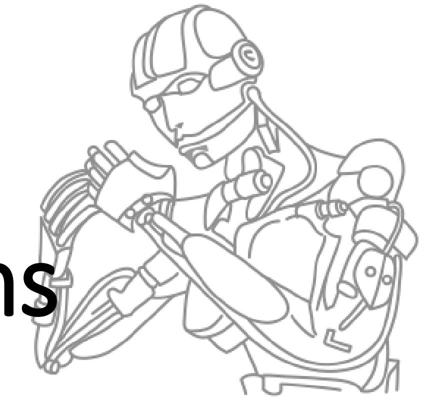
$$\boldsymbol{a}_t \sim \pi(\boldsymbol{a} | \boldsymbol{s}_t; \boldsymbol{\theta})$$

- stochastic control policy

Evaluate: quality of state-action pairs $(\boldsymbol{s}_t^{[i]}, \boldsymbol{a}_t^{[i]})$ by reward to come

$$Q_t^{[i]} = \sum_{h=t}^{T} r_h, \quad \mathcal{D} = \left\{ \boldsymbol{s}_t^{[i]}, \boldsymbol{a}_t^{[i]}, Q_t^{[i]} \right\}$$

Taxonomy of Model-Free Policy Search Algorithms



episode-based vs. step-based

Episode-based

Explore: in parameter space at the beginning of an episode

Evaluate: quality of parameter vectors $\boldsymbol{\theta}_i$ by the returns $R^{[i]}$

Properties:

- General formulation, no Markov assumption
- Correlated exploration, smooth trajectories
- Efficient for small parameter spaces (< 100)
- E.g. movement primitives

Structure-less optimization

➡„Black-Box Optimizer“

Step-Based

Explore: in action-space at each time step

Evaluate: quality of state-action pairs $(\boldsymbol{s}_t^{[i]}, \boldsymbol{a}_t^{[i]})$ by reward to come $Q_t^{[i]}$

Properties:

- Less variance in quality assessment.
- More data-efficient (in theory)
- Jerky trajectories due to exploration
- Can produce unreproducible trajectories for exploration-free policy

Use structure of the RL problem

➡decomposition in single timesteps

Taxonomy of Model-Free Policy Search Algorithms

episode-based vs. step-based

Episode-based

Explore: in parameter space at the beginning of an episode

Evaluate: quality of parameter vectors $\boldsymbol{\theta}_i$ by the returns $R^{[i]}$

Algorithms:

- Episodic REPS [Daniel, Neumann & Peters, 2012]
- PI2-CMA [Stulp & Sigaud, 2012]
- CMA-ES [Hansen et al., 2003]
- NES [Su, Wiestra, Schaul & Schmidhuber, 2009]
- PE-PG [Rückstiess, Sehnke, et al.2008]
- Cross-Entropy Search [Mannor et al. 2004]

Step-Based

Explore: in action-space at each time step

Evaluate: quality of state-action pairs $(\boldsymbol{s}_t^{[i]}, \boldsymbol{a}_t^{[i]})$ by reward to come $Q_t^{[i]}$

Algorithms:

- Reinforce [Williams 1992]
- Policy Gradient Theorem / GPOMDP [Baxter & Bartlett , 2001]
- Episodic Natural Actor Critic [Peters & Schaal, 2003]
- 2nd Order Policy Gradients [Furmston & Barber 2011]
- Deterministic Policy Gradients [Silver, Lever et al, 2014]

Taxonomy of Model-Free Policy Search Algorithms

episode-based vs. step-based

Episode-based

Explore: in paramet
beginning of an epi

Evaluate: quality of
$\boldsymbol{\theta}_i$ by the returns

Algorithms:

- Episodic REPS [CIT
- PI2-CMA [CITE]
- CMA-ES [CITE]
- NES [CITE]
- PE-PG [CITE]
- Cross-Entropy Sea

Hybrid

Explore: in parameter space at each time step

Evaluate: quality of state-action pairs $(\boldsymbol{s}_t^{[i]}, \boldsymbol{a}_t^{[i]})$ by reward to come $Q_t^{[i]}$

Properties:

- State dependent exploration
- Can be reproduced by noise-free policy

Algorithms:

- Power [Kober & Peters, 2008]
- PI2 [Theoudorou, Buchli & Schaal, 2010]

More recent versions of these algorithms are episode-based

space at each time

of state-action pairs
d to come $Q_t^{[i]}$

orem / GPOMDP [CITE]
tor Critic [CITE]
adients [CITE]

Model-Free Policy Updates

Use samples $\mathcal{D}_{\text{ep}} = \left\{\boldsymbol{\theta}^{[i]}, R^{[i]}\right\}$ or $\mathcal{D}_{\text{st}} = \left\{\boldsymbol{s}_t^{[i]}, \boldsymbol{a}_t^{[i]}, Q_t^{[i]}\right\}$

to directly update the policy

- Different optimization methods
  - Gradients: Reinforce [Williams 1992], Natural Actor Critic [Peters & Schaal, 2003][Peters & Schaal, 2006], PGPE [Rückstiess et al. 2009]
  - Success matching by weighted maximum likelihood: POWER [Kober & Peters 2008], Episodic REPS [Daniel , Neumann & Peters, 2012], Path Integrals [Theodorou, Buchli & Schaal 2010]
  - Evolutionary strategies [Hansen 2003], Cross-entropy [Mannor 2004], ...
  - Many of them can be used for step-based and episode-based policy search
- Different metrics to define the step-size of update
  - Euclidian (distance in parameter space) [Williams 1992][Rückstiess et al., 2009]
  - Relative Entropy ("distance" in probability space) [Bagnell et al. 2003], [Peters & Schaal 2006], [Peters et al. 2010], [Daniel, Neumann & Peters 2012]
  - Heuristics [Kober & Peters 2008, Theoudorou, Buchli & Schaal,2010, Hansen et al., 2003]
- Before discussion of algorithms: Analyze consequence of step size

Model-Free Policy Updates

- Reproduce trajectories with high quality / Avoid trajectories with low quality
- We learn stochastic policies:

$$\boldsymbol{\theta}_i \sim \pi(\boldsymbol{\theta}; \boldsymbol{\omega})$$

Episode-based

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

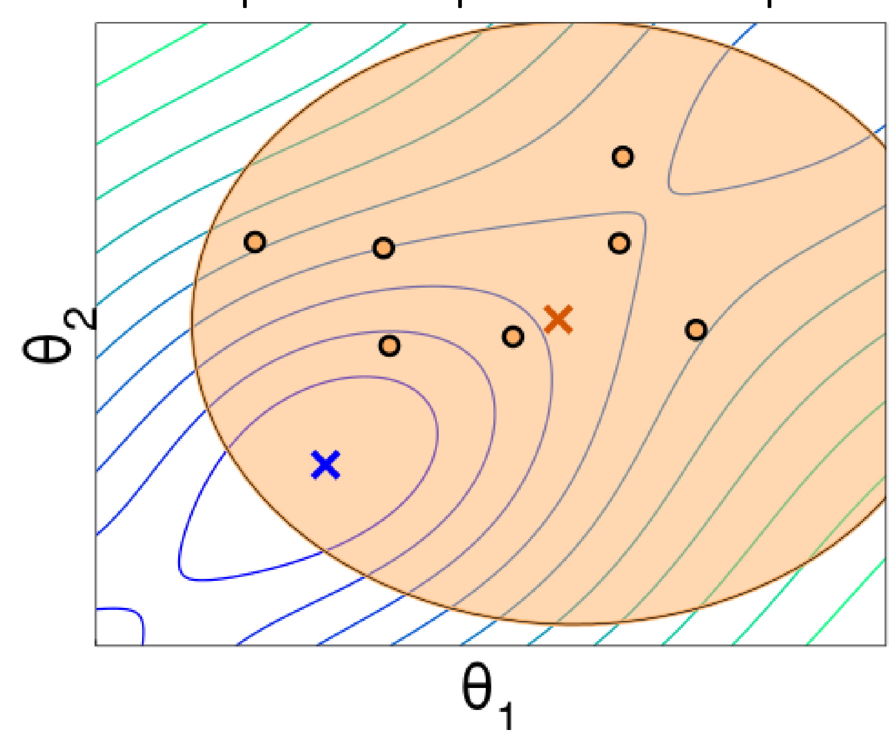
Step-based

  - Used for exploration!
- **Efficient Learning:** also update exploration rate!
- E.g. For Gaussian policies:

$$\boldsymbol{\theta}_i \sim \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

  - Update mean and covariance!
  - Mean $\boldsymbol{\mu}$: easy!
  - Covariance $\boldsymbol{\Sigma}$: hard!

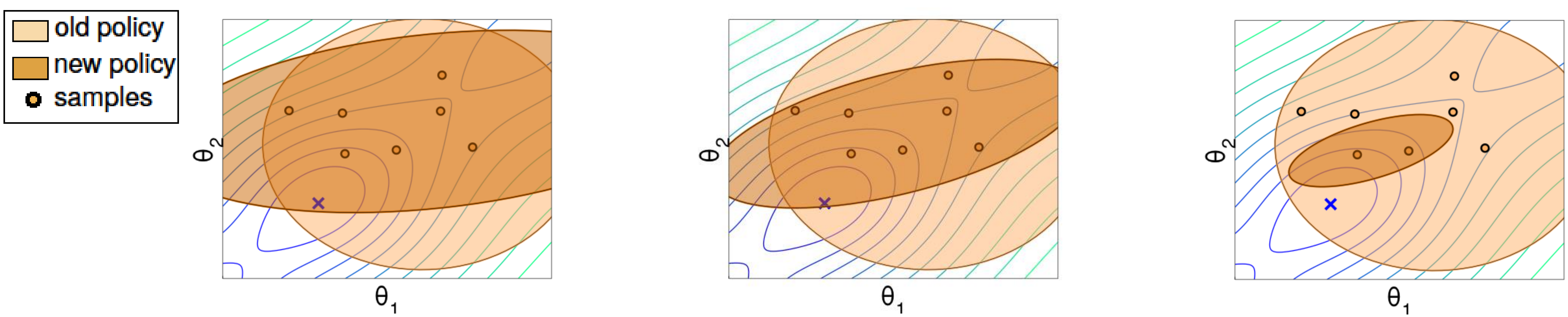
Example: 2-D parameter space



Desired Properties for the Policy Update

Desired properties:

- Invariance to parameter or reward transformations
- Regularize policy update
  - Update is computed based on data
    - ➡ **stay close to data!**
  - Smooth learning progress
- Controllable exploration-exploitation trade-off



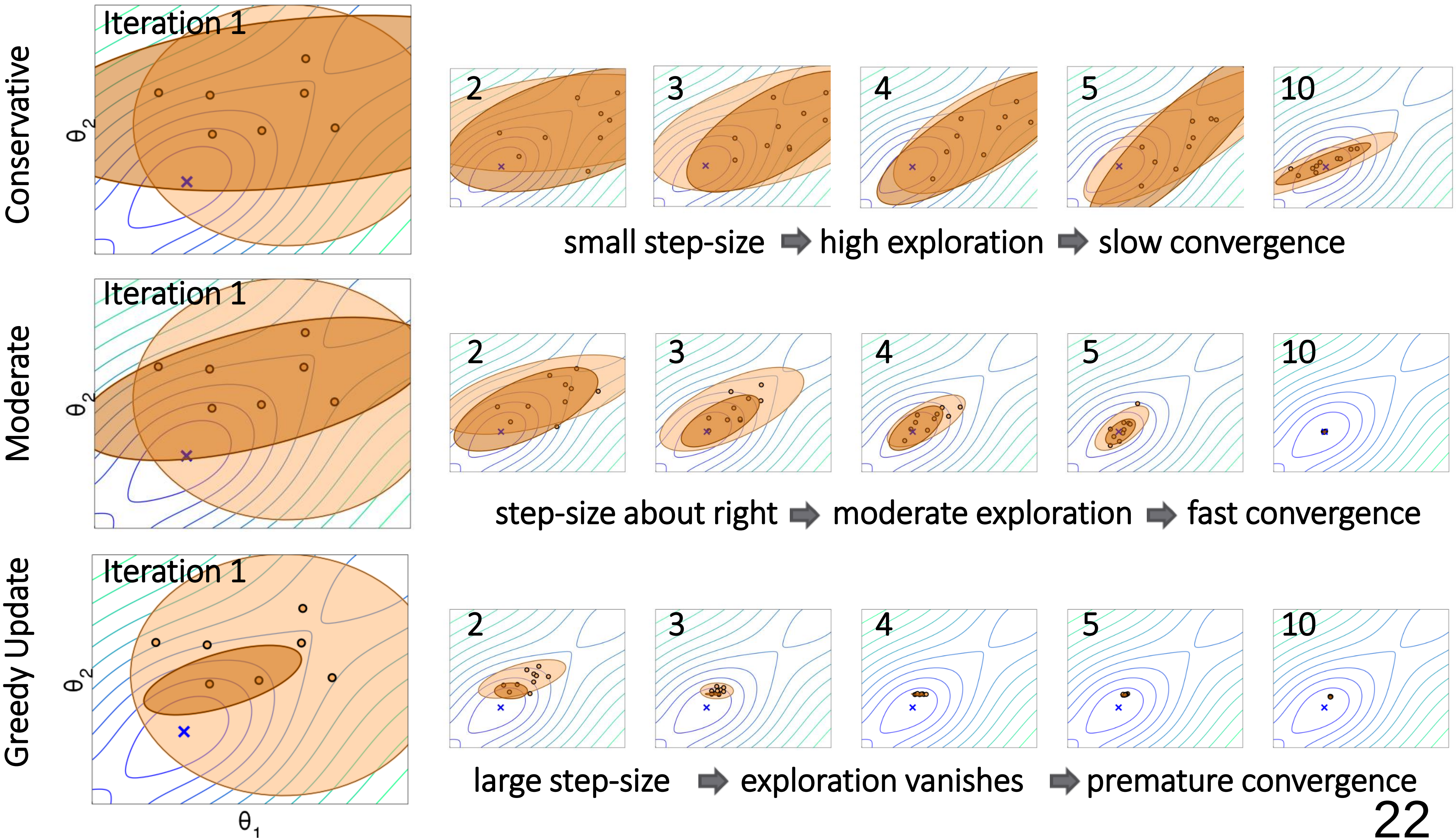
Conservative Update
Small "step size"

Moderate Update,
Moderate "step size"

Greedy update
Large "step size"

Which policy update should we use?

Illustration of Policy Updates



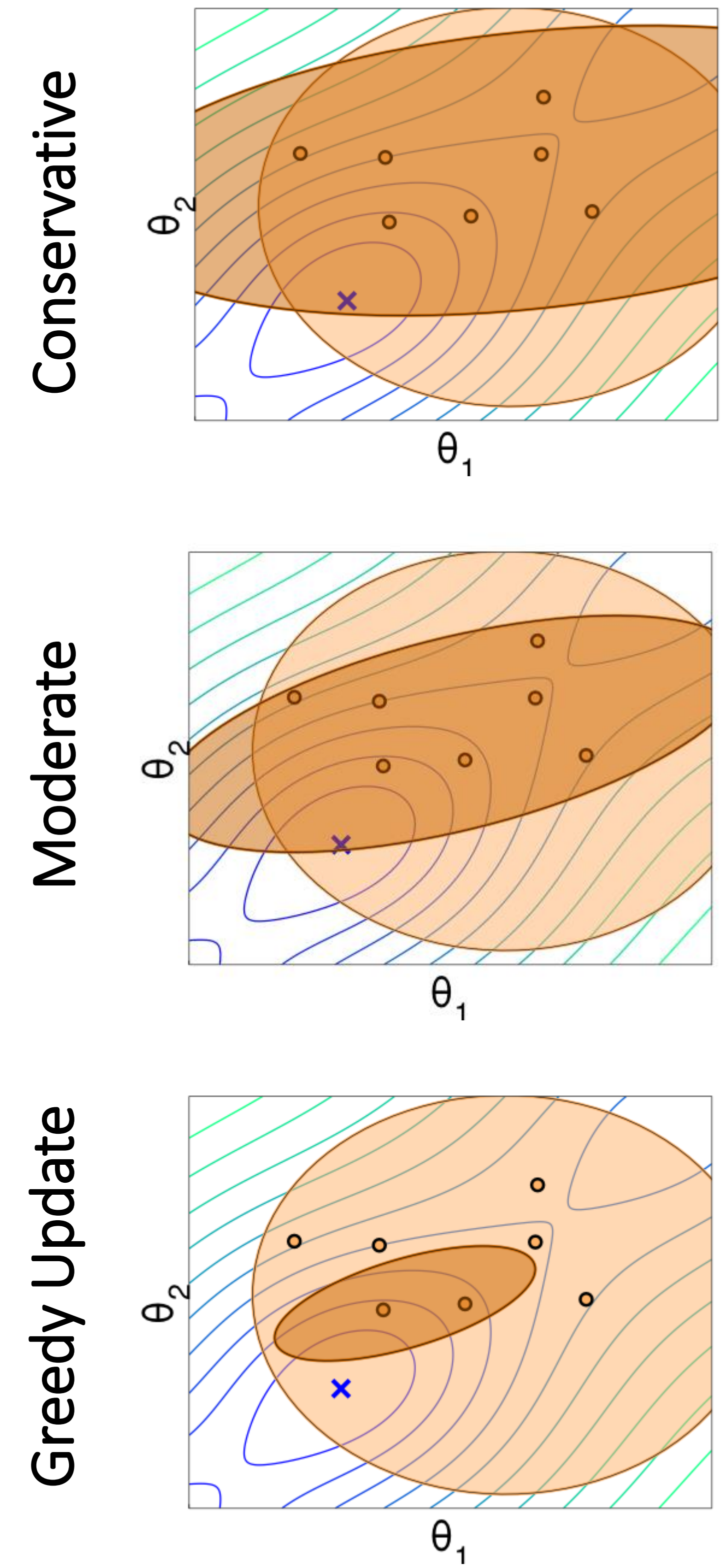
Metrics used for the Policy Update

Desired properties:

- Invariance to parameter or reward transformations
- Regularize policy update
  - Update is computed based on data
    - ➡ stay close to data
  - Smooth learning progress
- Controllable exploration-exploitation trade-off
  - Explore: Higher reward in future / Lower reward now
  - Exploit: Higher reward now / Lower reward in the future
  - Which one to choose? Do not know… problem specific
  - But: algorithm should allow us to choose the greediness

Metric used for the policy update

- Different metrics are used to define the step-size of the update
- Need metric that can measure the greediness of the update



Outline

Taxonomy of Policy Search Algorithms

Model-Free Policy Search Methods

- Policy Gradients
  - Likelihood Gradients: REINFORCE [Williams, 1992], PGPE [Rückstiess et al, 2009]
  - Natural Gradients: episodic Natural Actor Critic (eNAC), [Peters & Schaal, 2006]
- Weighted Maximum Likelihood Approaches
  - Success-Matching Principle [Kober & Peters, 2006]
  - Information Theoretic Methods [Daniel, Neumann & Peters, 2012]
- Extensions: Contextual and Hierarchical Policy Search

Model-Based Policy Search Methods

- Greedy Updates: PILCO [Deisenroth & Rasmussen, 2011]
- Bounded Updates: Model-Based REPS [Peters at al., 2010], Guided Policy Search by Trajectory Optimization [Levine & Koltun, 2010]

Policy Gradients

Optimization Method: Gradient Ascent

- Compute gradient from samples

$$\mathcal{D}_{\text{ep}} = \left\{ \boldsymbol{\theta}^{[i]}, R^{[i]} \right\} \quad \text{or} \quad \mathcal{D}_{\text{st}} = \left\{ \boldsymbol{s}_t^{[i]}, \boldsymbol{a}_t^{[i]}, Q_t^{[i]} \right\}$$

$$\partial J_{\boldsymbol{\theta}} / \partial \boldsymbol{\omega} = \nabla_{\boldsymbol{\omega}} J_{\boldsymbol{\omega}} \quad \text{or} \quad \partial J_{\boldsymbol{\theta}} / \partial \boldsymbol{\theta} = \nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}$$

- Update policy parameters in the direction of the gradient

$$\omega_{k+1} = \omega_{k+1} + \alpha \nabla_{\boldsymbol{\omega}} J_{\boldsymbol{\omega}_k} \quad \text{or} \quad \boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha \nabla_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}_k}$$

  - $\alpha \ldots$ learning rate

Likelihood Policy Gradients

Episode-Based: Policy $\boldsymbol{\theta} \sim \pi(\boldsymbol{\theta}; \boldsymbol{\omega})$

We can use the log-ratio trick to compute the policy gradient

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

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) [Rückstiess et al., 2009]

Baselines...

We can always **subtract a baseline *b*** 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 baselines:

- Average reward
- but there are optimal baselines for each algorithm that minimize the variance [Peters & Schaal, 2006], [Deisenroth, Neumann & Peters, 2013]

Step-based Policy Gradient Methods

The returns can still have a lot of variance

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

... as 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
- We have to compute the gradient $\nabla_{\boldsymbol{\theta}} J$ for the low-level policy $\pi(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{\theta})$

Step-based Policy Gradient Methods

Plug in the temporal structure of the RL problem

- Trajectory distribution:

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

- Return for a single trajectory:

$$R(\boldsymbol{\tau}) = \sum_{t=1}^{T} r_t$$

➡ Expected long term reward $J_{\boldsymbol{\theta}}$ can be written as expectation over the trajectory distribution

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

Step-Based Likelihood Ratio Gradient

Using the log-ratio trick, we arrive at

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

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

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

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

Model-dependent terms cancel due to the derivative

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

Step-Based Policy Gradients

Plug it back in...

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

$$= \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_t^{[i]} | \boldsymbol{s}_t^{[i]}; \boldsymbol{\theta}) \left(\sum_{t=1}^{T} r_t^{[i]} \right)$$

This algorithm is called the REINFORCE Policy Gradient [Williams 1992]

- Wait... we still use the returns $R(\boldsymbol{\tau})$
  - ➡ 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 [Sutton 1999]

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

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

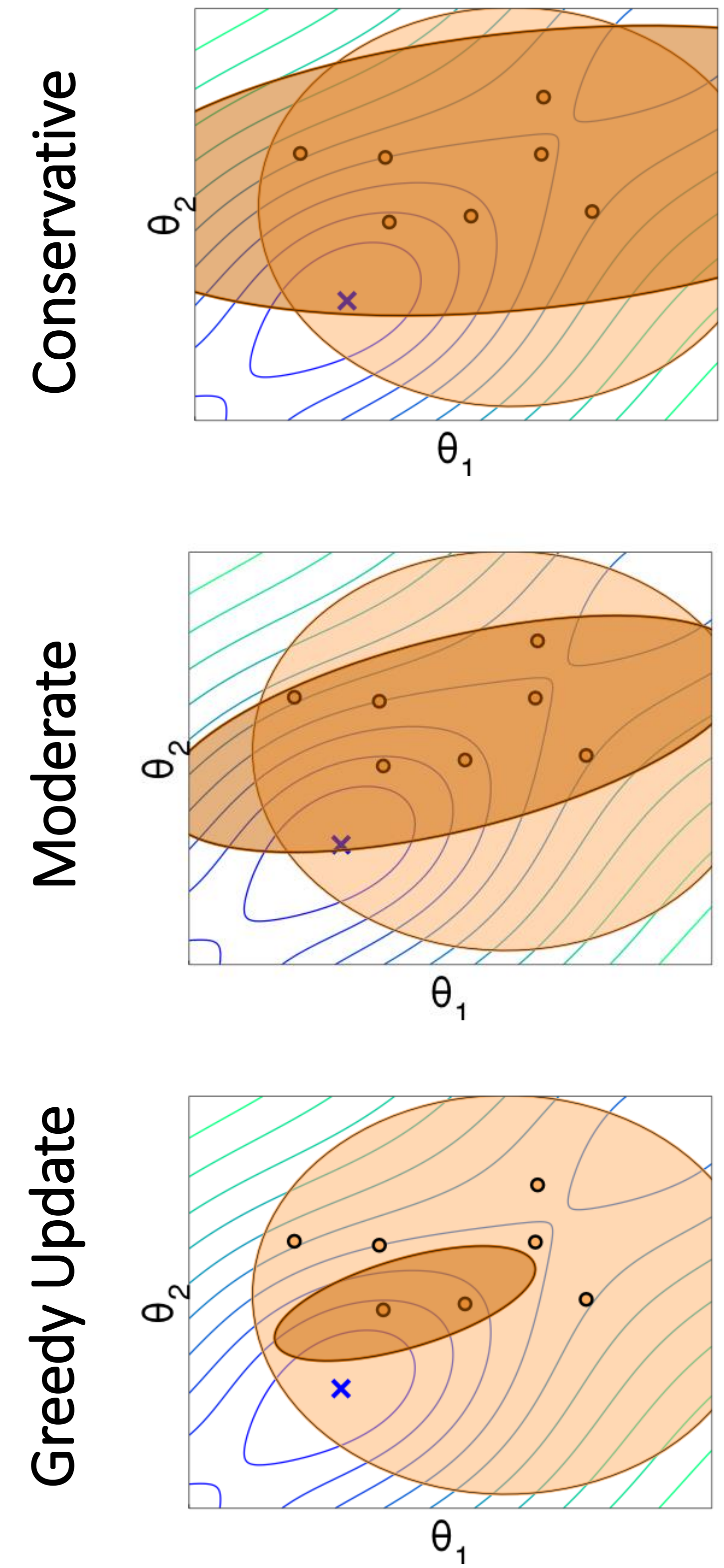
- The rewards to come have less variance
- Can also be done with a baseline…

Metric in standard gradients

Ok, how can we choose the learning rate α?

Metric used for policy gradients:

- Standard gradients use euclidian distance in parameter space as metric
- Episode-based: $L_2(\pi_{k+1}, \pi_k) = ||\boldsymbol{\omega}_{k+1} - \boldsymbol{\omega}_k||$
- Step-based: $L_2(\pi_{k+1}, \pi_k) = ||\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k||$
- Invariance to reward transformations
  - Choose learning rate, such that $L_2(\pi_{k+1}, \pi_k) \leq \epsilon$
  - Resulting learning rate: $\alpha_k = \dfrac{1}{||\nabla J||}\epsilon$
- No Invariance to parameter transformations
- Euclidian metric can not capture the greediness of the update



We need to find a better metric…

Policies are probabilty distributions

➡We can measure „distances“ of distributions

Better Metric: Relative Entropy or Kullback-Leibler divergence

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

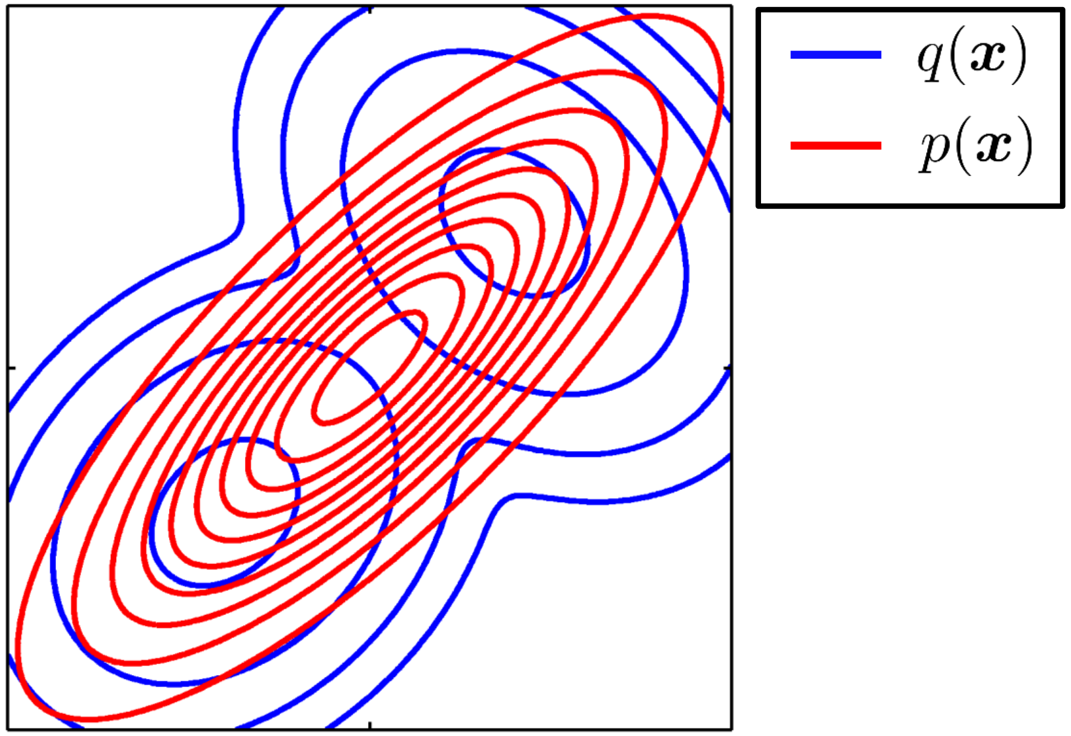
- Information-theoretic „distance“ measure between distributions
- **Properties:**
  - Always larger 0: $\mathrm{KL}(p||q) \geq 0$
  - Only 0 iff both distributions are equal: $\mathrm{KL}(p||q) = 0 \Leftrightarrow p = q$
  - Not symetric, so not a real distance $\mathrm{KL}(p||q) \neq \mathrm{KL}(q||p)$

Kullback-Leibler Divergences

2 types of KLs that can be minimized:

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

- *p* is large where ever *q* is large
- Match the moments of *q* with the moments of *p*
- Same as **Maximum Likelihood** estimate !



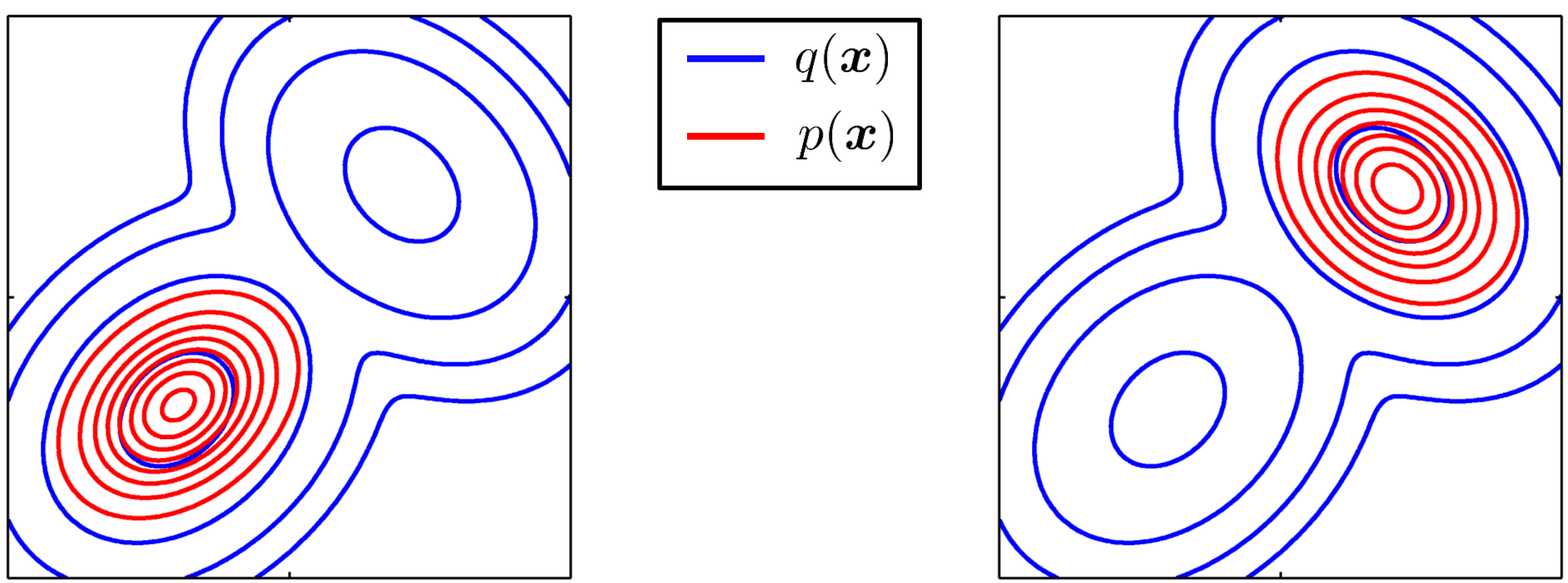
Bishop, 2006

Kullback-Leibler Divergence

2 types of KLs that can be minimized:

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

- *p* is zero wherever *q* is zero (zero forcing): no wild exploration
- not unique for most distributions
- Contains the entropy of *p:* important for exploration



Bishop, 2006

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 a **single parameter influences the distribution**

Natural Gradients

The natural gradient [Amari 1998] uses the Fisher information matrix as metric

- Find direction maximally correlated with gradient
- Constraint: (approximated) KL should be bounded

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

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

The solution to this optimization problem is given as:

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

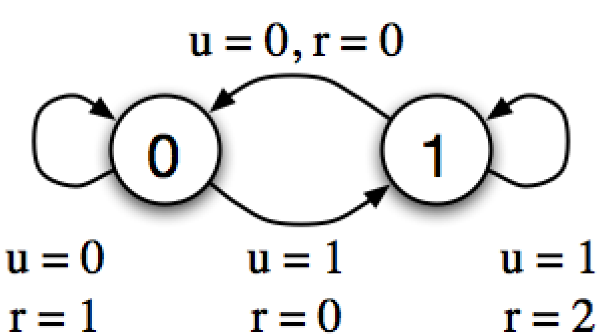
- Inverse of the FIM: every parameter has the same influence!
- Invariant to linear transformations of the parameter space!

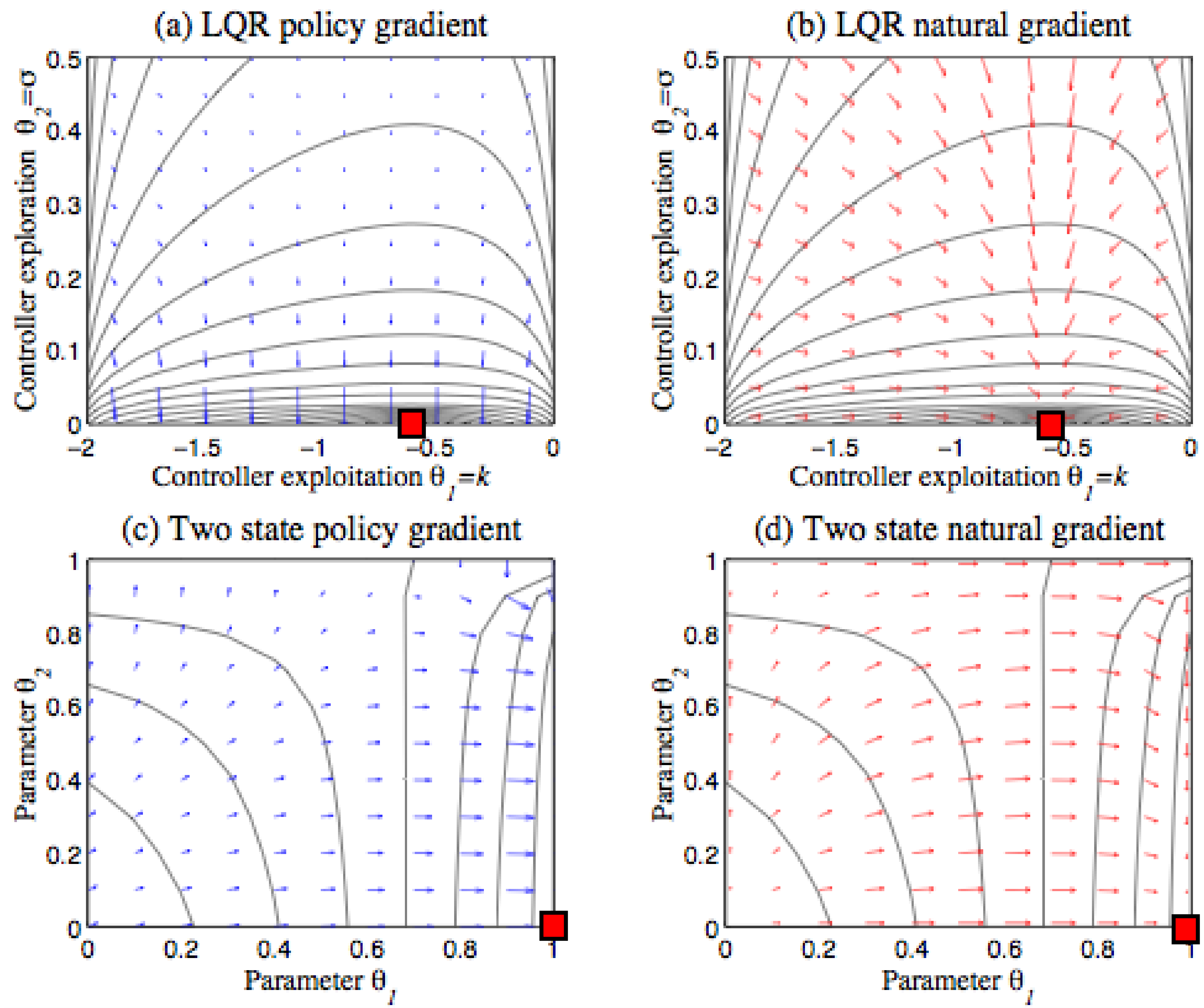
Are they useful?

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!

Computing the Natural Gradient

Episode-Based:

- Natural Evolution Strategy [Sun, Wiestra, Schaul & Schmidhuber, 2009], Rock-Star [Hwangbo & Buchli, 2014]
- FIM can be computed in closed form for Gaussians

Step-Based:

- Natural actor critic [Peters & Schaal, 2006,2008]
- Episodic natural actor critic [Peters & Schaal, 2006]
- Avoid FIM computation due to compatible value function approximation

Computing the NG (step-based)

Back to Policy Gradient Theorem with baseline

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

Estimate the reward to come (minus baseline) by function approximation

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

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

as gradient

It can be shown that this gradient is still unbiased if: $\psi(\boldsymbol{s}, \boldsymbol{a}) = \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a} | \boldsymbol{s})$

- Called compatible function approximation [Sutton 1999]
- Log-gradient of the policy defines optimal features

Compatible Function Approximation

Compatible Function Approximation:

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

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

Baseline

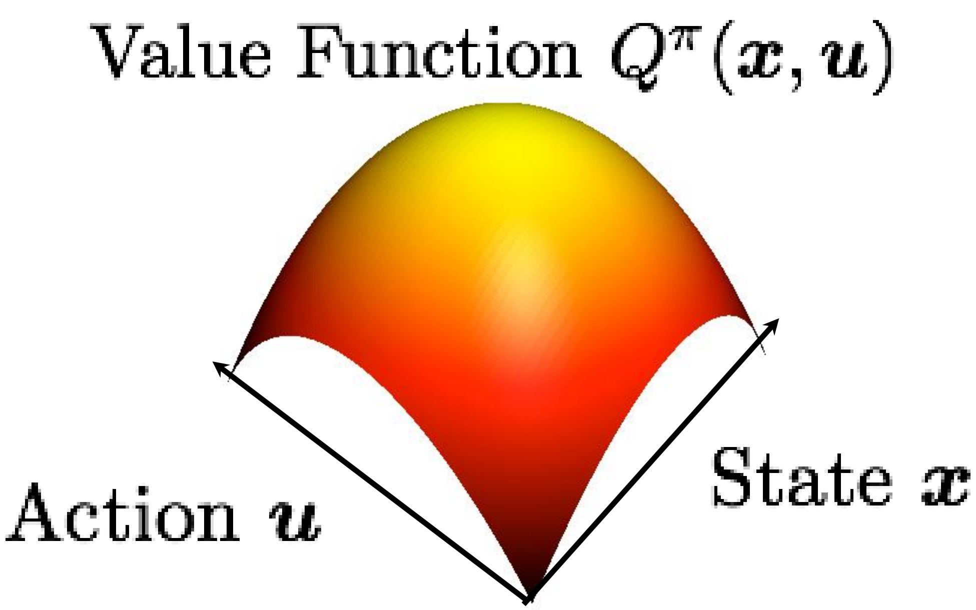
$$f_{\boldsymbol{w}}(\boldsymbol{s},\boldsymbol{a}) = \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{\theta})^T\boldsymbol{w} = Q^\pi(\boldsymbol{s},\boldsymbol{a}) - V^\pi(\boldsymbol{s})$$

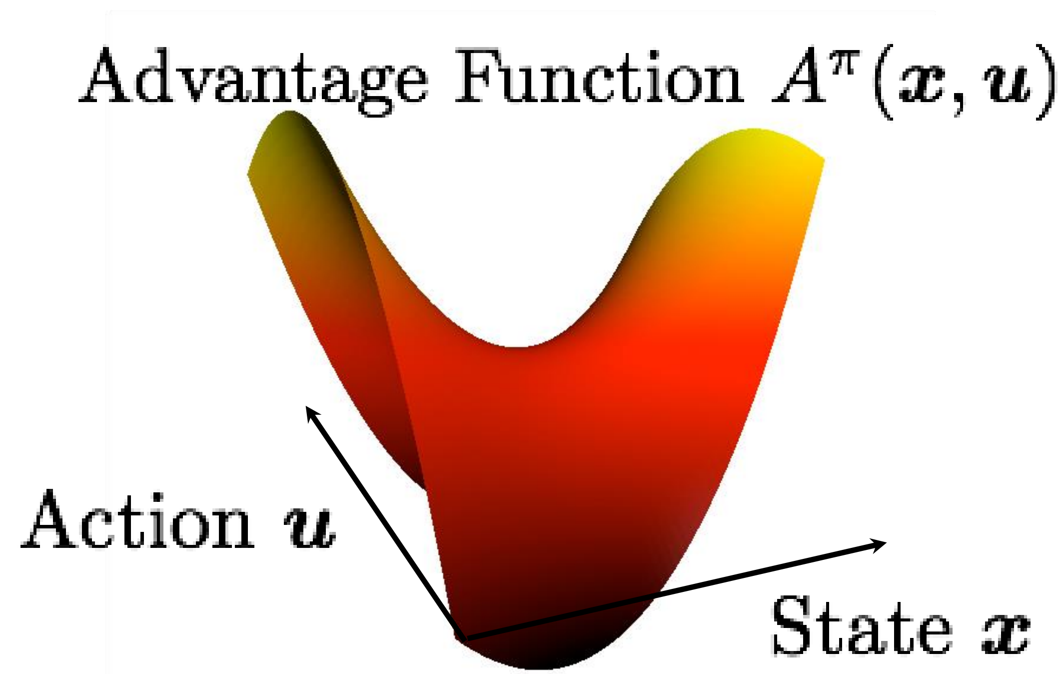
Can the Compatible FA be learned?

The compatible function approximation represents an advantage function [Peters et al. 2003, 2005]

$$f_{\boldsymbol{w}}(\boldsymbol{s}, \boldsymbol{a}) = Q^\pi(\boldsymbol{s}, \boldsymbol{a}) - V^\pi(\boldsymbol{s}) = A^\pi(\boldsymbol{s}, \boldsymbol{a})$$

The advantage function is very different from the value functions





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

Compatible Function Approximation

Gradient with Compatible Function Approximation:

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

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

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

It can be shown that [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

Lets put the parts together:

- Combatible Function Approximation:

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

- [Peters & Schaal, 2008] 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 $\boldsymbol{w}$

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 have only one initial state

Episodic Natural Actor-Critic

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

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

Critic: Episodic Evaluation

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

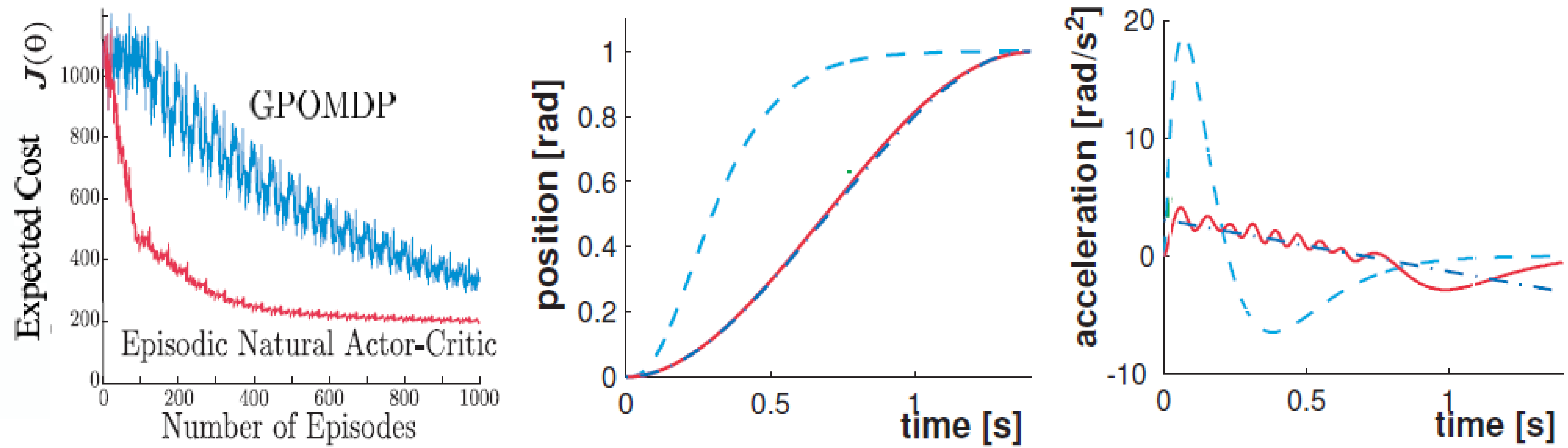
$$\mathbf{R} = [R_1, R_2^T, \ldots, R_N^T]^T$$

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

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)

Learning T-Ball

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

Good performance often after 150-300 trials.

What we have seen from the policy gradients

- Policy gradients dominated policy search for a long time and solidly working methods exist.
- They still need a lot of samples
- We need to tune the learning rate
- Learning the exploration rate / variance is still difficult

Outline

Taxonomy of Policy Search Algorithms

Model-Free Policy Search Methods

- Policy Gradients
  - Likelihood Gradients: REINFORCE [Williams, 1992], PGPE [Rückstiess et al, 2009]
  - Natural Gradients: episodic Natural Actor Critic (eNAC), [Peters & Schaal, 2006]
- Weighted Maximum Likelihood Approaches
  - Success-Matching Principle [Kober & Peters, 2006]
  - Information Theoretic Methods [Daniel, Neumann & Peters, 2012]
- Extensions: Contextual and Hierarchical Policy Search

Model-Based Policy Search Methods

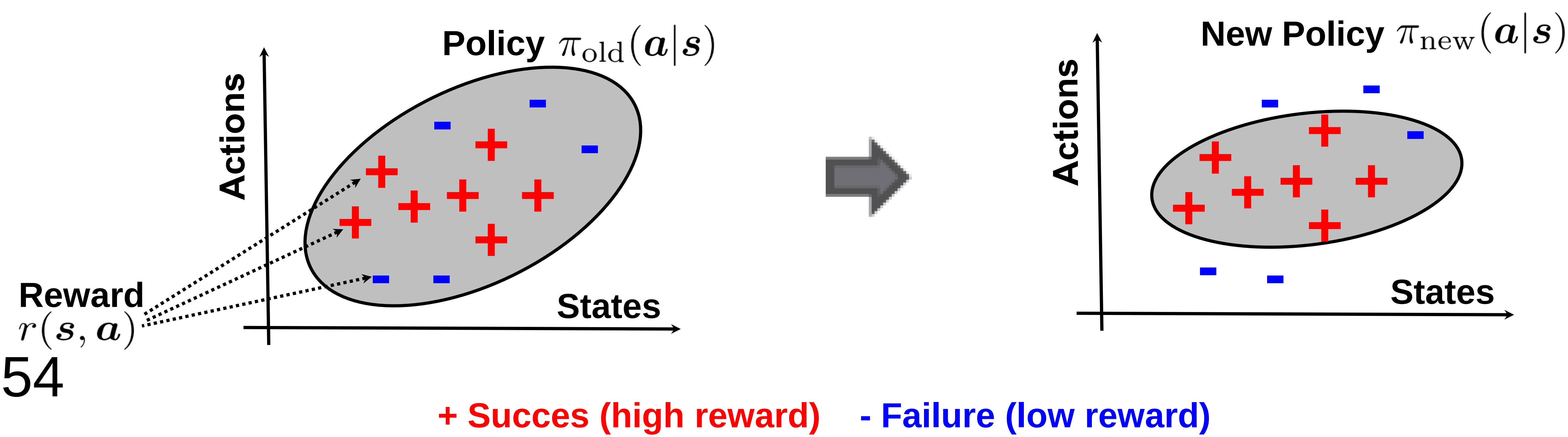
- Greedy Updates: PILCO [Deisenroth & Rasmussen, 2011]
- Bounded Updates: Model-Based REPS [Peters at al., 2010], Guided Policy Search by Trajectory Optimization [Levine & Koltun, 2010]

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 reweighting by success probability *f(r)*

$$\pi_{\text{new}}(\boldsymbol{a}|\boldsymbol{s}) \propto f(r(\boldsymbol{s},\boldsymbol{a}))\pi_{\text{old}}(\boldsymbol{a}|\boldsymbol{s})$$



Success Matching Principle

Success-Matching: policy reweighting by success probability *f(r)*

$$\pi_{\text{new}}(\boldsymbol{a}|\boldsymbol{s}) \propto f(r(\boldsymbol{s},\boldsymbol{a}))\pi_{\text{old}}(\boldsymbol{a}|\boldsymbol{s})$$

Can be derived in many ways:

- Expectation maximization [Kober & Peters., 2008][Vlassis & Toussaint., 2009]
- Optimal Control [Theodorou, Buchli & Schaal, 2010]
- Information Theory [Peters et al, 2010, Daniel, Neumann & Peters, 2012]

Basic principles of all algorithms are similar

- Success probability computation might differ
- Have been derived for step-based (hybrid) and episode-based policy search

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 $p_k(\boldsymbol{\theta}^{[i]})$

Computing the weights...

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

Transform the returns in an **improper probability distribution**

Exponential transformation [Peters 2005]**:**

$$w^{[i]} = \exp(\beta(R^{[i]} - \max R^{[i]})$$

- $\beta \ldots$ Temperature of the distribution
- Often set by heuristics [Kober & Peters, 2008][Theodorou, Buchli, & Schaal, 2010], e.g.:

$$\beta = \frac{10}{\max R^{[i]} - \min R^{[i]}}$$

- Or information theoretic principles [Daniel, Neumann & Peters, 2012]

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 computing the M-projection of $p(\boldsymbol{\theta}^{[i]})$:

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

We sampled from the old policy

Optimization: weighted maximum likelihood estimate!

- Closed form solutions exists, no learning rates!

Weighted Maximum Likelihood Solutions…

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

Weighted mean:

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

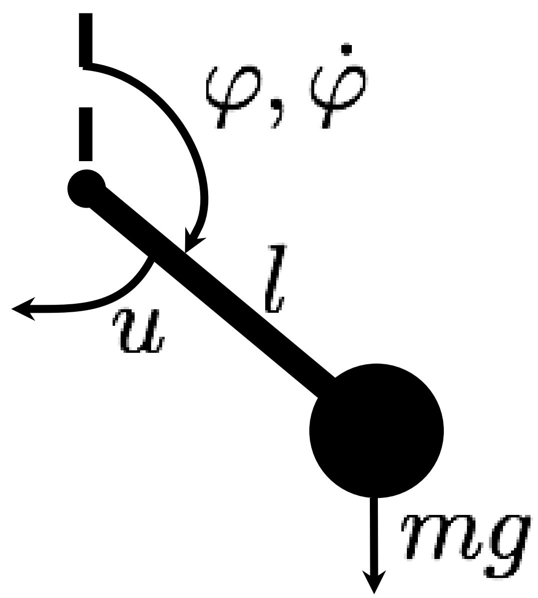
Weighted covariance:

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

- But more general: Also for mixture models, GPs and so on…
- Invariant to transformations of the parameters

Underactuated Swing-Up

swing heavy pendulum up



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

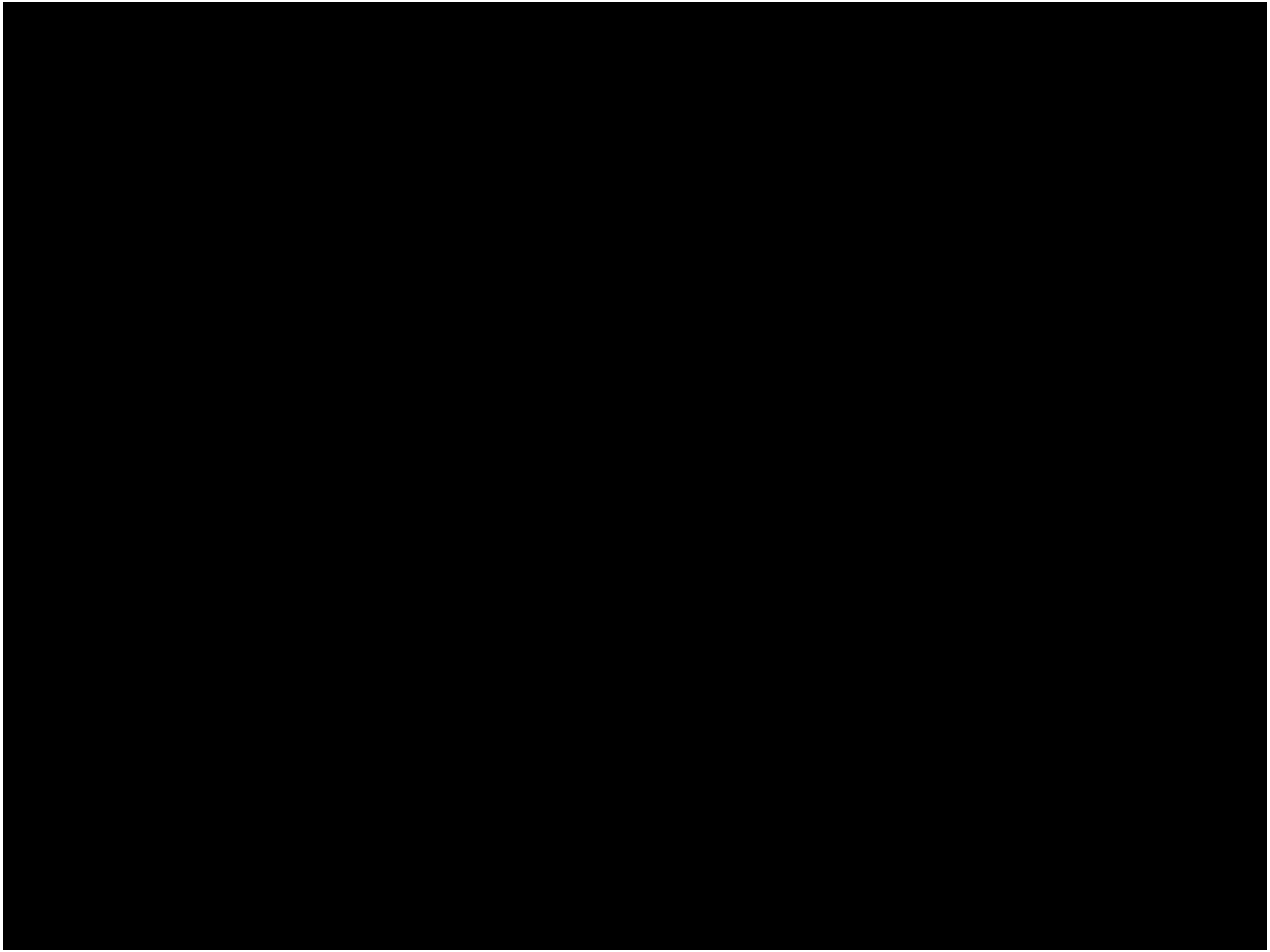
- motor torques limited, Policy: DMPs

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

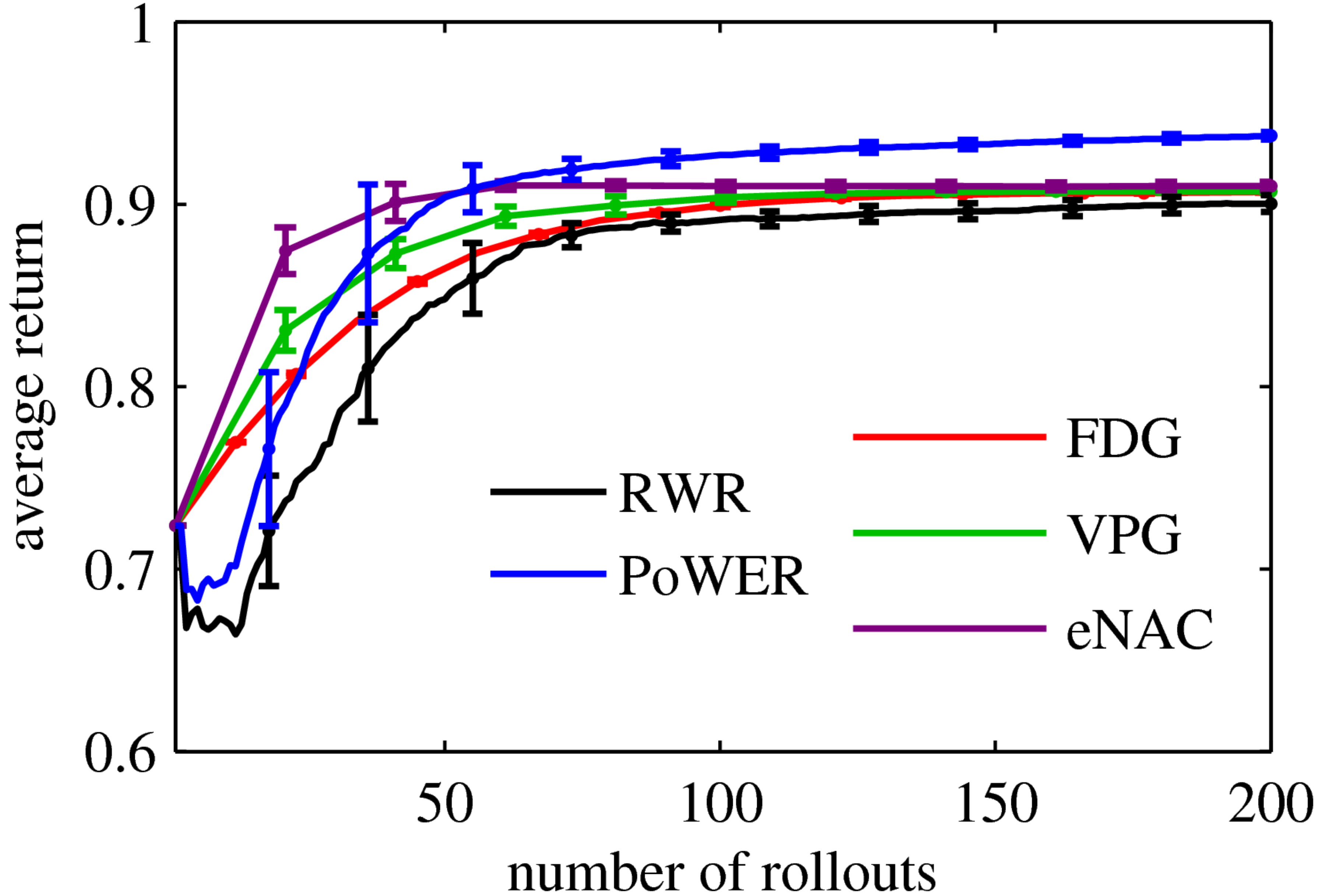
- reward function

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

(Schaal, NIPS 1997; Atkeson, ICML 1997)



Underactuated Swing-Up



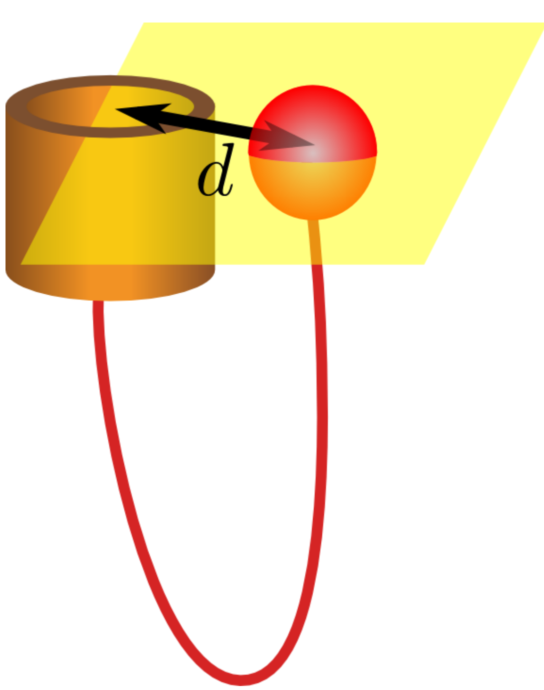
(Peters & Schaal, IROS 2006; Peters & Schaal, ICML 2007)

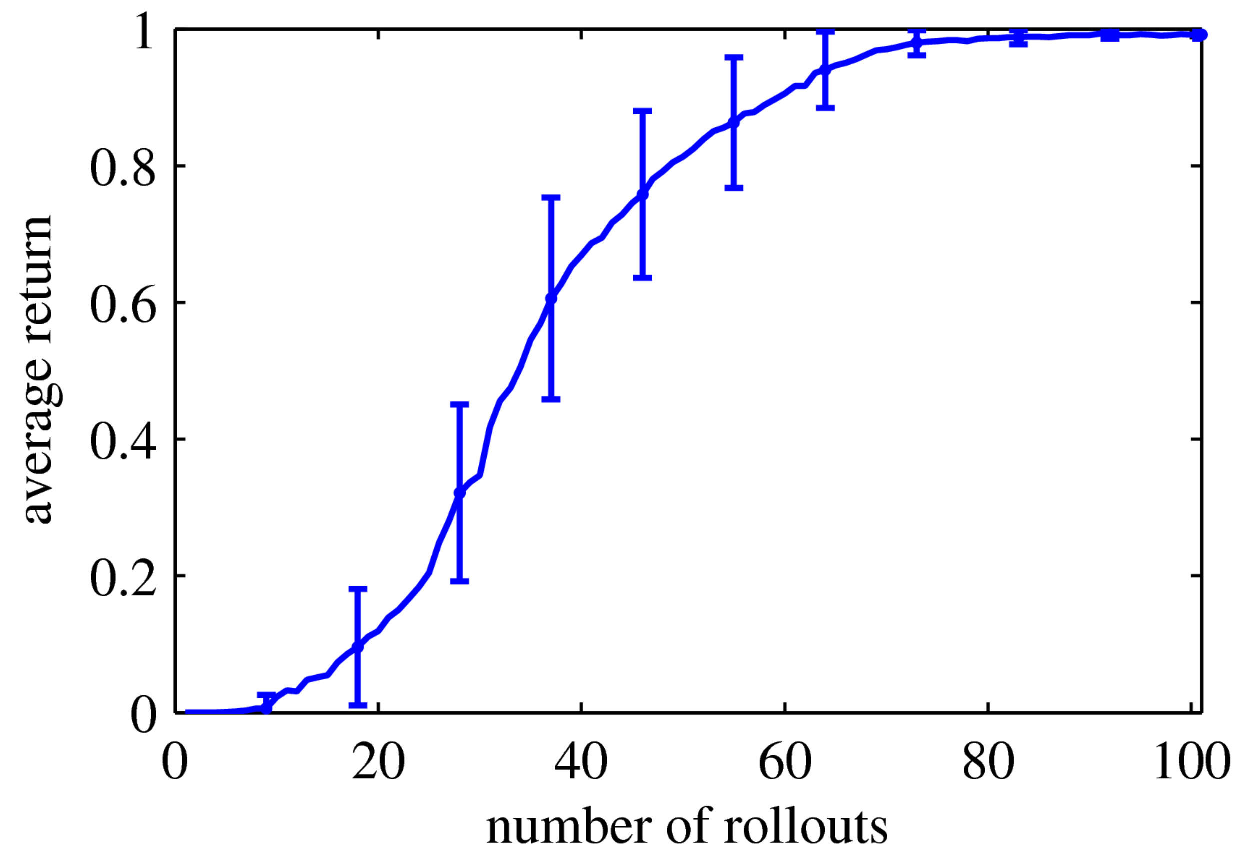
Ball-in-a-Cup [Kober & Peters, 2008]

Reward function:

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

Policy: DMPs





Ball-in-a-Cup

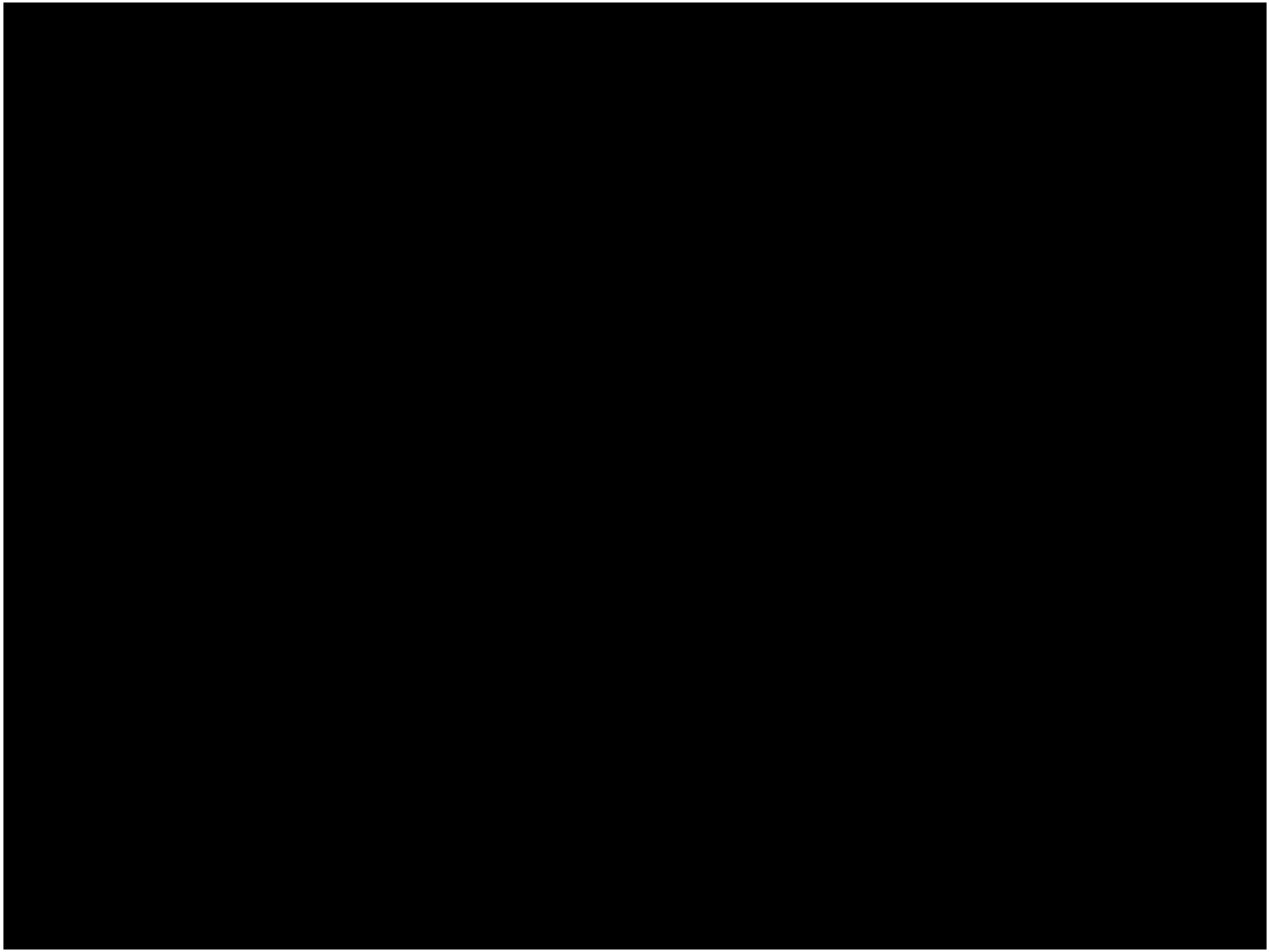
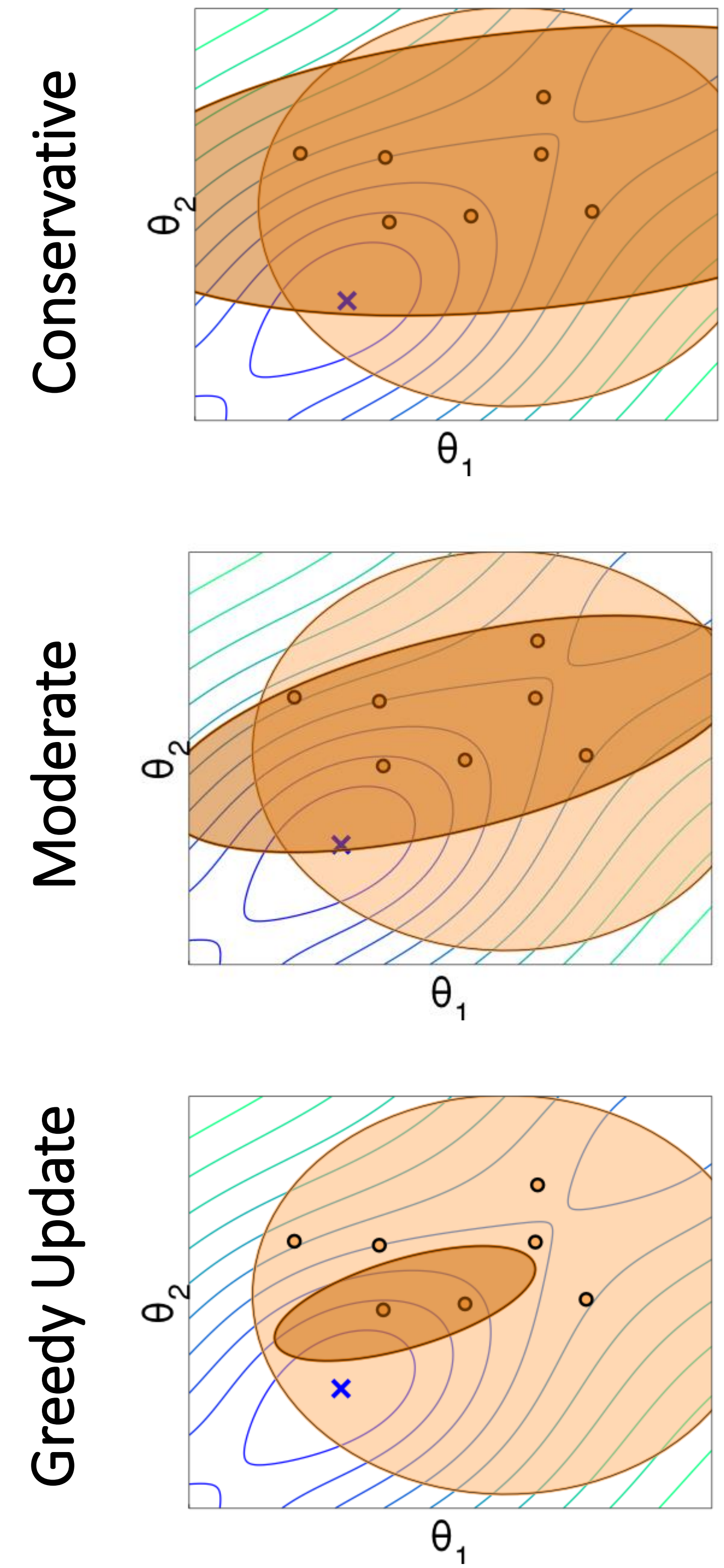


Table Tennis [Mülling, Kober, Krömer & Peters, 2013]

Weighted ML estimates

- Invariant to transformations of the parameters
- No learning rate needs to be tuned
- Controllable exploration-exploitation tradeoff?
  - Difficult… but can be adjusted with temperature β



Outline

Taxonomy of Policy Search Algorithms

Model-Free Policy Search Methods

- Policy Gradients
  - Likelihood Gradients: REINFORCE [Williams, 1992], PGPE [Rückstiess et al, 2009]
  - Natural Gradients: episodic Natural Actor Critic (eNAC), [Peters & Schaal, 2006]
- Weighted Maximum Likelihood Approaches
  - Success-Matching Principle [Kober & Peters, 2006]
  - Information Theoretic Methods [Daniel, Neumann & Peters, 2012]
- Extensions: Contextual and Hierarchical Policy Search

Model-Based Policy Search Methods

- Greedy Updates: PILCO [Deisenroth & Rasmussen, 2011]
- Bounded Updates: Model-Based REPS [Peters at al., 2010], Guided Policy Search by Trajectory Optimization [Levine & Koltun, 2010]

Episodic Relative Entropy Policy Search

For success matching, directly use relative entropy as metric between two policies

We get the following optimization problem:

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

$$\text{s.t:} \quad \text{KL}(\pi(\boldsymbol{\theta}) || q(\boldsymbol{\theta})) \leq \epsilon$$ Stay close to the old policy $q(\boldsymbol{\theta})$

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

- Stay close to the data
- Epsilon directly controls the exploration-exploitation trade-off
  - $\epsilon = 0 \ldots$ continue to explore with policy $q(\boldsymbol{\theta})$
  - $\epsilon \rightarrow \infty \ldots$ greedily jump to best sample

Relative Entropy Policy Search

Which has the following analytic solution:

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

- That‘s exactly sucess matching with exponential transformation!
- **Scalingfactor** $\eta = 1/\beta$:
  - Automatically chosen from optimization (Lagrange Multiplier)
  - Specified by KL-bound ϵ
- How to compute η ?
  - Solve the dual problem [Boyd&Vandenberghe, 2004]
  - Convex Optimization

Outline

Taxonomy of Policy Search Algorithms

Model-Free Policy Search Methods

- Policy Gradients
  - Likelihood Gradients: REINFORCE [Williams, 1992], PGPE [Rückstiess et al, 2009]
  - Natural Gradients: episodic Natural Actor Critic (eNAC), [Peters & Schaal, 2006]
- Weighted Maximum Likelihood Approaches
  - Success-Matching Principle [Kober & Peters, 2006]
  - Information Theoretic Methods [Daniel, Neumann & Peters, 2012]
- Extensions: Contextual and Hierarchical Policy Search

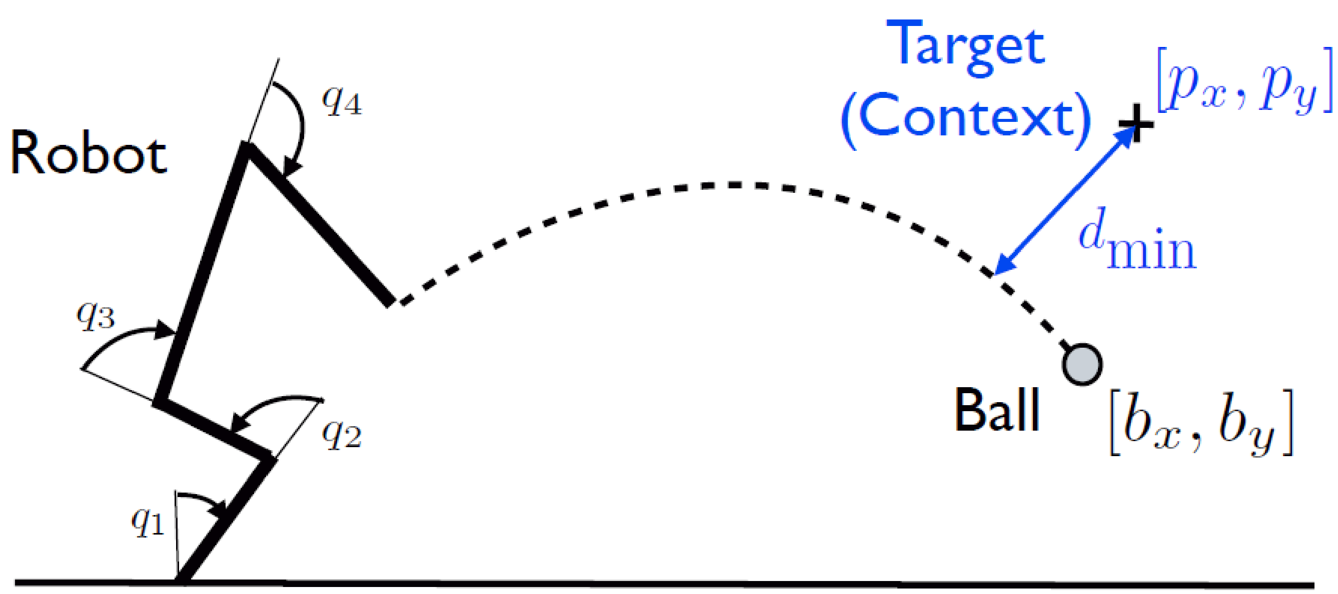
Model-Based Policy Search Methods

- Greedy Updates: PILCO [Deisenroth & Rasmussen, 2011]
- Bounded Updates: Model-Based REPS [Peters at al., 2010], Guided Policy Search by Trajectory Optimization [Levine & Koltun, 2010]

Extension: Contextual Policy Search with REPS

Context:

- Context $\boldsymbol{x}$ describes objectives of the task (fixed before task execution)
- E.g.: Target location to throw a ball
- Adapt the control policy parameters $\boldsymbol{\theta}$ to the target location $\boldsymbol{x}$



Contextual Policy Search with REPS

[Kupscik, Deisenroth, Peters & Neumann, 2013]

Context:

- Context ***x*** describes objectives of the task (fixed before task execution)
- E.g.: Target location to throw a ball
- Adapt the control policy parameters $\boldsymbol{\theta}$ to the target location ***x***
- Learn an upper level policy $\pi(\boldsymbol{\theta}|\boldsymbol{x};\boldsymbol{\omega})$

Objective:

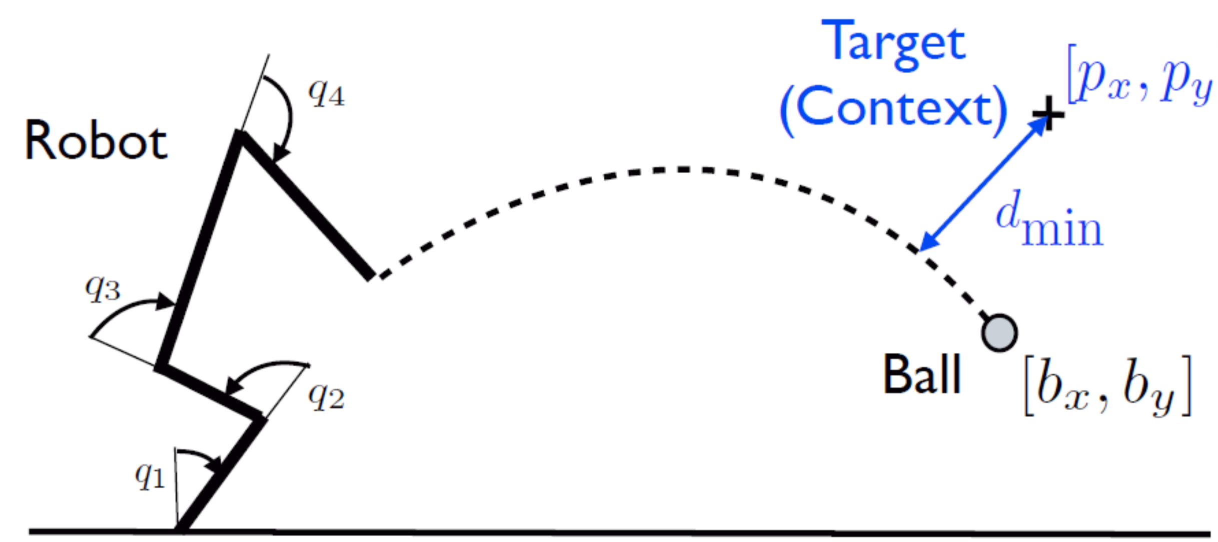
$$J_\pi = \iint \mu_0(\boldsymbol{x})\pi(\boldsymbol{\theta}|\boldsymbol{x})\mathcal{R}_{\boldsymbol{x}\boldsymbol{\theta}} d\boldsymbol{x} d\boldsymbol{\theta}$$

- Average reward over all contexts
- $\mu_0(\boldsymbol{x})$ ...context distribution

Dataset for policy update:

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

- Also contains context vectors



Contextual Policy Search with REPS

[Kupscik, Deisenroth, Peters & Neumann, 2013]

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

- Otherwise independent optimization problems for each context

We get the following optimization problem [CITE]:

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

maximize rewards

$$\text{s.t.:} \quad \sum_{\boldsymbol{x},\boldsymbol{\theta}} p(\boldsymbol{x}, \boldsymbol{\theta}) = 1$$

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 $\boldsymbol{\mu}_0(\boldsymbol{x})$

Contextual Policy Search with REPS

[Kupscik, Deisenroth, Peters & Neumann, 2013]

Closed form solution:

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

- We automatically get a baseline *V(x)* for the returns
- Function approximation for *V(x)* achieved by matching feature averages instead of distributions

$$\sum_{\boldsymbol{x}} p(\boldsymbol{x})\phi(\boldsymbol{x}) = \hat{\phi} \quad \Rightarrow \quad V(\boldsymbol{x}) = \boldsymbol{\phi}^T(\boldsymbol{x})\boldsymbol{v}$$

- $\boldsymbol{v}\ldots$ given by Lagrangian multipliers
- Obtain $\boldsymbol{v}$ again by optimizing the dual

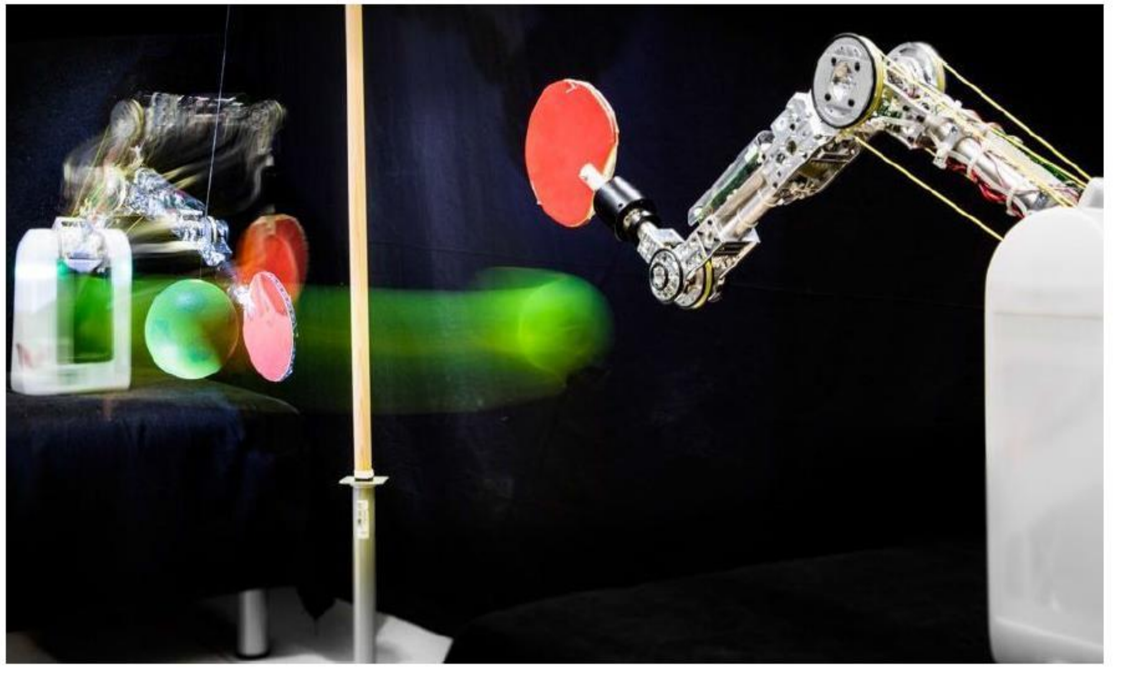
Policy $\pi(\boldsymbol{\theta}|\boldsymbol{x};\boldsymbol{\omega}_{k+1})$ again obtained by a weighted maximum likelihood estimate

- E.g. weighted linear regression in the simplest case

Results: Thetherball

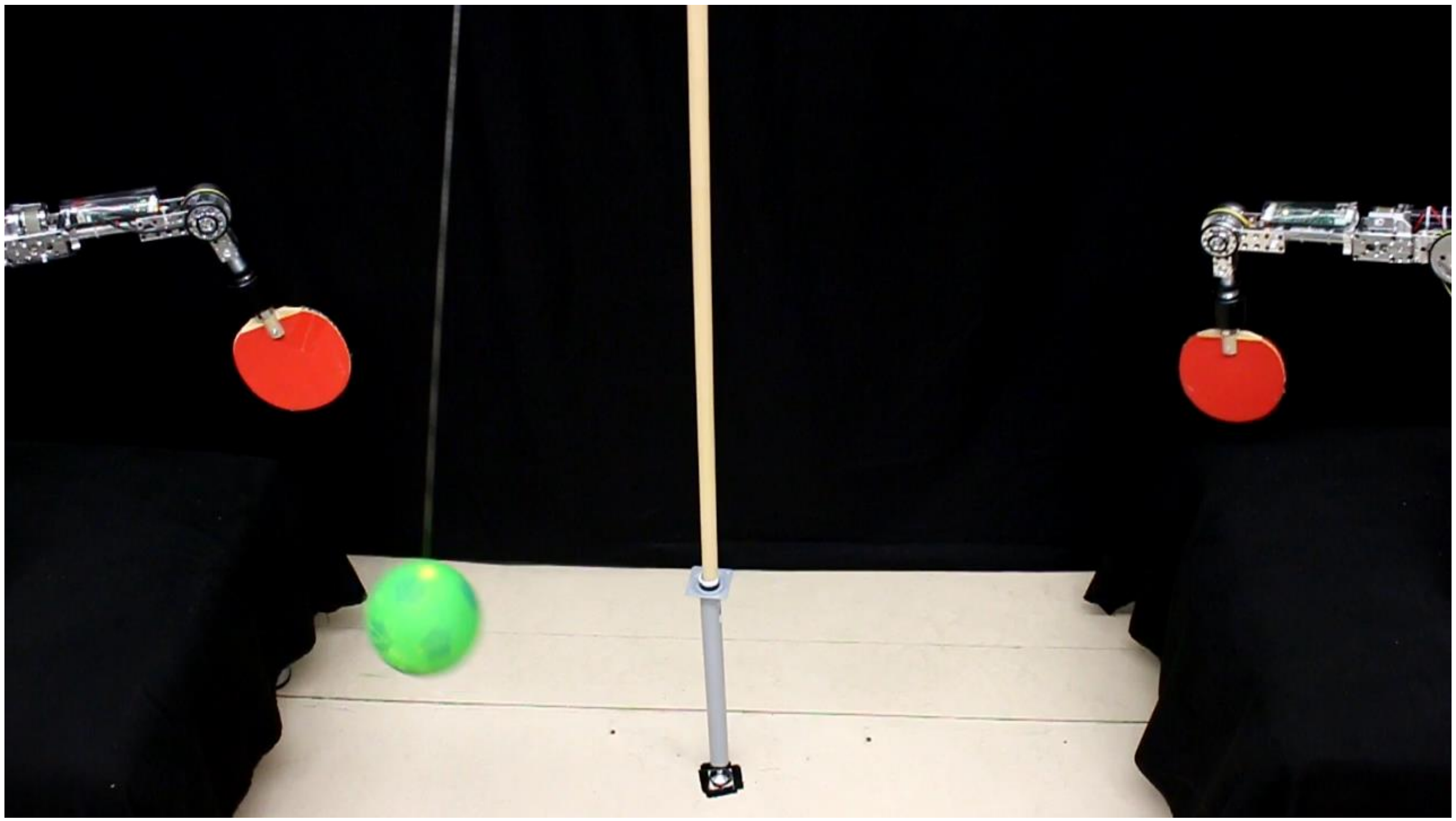
Tetherball:

- Six degrees of freedom
- Highly dynamic behavior due to springs
- Cable driven lightweight robots
- Very complex forward dynamics model
- High dimensional context space (TODO!)



[Parisi, Peters, et. al, IROS 2015]

Real Robot Experiment



Player	Hit rate	Matches won	Total score
Analytical	71%	6/25	8
Learned	**85%**	**19/25**	**38**

Extension: Learning Hierarchical Policies with REPS [Daniel, Neumann & Peters, 2012]

Motivation:

- Many motor tasks have multiple solutions.
- We want to learn all of them

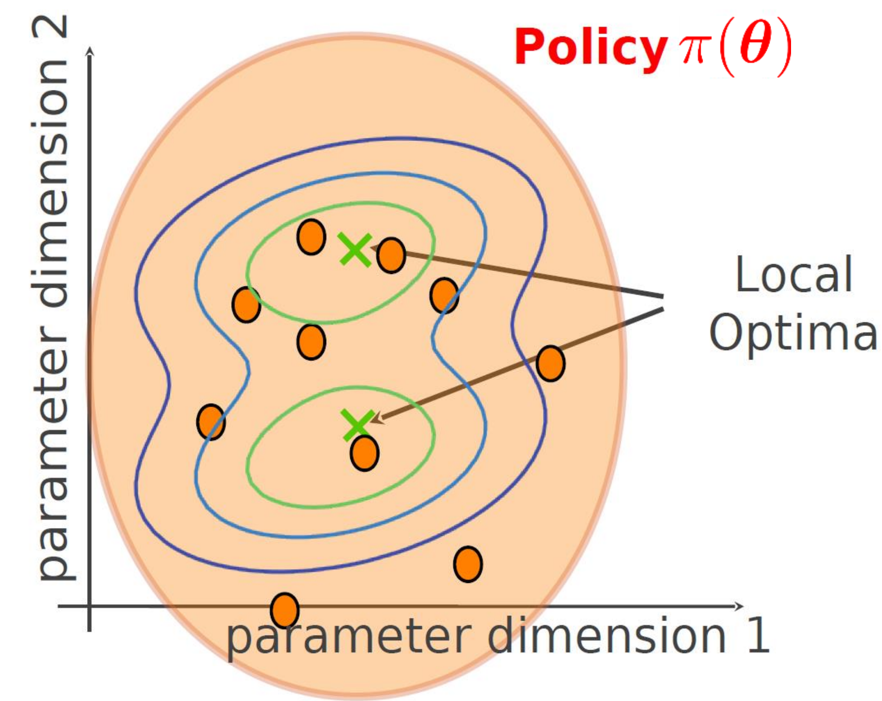
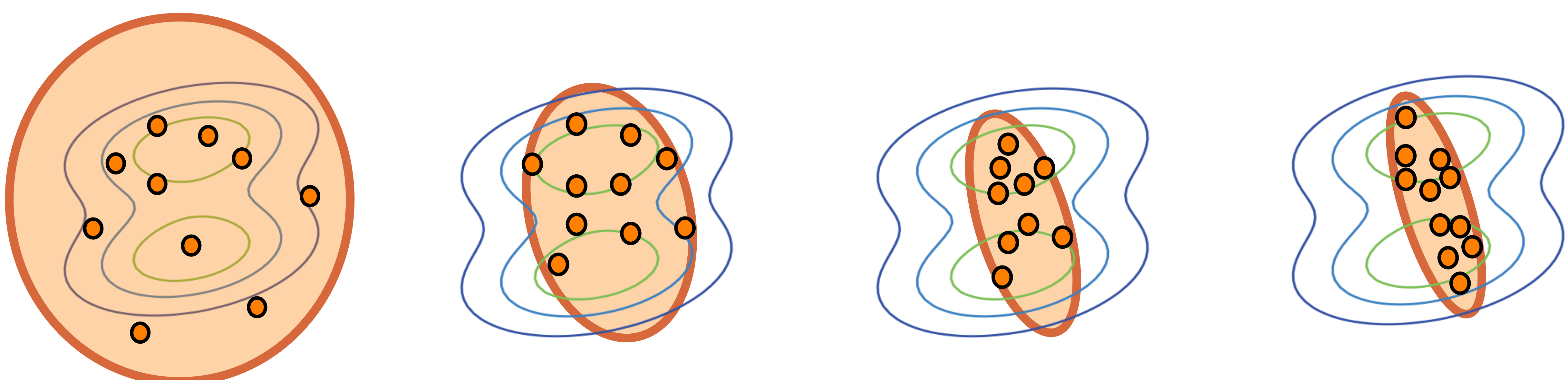


Illustration: The weighted ML update averages over all solutions!

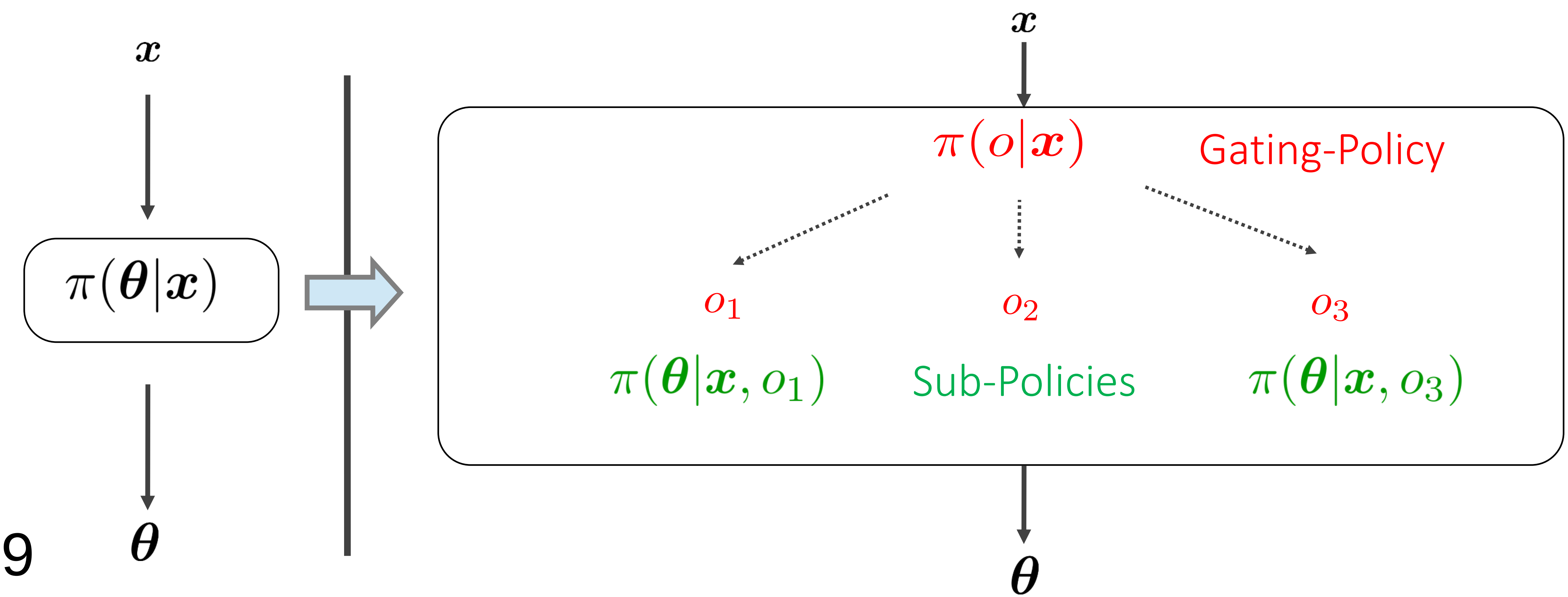


Introduce Hierarchy

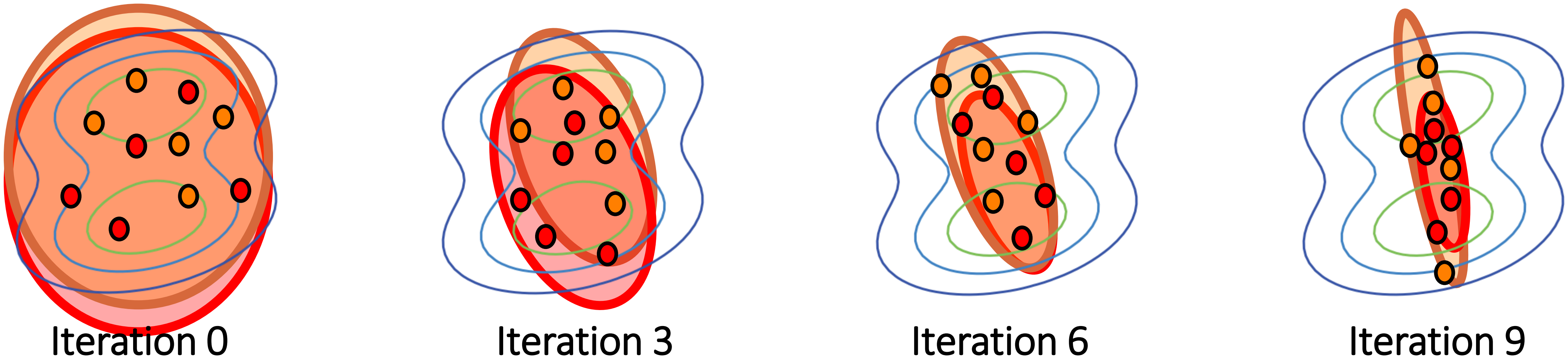
Upper-level policy $\pi(\boldsymbol{\theta}|\boldsymbol{x})$ as hierarchical policy

- Selection of the sub-policy: Gating-policy $\pi(o|\boldsymbol{x})$
- Selection of the parameters: Sub-policy $\pi(\boldsymbol{\theta}|\boldsymbol{x}, o)$
- Structure of the hierarchical policy:

$$\pi(\boldsymbol{\theta}|\boldsymbol{x}) = \sum_{o} \pi(o|\boldsymbol{x})\pi(\boldsymbol{\theta}|\boldsymbol{x}, o)$$



Learning versatile Sub-Policies



Sub-Policies should represent distinct solutions.

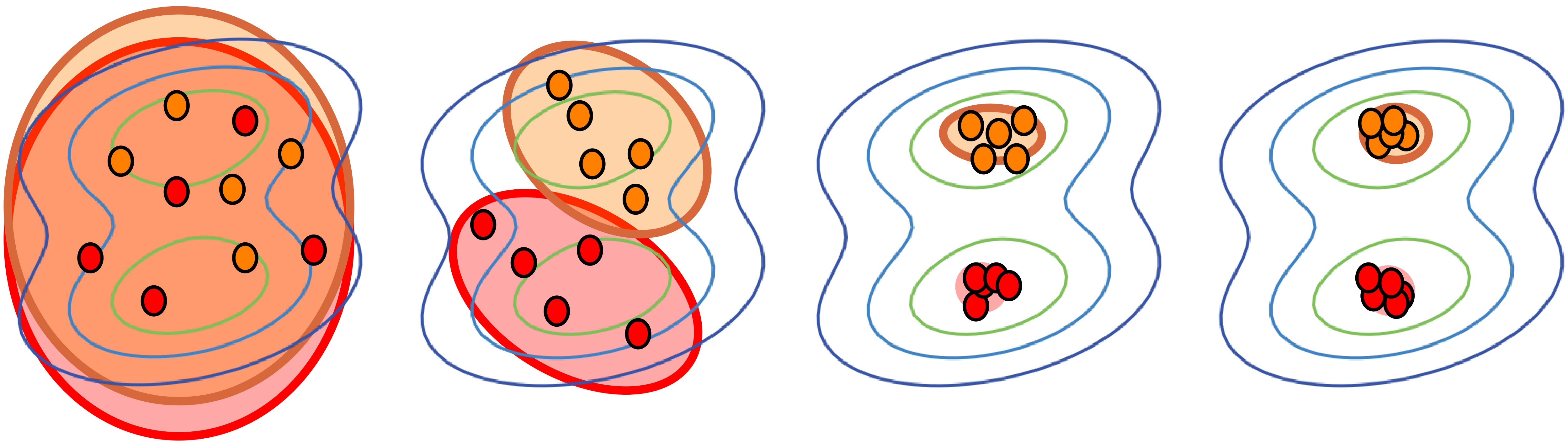
⇨ Limit the overlap of the options

- Responsibilities $p(o|\boldsymbol{x},\boldsymbol{\theta})$ tell us whether we can identify an option, given
  - High entropy of responsibilities $p(o|\boldsymbol{x},\boldsymbol{\theta})$ ⇨ high overlap
  - Limit the entropy $p(o|\boldsymbol{x},\boldsymbol{\theta})$ ⇨ less overlap

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

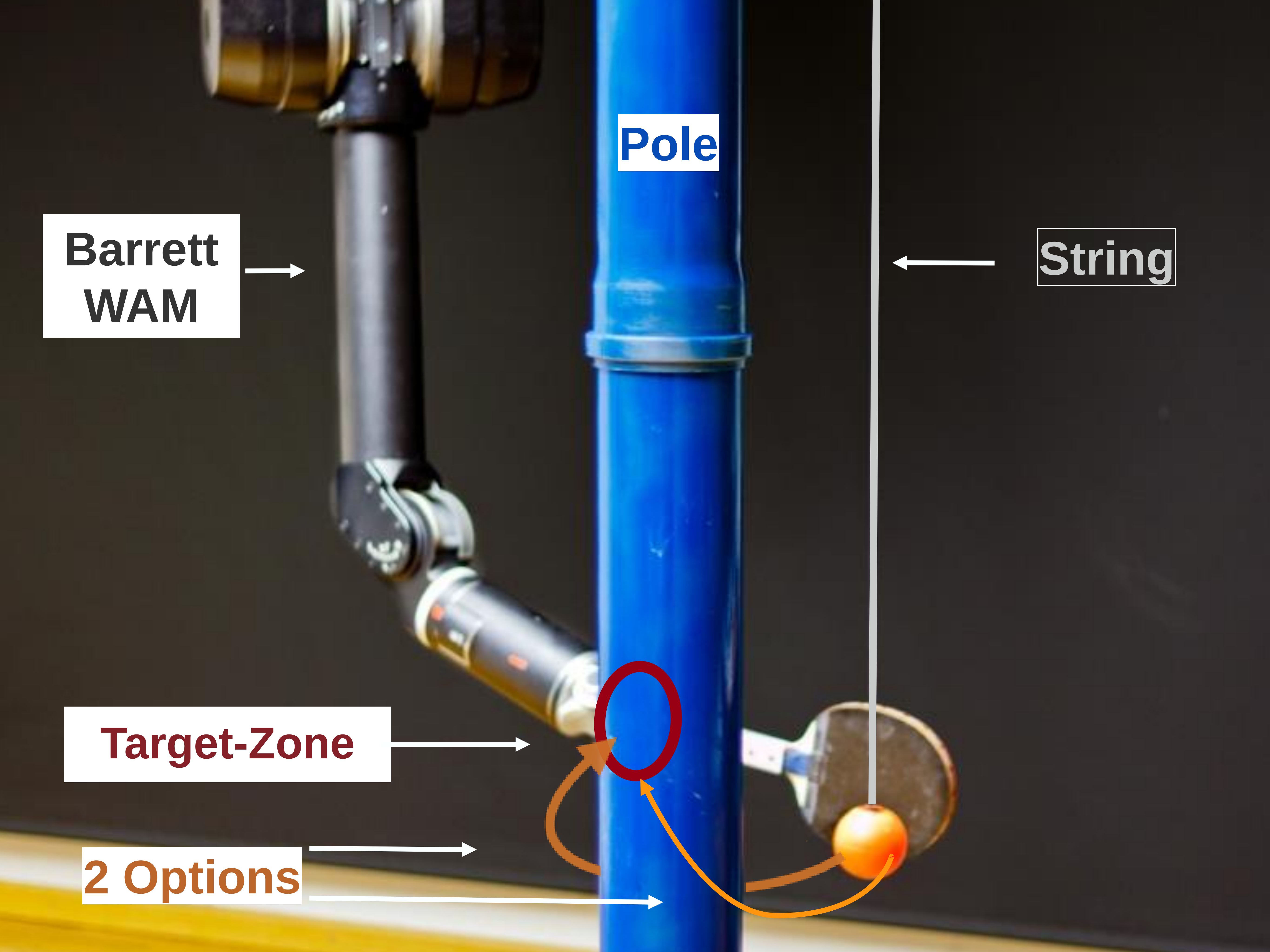
Hierarchical REPS

Bounding the overlap of sub-policies:

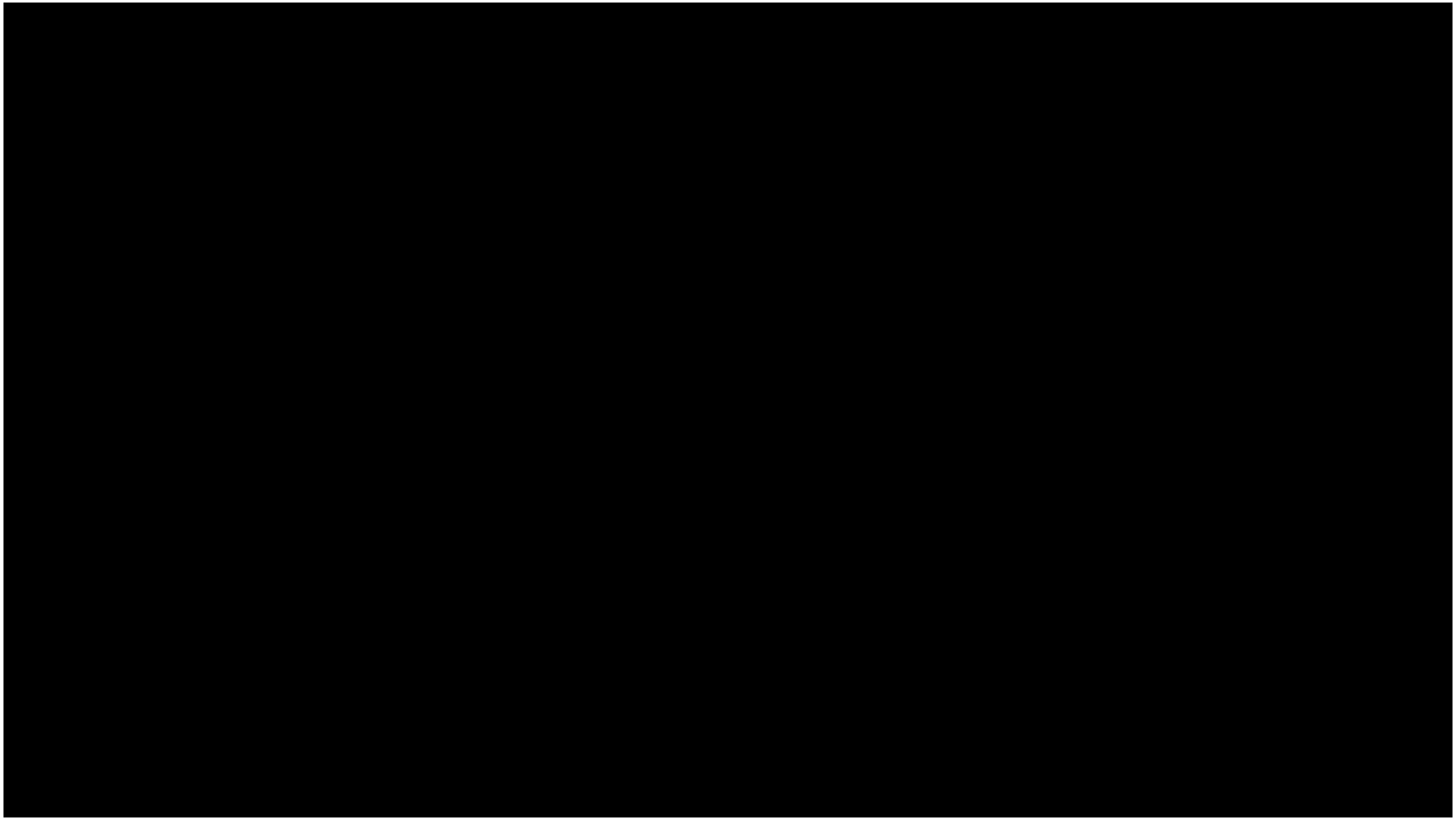


Iteration 0 Iteration 3 Iteration 6 Iteration 9

Learning of versatile, distinct solutions due to separation of sub-policies.


Pole
Barrett WAM
String
Target-Zone
2 Options

Video



Outline

Taxonomy of Policy Search Algorithms

Model-Free Policy Search Methods

- Policy Gradients
  - Likelihood Gradients: REINFORCE [Williams, 1992], PGPE [Rückstiess et al, 2009]
  - Natural Gradients: episodic Natural Actor Critic (eNAC), [Peters & Schaal, 2006]
- Weighted Maximum Likelihood Approaches
  - Success-Matching Principle [Kober & Peters, 2006]
  - Information Theoretic Methods [Daniel, Neumann & Peters, 2012]
- Extensions: Contextual and Hierarchical Policy Search

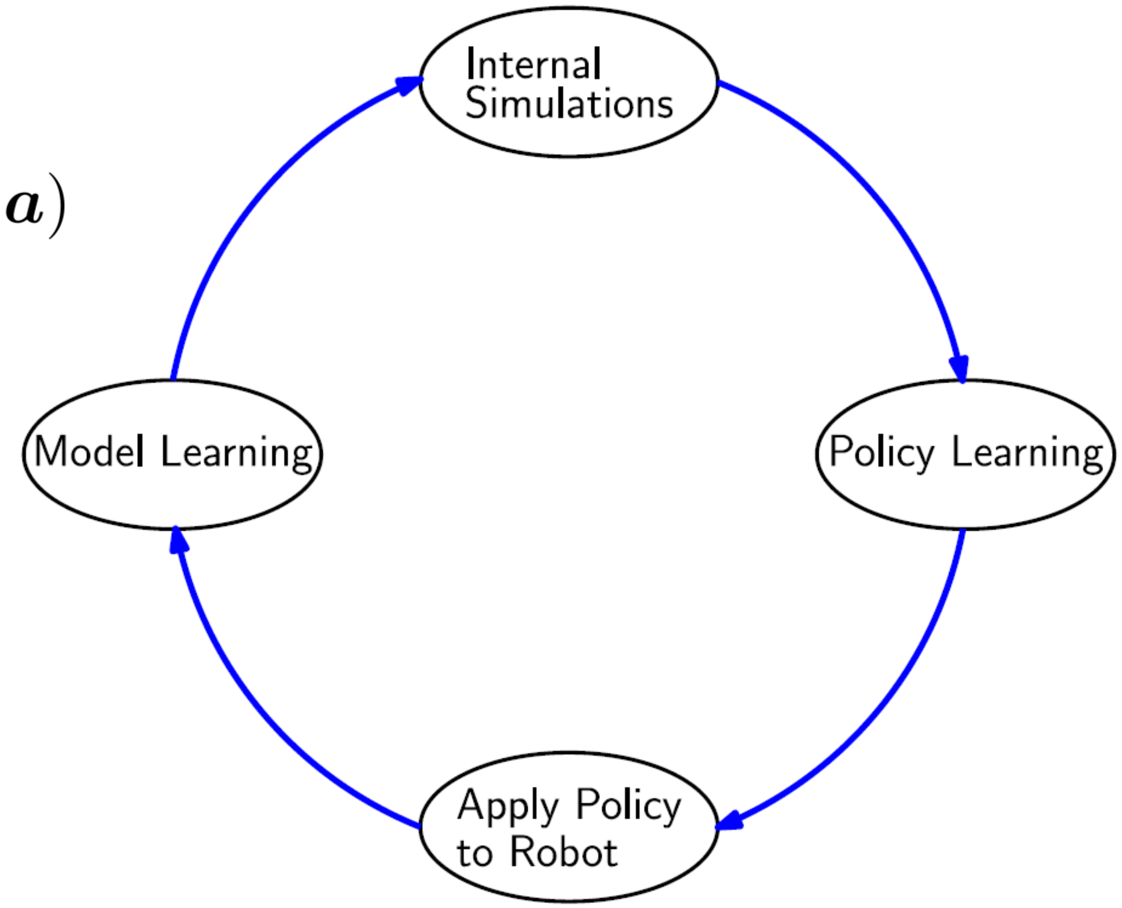
Model-Based Policy Search Methods

- Greedy Updates: PILCO [Deisenroth & Rasmussen, 2011]
- Bounded Updates: Model-Based REPS [Peters at al., 2010], Guided Policy Search by Trajectory Optimization [Levine & Koltun, 2010]

Model-Based Policy Search Methods

Learn dynamics model from data-set

$$\mathcal{D} = \left\{ \left(\boldsymbol{s}_{1:T}^{[i]}, \boldsymbol{a}_{1:T-1}^{[i]} \right) \right\} \rightarrow \hat{\mathcal{P}}(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) \approx \mathcal{P}(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$$



+ More data efficient than model-free methods
+ More complex policies can be optimized
  - RBF networks [Deisenroth & Rasmussen, 2011]
  - Time-dependent feedback controllers [Levine & Koltun, 2014]
  - Gaussian Processes [Von Hoof, Peters & Nemann, 2015]
  - Deep neural nets [Levine & Koltun, 2014][Levine & Abbeel, 2014]

Limitations:

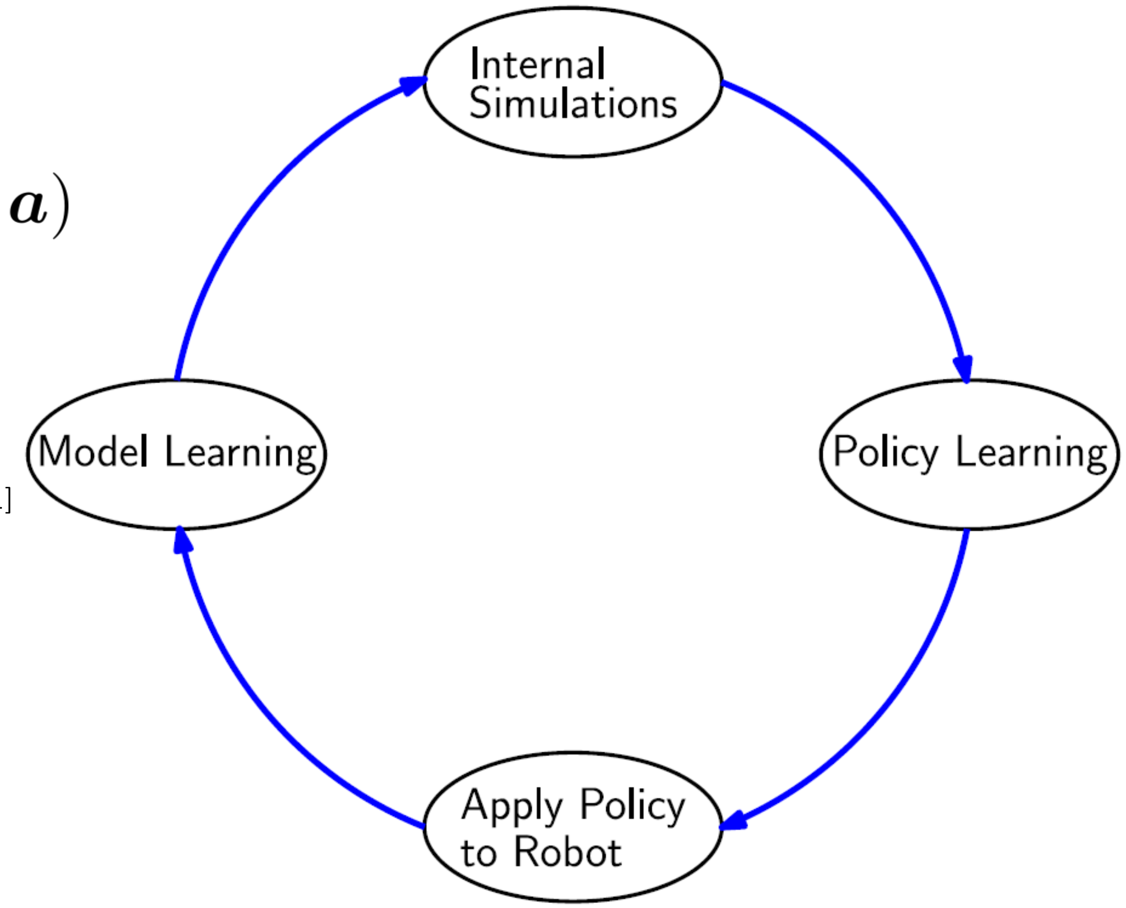
- Learning good models is often very hard
- Small model errors can have drastic damage on the resulting policy (due to optimization)
- Some models are hard to scale
- Computational Complexity

Model-Based Policy Search Methods

Learn dynamics model from data-set

$$\mathcal{D} = \left\{ \left(\boldsymbol{s}_{1:T}^{[i]}, \boldsymbol{a}_{1:T-1}^{[i]} \right) \right\} \rightarrow \hat{\mathcal{P}}(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) \approx \mathcal{P}(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$$

- Gaussian Processes [Deisenroth & Rasmussen 2011] [Kupcsik, Deisenroth, Peters & Neumann, 2013]
- Bayesian Locally Weighted Regression [Bagnell & Schneider, 2001]
- Time-Dependent Linear Models [Lioutikov, Peters, Neumann 2014] [Levine & Abbeel 2014]



Use learned model as simulator

- Sampling [Kupcsik, Diesenroth, Peters & Neumann 2013][Ng 2000]
- (Approximate) probabilistic Inference [Deisenroth & Rasmussen 2011, Levine & Koltun, 2014]

Update Policy

- Model-free methods on virtual sample trajectories [Kupcsik, Diesenroth, Peters & Neumann 2013]
- Analytic Policy Gradients [Deisenroth & Rasmussen, 2011]
- Trajectory optimization [Levine & Koltun, 2014]

Metrics used in Model-Based Policy Search

Bound the policy update for model-based policy search?

- Greedy methods: [Deisenroth & Rasmussen, 2011, Ng et al. 2001]
  - Deterministic policy
  - Compute optimal policy based on current model
  - Exploration: Optimistic UCB like exploration bonus can be used
- “Bounded” methods: [Kupcsik Deisenroth, Peters & Neumann, 2013][Levine & Koltun 2014][Lioutikov, Peters, Neumann 2014]
  - Stochastic Policy
  - The model is only correct in the vicinity of the data-set

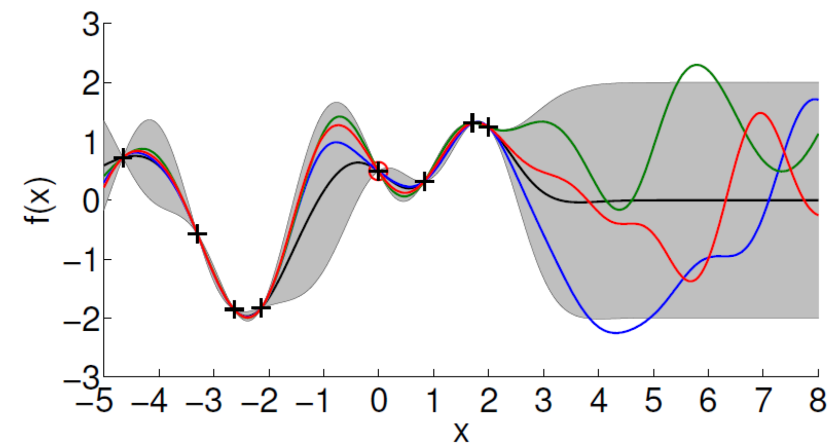
  ➡ Stay close to the data!
  - All these methods use some sort of KL-bound

  ➡ Ideas from model-free PS directly transfer
  - Exploration: Step-size of the policy update is bounded

Greedy Policy Updates: PILCO [Deisenroth & Rasmussen 2011]

Model Learning:

- Use Bayesian models which integrate out model uncertainty ➡ Gaussian Processes
- Reward predictions are not specialized to a single model

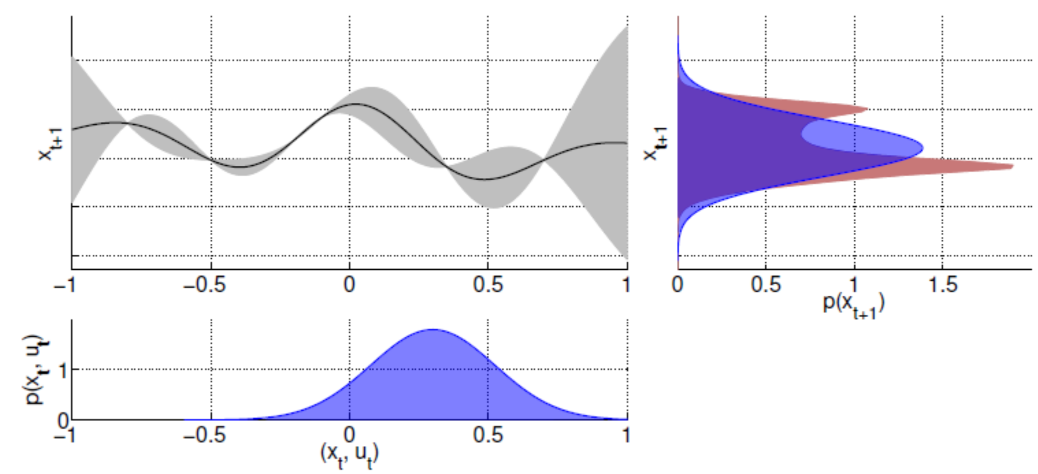


Internal Stimulation:

- Iteratively compute $p(\boldsymbol{s}_1|\boldsymbol{\theta})\dots p(\boldsymbol{s}_T|\boldsymbol{\theta})$

$$p(\boldsymbol{s}_t|\boldsymbol{\theta}) = \int \underbrace{\hat{\mathcal{P}}(\boldsymbol{s}_t|\boldsymbol{s}_{t-1}, \pi(\boldsymbol{s};\boldsymbol{\theta}))}_{\text{GP prediction}} \underbrace{p(\boldsymbol{s}_{t-1}|\boldsymbol{\theta})}_{\mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t)} d\boldsymbol{s}_{t-1}$$

- Moment matching: deterministic approximate inference



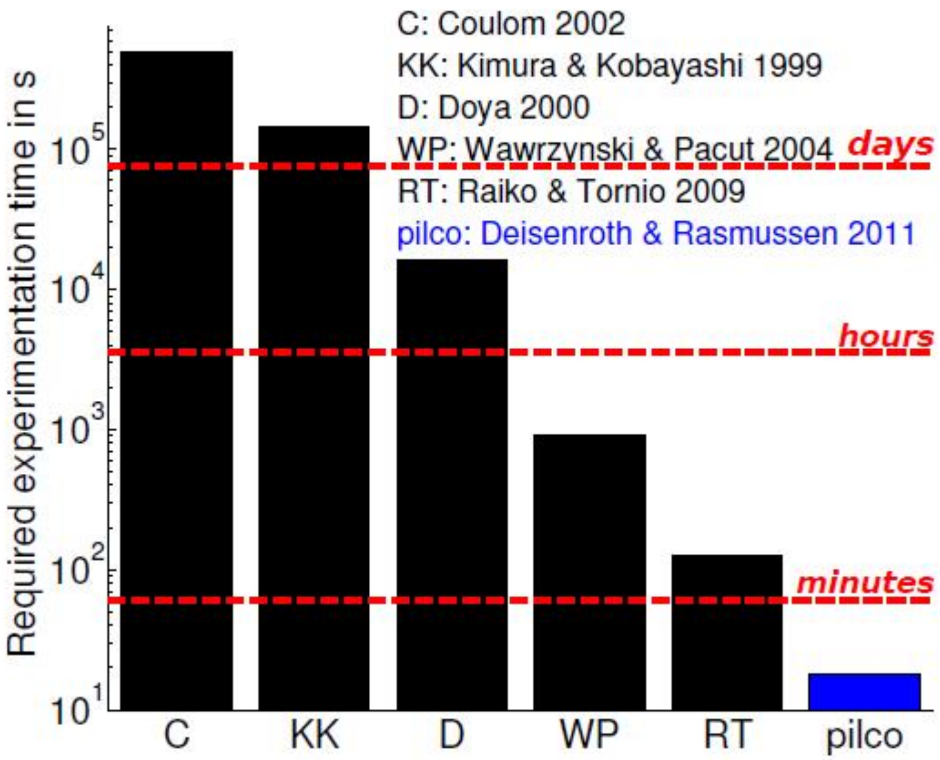
Policy Update:

- Analytically compute expected return and its gradient
- Greedily Optimize with BFGS

$$J_{\boldsymbol{\theta},\hat{\mathcal{P}}} = \sum_{t=1}^{T} \int p(\boldsymbol{x}_t|\boldsymbol{\theta}) r(\boldsymbol{x}_t) d\boldsymbol{x}_t$$

$$\boldsymbol{\theta}_{\text{new}} = \arg\min_{\boldsymbol{\theta}} J_{\boldsymbol{\theta},\hat{\mathcal{P}}}$$

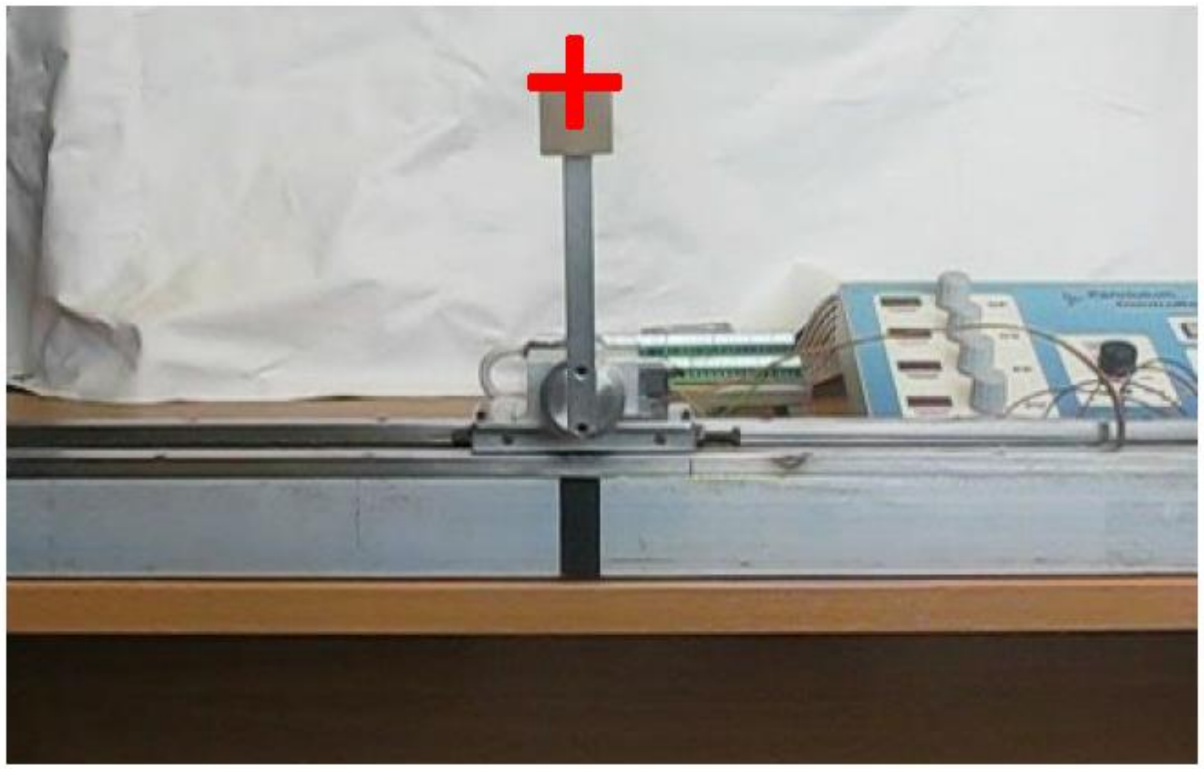
PILCO: some results

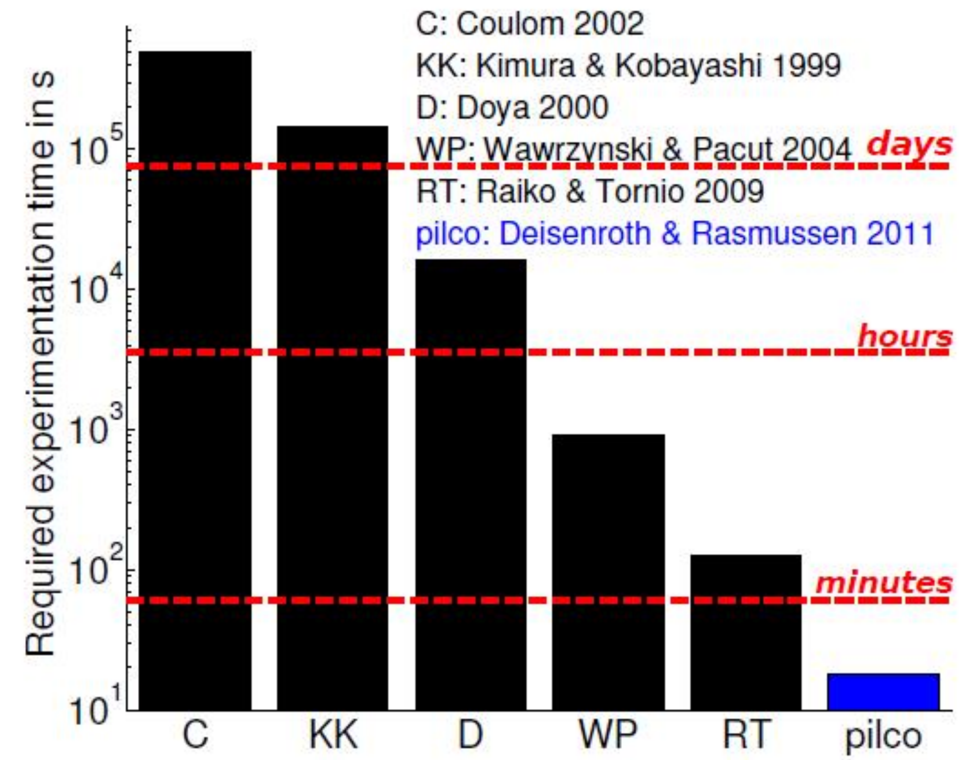


- Swing up and balance a freely swinging pendulum on a cart
- No knowledge about nonlinear dynamics Learn from scratch
- Unprecedented learning speed compared to state-of-the-art (2011)

More applications: Learning to Pick up Objects [Bischoff et al. 2013] Controlling Throttle Valves in Combustion Engines [Bischoff et al. 2014]

PILCO: some results





- Swing up and balance a freely swinging pendulum on a cart
- No knowledge about nonlinear dynamics Learn from scratch
- Unprecedented learning speed compared to state-of-the-art (2011)

Also some limitations:

- GP-models are hard to scale to high-D
- Computationally very demanding
- Can only be used for specific parametrizations of the policy and the reward function

Metrics used in Model-Based Policy Search

Bound the policy update for model-based policy search?

- Greedy methods: [Deisenroth & Rasmussen, 2011, Ng et al. 2001]
- "Bounded" methods: [Kupcsik Deisenroth, Peters & Neumann, 2013][Levine & Koltun 2014][Lioutikov, Peters, Neumann 2014]
  - Stochastic Policy
  - The model is only an approximation
    - Do not fully trust it!
  - The model is only good in the vicinity of the data-set
  - ➡ Stay close to the data!
  - All these methods use some sort of KL-bound

$$\arg\max_{\pi} \mathbb{E}_{\hat{P},\pi}\left[\sum_{t=1}^{T} r(\boldsymbol{s}_t, \boldsymbol{a}_t)\right], \quad \text{s.t.: } \mathrm{KL}(\pi||q) \leq \epsilon$$

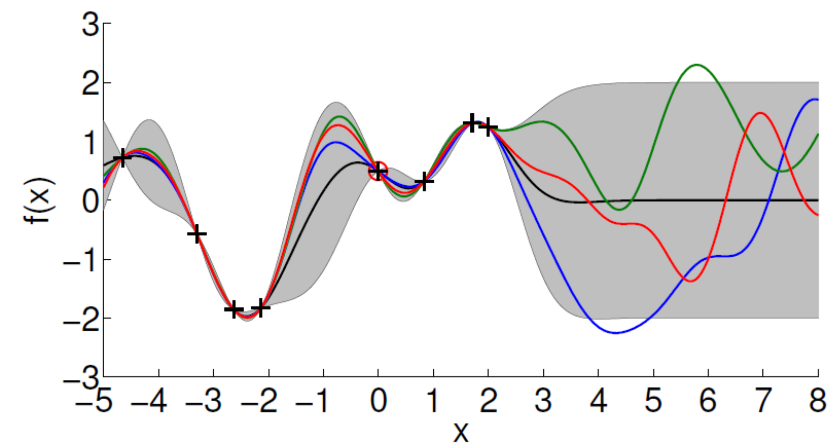
  - ➡ Ideas from model-free PS directly transfer
  - Exploration: Step-size of the policy update is bounded

GP-REPS [Kupcsik, Deisenroth, Peters & Neumann, 2013]

Model-based extension used for contextual policy search

Model Learning:

- Gaussian Processes for learning the dynamics of robot and environment



Internal Stimulation:

- Sampling trajectories from $\mathcal{P}(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})$ following policy $\pi(\boldsymbol{s};\boldsymbol{\theta})$
- Generate a high number of trajectories for different parameter vectors $\boldsymbol{\theta}$ and context vectors $\boldsymbol{x}$

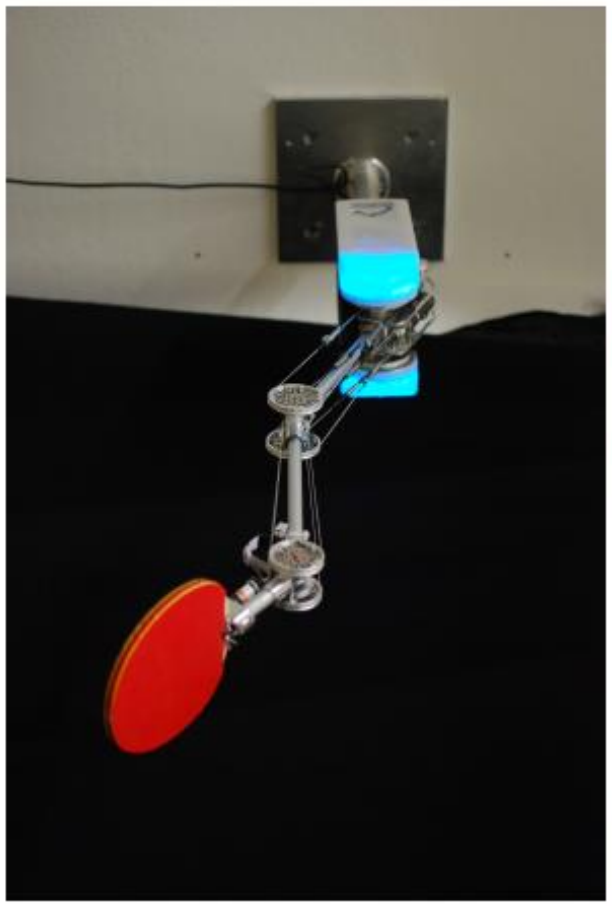
Policy Update:

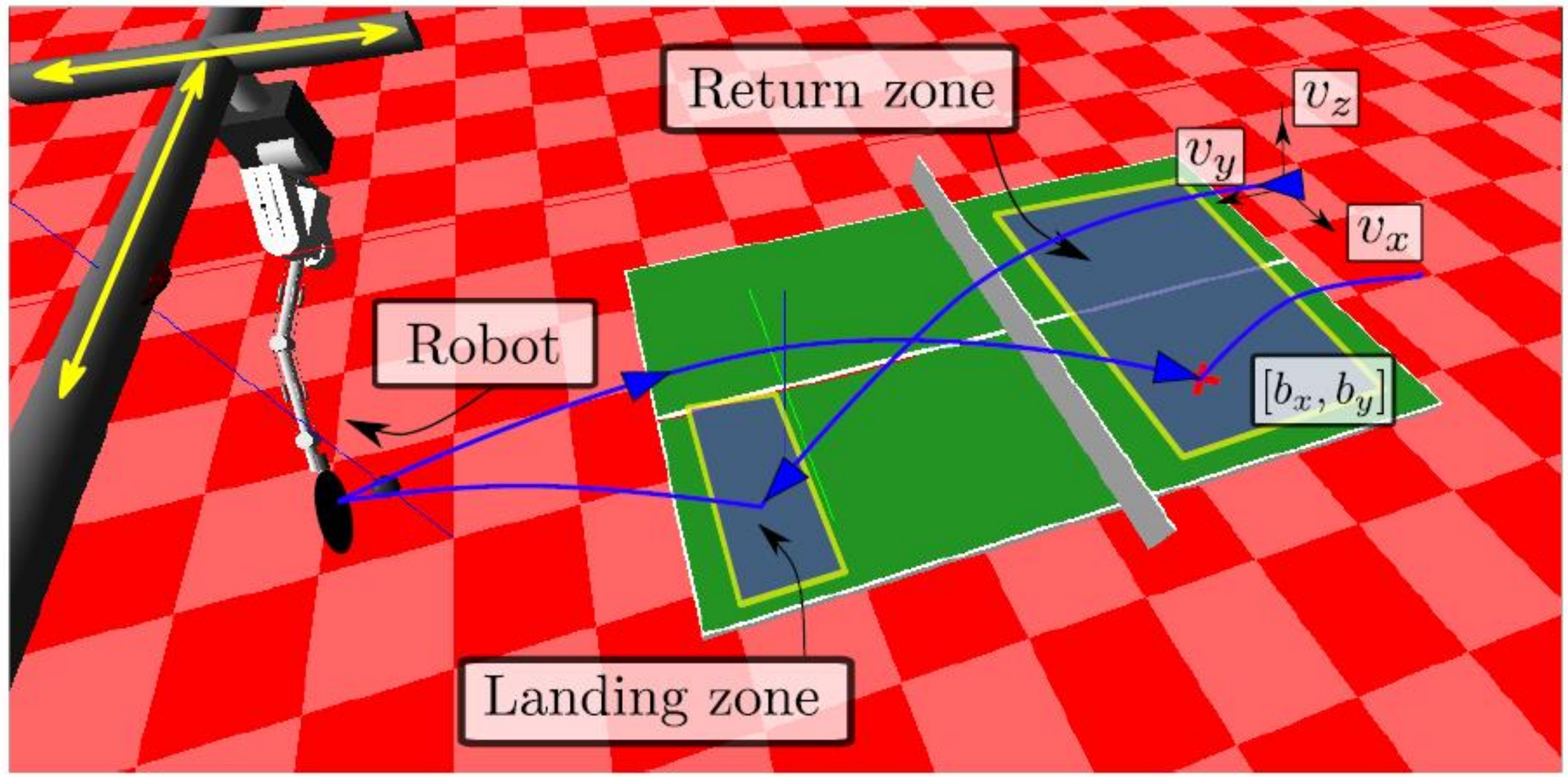
- Use contextual REPS on the artificial samples
- Trajectories will stay in the area where we have dynamics data

$$\arg\max_{\pi} \quad \mathbb{E}_{\hat{P},\pi}\left[R_{\boldsymbol{x}\boldsymbol{\theta}}\right],$$
$$\text{s.t.:} \quad \mathrm{KL}\big(\pi(\boldsymbol{\theta}|\boldsymbol{x})||q(\boldsymbol{\theta}|\boldsymbol{x})\big) \leq \epsilon$$

Table tennis experiment

[Kupcsik, Deisenroth, Peters & Neumann et al. 2015]





19 Policy Parameters (DMPs)

5 context variables (initial ball velocities, desired target location)

[Kupscik, Neumann et al, Artificial Intelligence, 2013]

Table tennis experiments

Learn GP models for:

- Ball contact on landing zone
- Ball trajectory from contact
- Racket trajectory from policy parameters
- Detect contact with racket (yes/no)
- If contact, predict return position on opponents field

A lot of prior knowledge is needed to decompose this MDP into simpler models

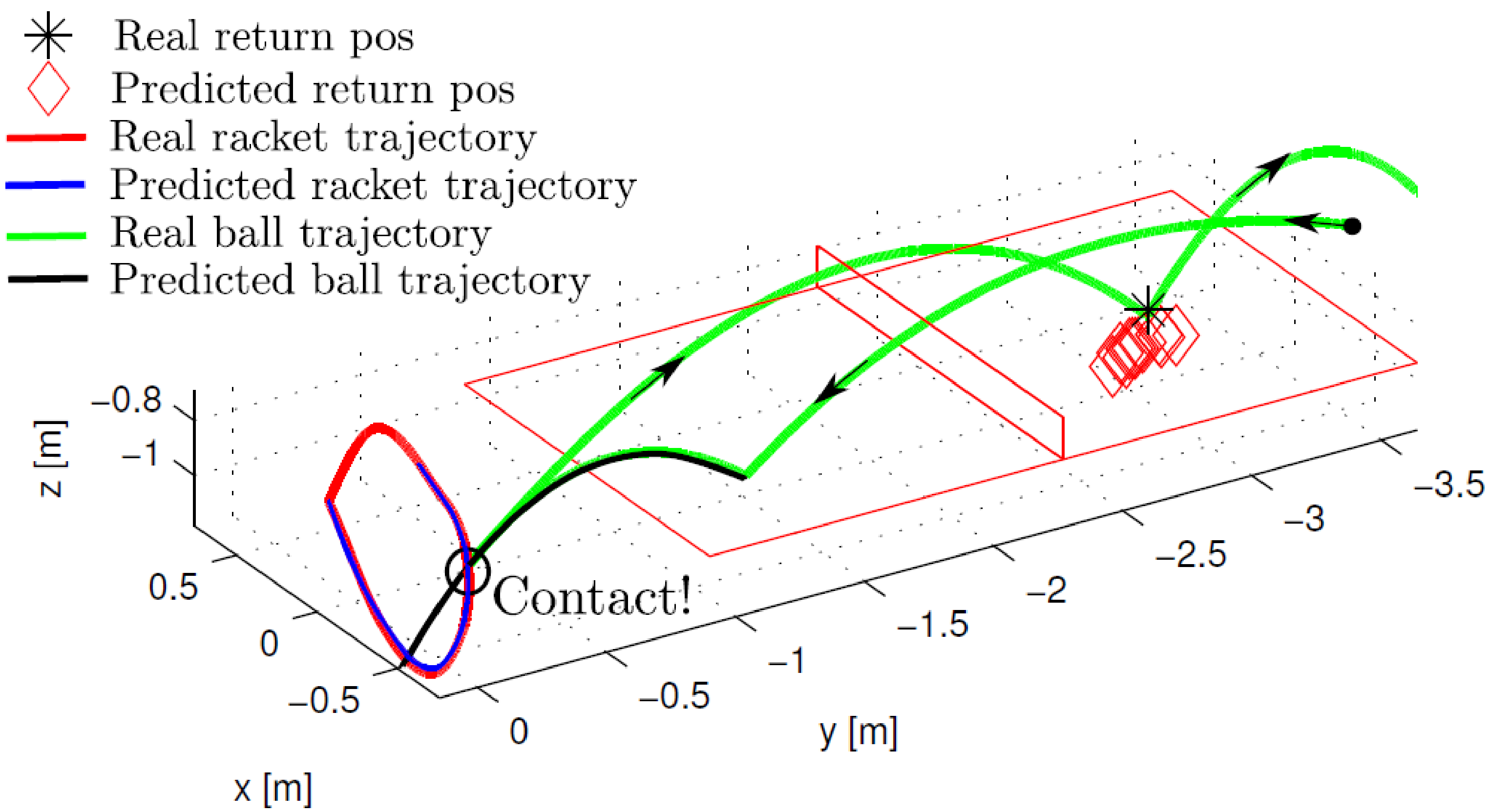
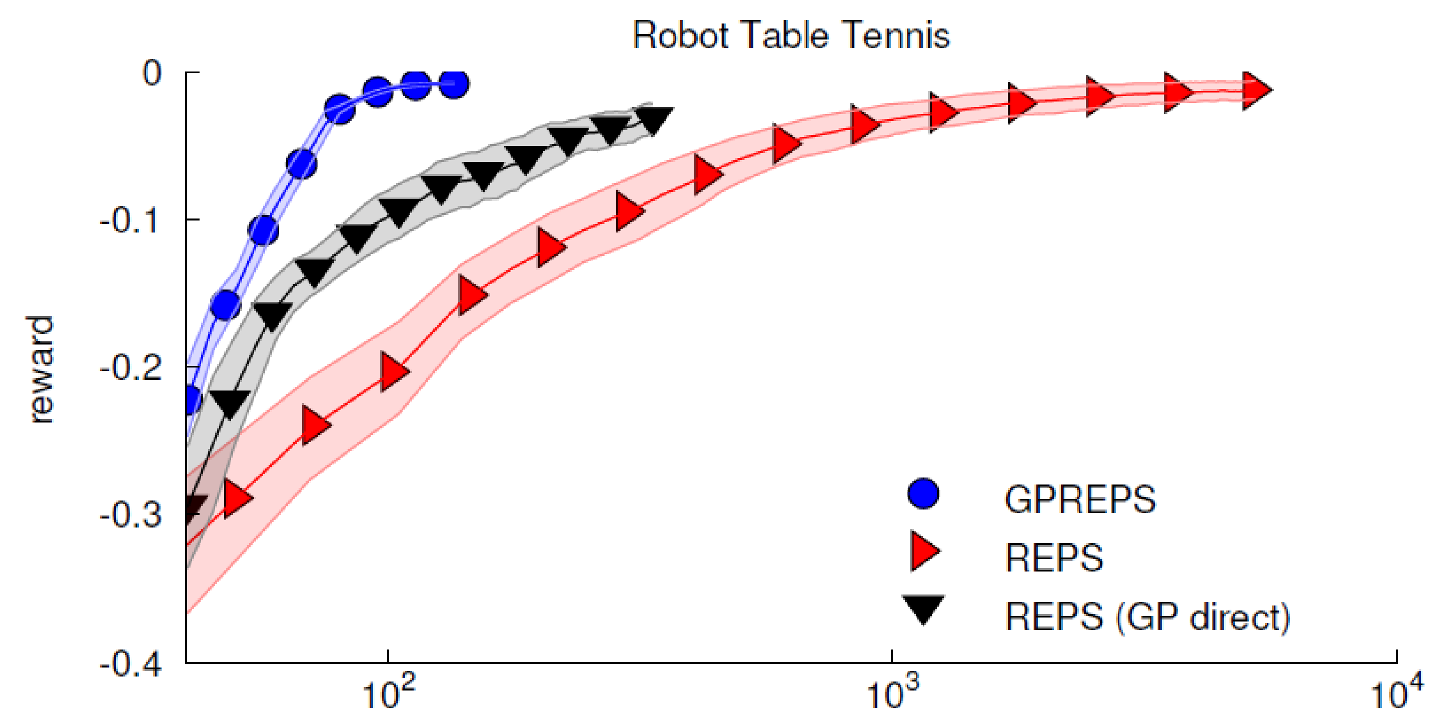


Table tennis experiments

REPS with learned forward models

- Complex behavior can be learned within 100 episodes
- 2 order of magnitudes faster than model-free REPS



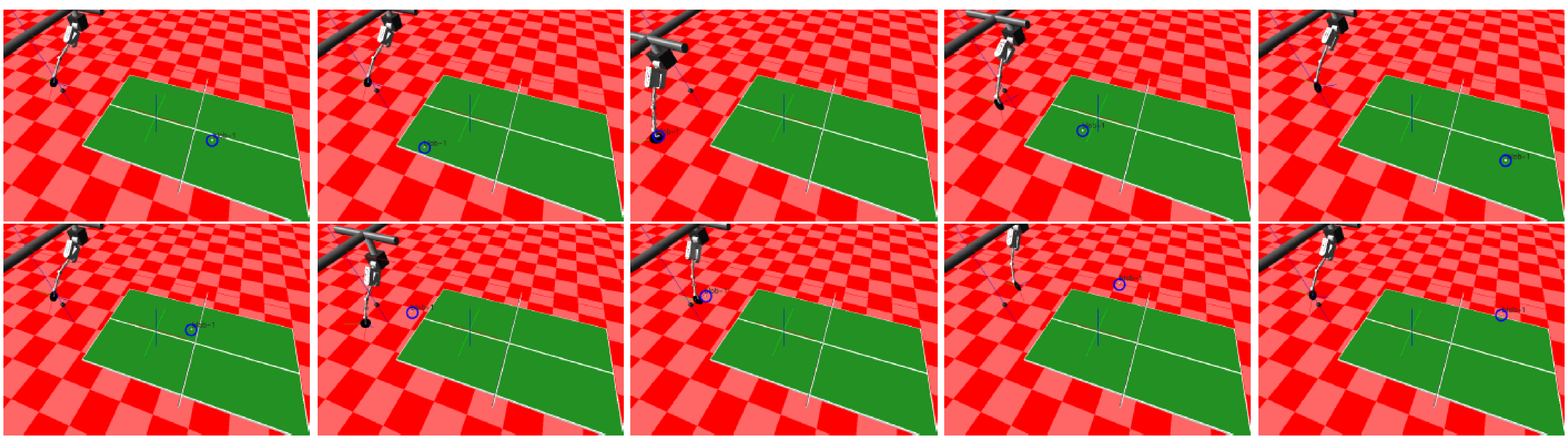
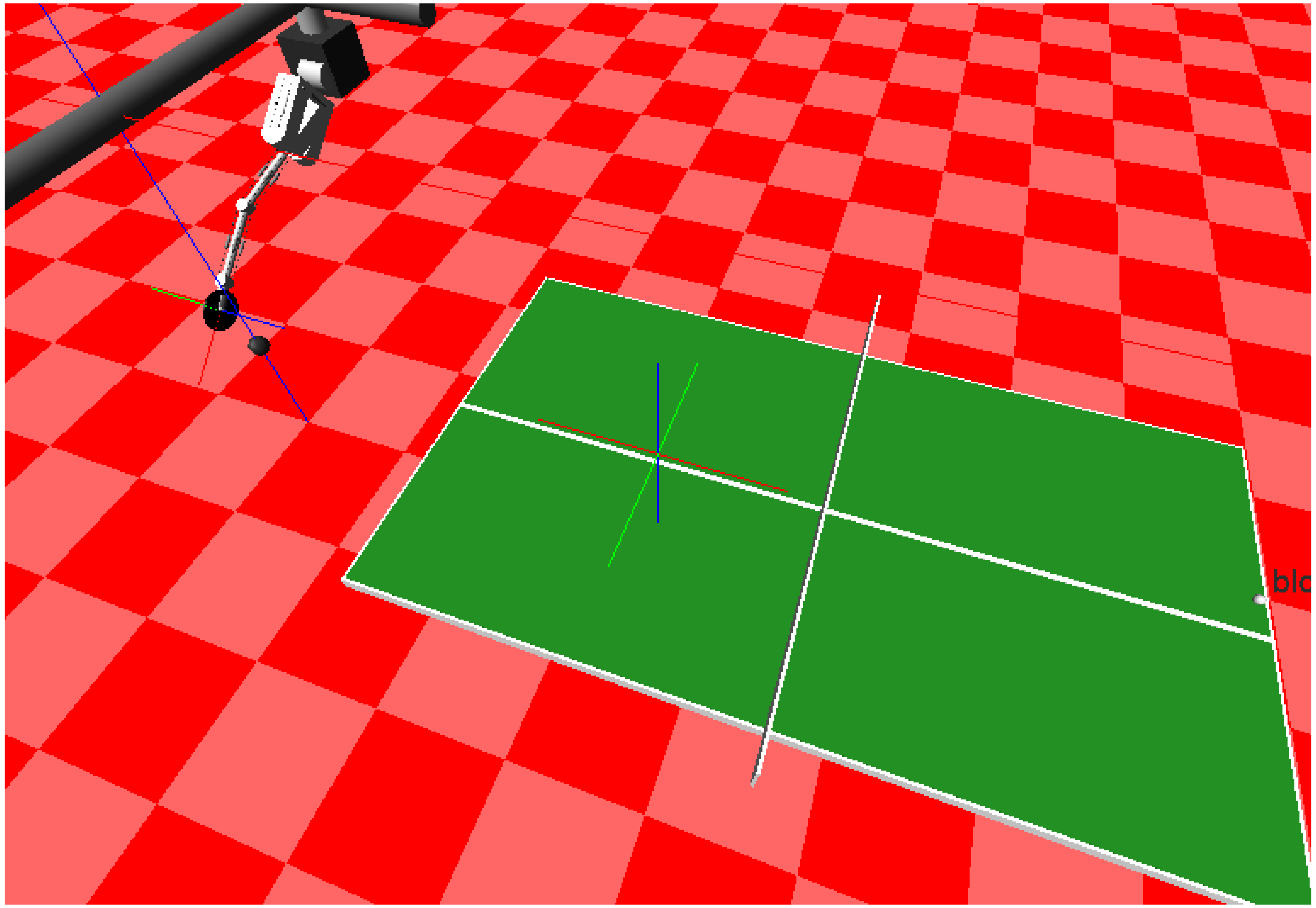


Table tennis experiments

Illustration: 2 shots for different contexts



- Works well for trajectory generators (small number of parameters)
- For more complex policies we need a step-based policy update!

Step-based REPS [Peters et al., 2010]

We can also formulate the REPS with states and actions

- Original formulation can be found in [Peters et al., 2010]

2 different formulations:

- Infinite Horizon: Average reward formulation using a stationary state distribution
  - Original REPS paper [Peters et al., 2010]
  - Non-parametric REPS [Von Hoof, Peters & Neumann, 2015]
- Finite Horizon: Accumulated reward formulation using trajectories
  - Guided policy search with trajectory optimization [Levine & Koltun, 2014], [Levine & Abeel, 2014]
  - Time-Indexed REPS [Daniel Neumann, Kroemer & Peters, 2013][Lioutikov, Paraschos, Peters & Neumann, 2014]

Infinite Horizon Formulation

Bound the change in the resulting state action distribution $\mu^\pi(\boldsymbol{s})\pi(\boldsymbol{a}|\boldsymbol{s})$

$$\max_\pi \iint \mu^\pi(\boldsymbol{s})\pi(\boldsymbol{a}|\boldsymbol{s})r(\boldsymbol{s},\boldsymbol{a})d\boldsymbol{s}d\boldsymbol{a}$$

Maximize average reward

$$\text{s.t.:} \quad \epsilon \geq \mathrm{KL}\big(\mu^\pi(\boldsymbol{s})\pi(\boldsymbol{a}|\boldsymbol{s})||q(\boldsymbol{s},\boldsymbol{a})\big)$$

KL should be bounded to old state action distribution

$$1 = \iint \pi(\boldsymbol{a}|\boldsymbol{s})\mu^\pi(\boldsymbol{s})d\boldsymbol{s}d\boldsymbol{a}$$

It's a distribution

$$\forall \boldsymbol{s}', \mu^\pi(\boldsymbol{s}') = \iint \mu^\pi(\boldsymbol{s})\pi(\boldsymbol{a}|\boldsymbol{s})\mathcal{P}(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})d\boldsymbol{s}d\boldsymbol{a}$$

State distribution needs to be consistent with policy and learned dynamics model

Infinite Horizon Formulation

Closed form solution:

$$\mu^{\pi}(\boldsymbol{s})\pi(\boldsymbol{a}|\boldsymbol{s}) \propto q(\boldsymbol{s},\boldsymbol{a}) \exp\left(\frac{r(\boldsymbol{s},\boldsymbol{a}) + \mathbb{E}_{\hat{\mathcal{P}}}[V(\boldsymbol{s}')|\boldsymbol{s},\boldsymbol{a}] - V(\boldsymbol{s})}{\eta}\right)$$

- We automatically get a softmax over the advantage function

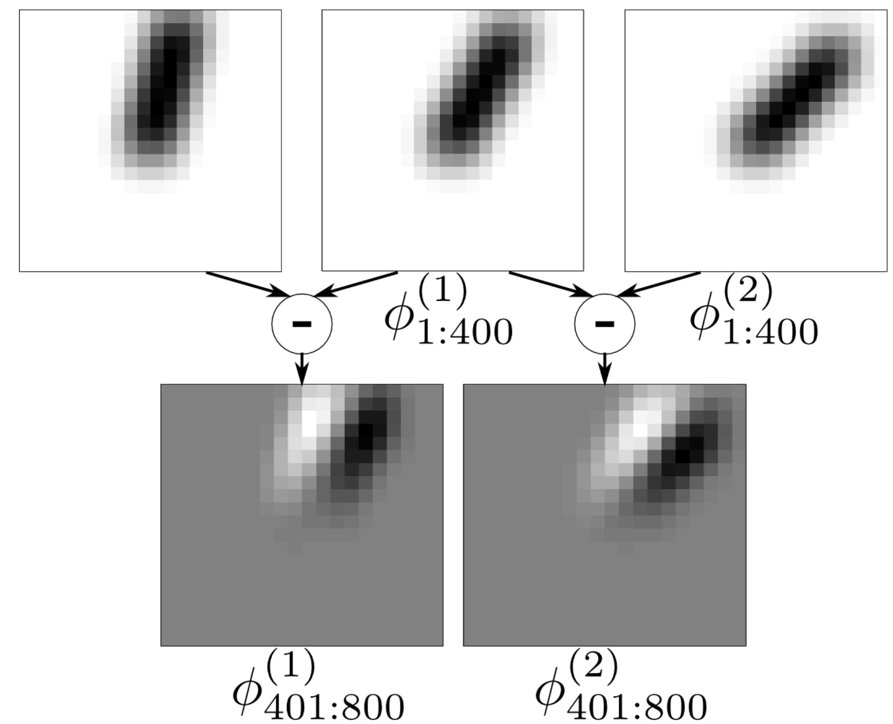
$$A(\boldsymbol{s},\boldsymbol{a}) = r(\boldsymbol{s},\boldsymbol{a}) + \mathbb{E}_{\hat{\mathcal{P}}}[V(\boldsymbol{s}')|\boldsymbol{s},\boldsymbol{a}] - V(\boldsymbol{s})$$

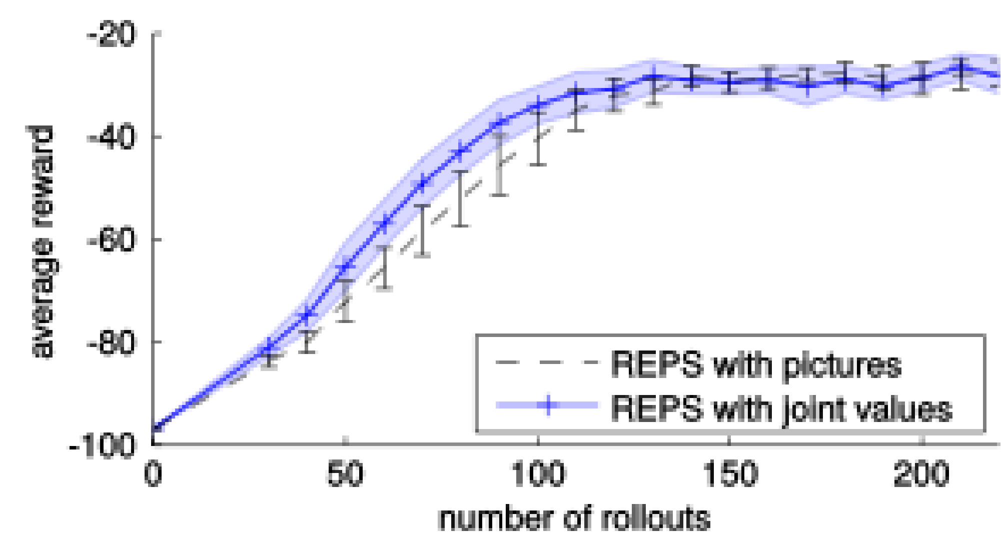
- *V(s)*… Lagrangian multiplier, resembles a value function
  - Linear function approximation [Peters et al. 2010]: $V(\boldsymbol{s}) = \phi(\boldsymbol{s})^T \boldsymbol{v}$
  - Put in a reproducing kernel Hilbert space (RKHS): [Von Hoof, Peters, Neumann 2015] $$V(\boldsymbol{s}) = \sum_{\boldsymbol{s}_i} \alpha_i k(\boldsymbol{s}_i, \boldsymbol{s})$$
- The model is needed to evaluate expectation $\mathbb{E}_{\hat{\mathcal{P}}}[V(\boldsymbol{s}')|\boldsymbol{s},\boldsymbol{a}]$
  - Either approximated by single sample outcomes [Peters et al., 2010, Daniel , Neumann & Peters, 2013]
  - or conditional operators in an RKHS [Von Hoof, Peters & Neumann, 2015]

Image-based pendulum swing-up

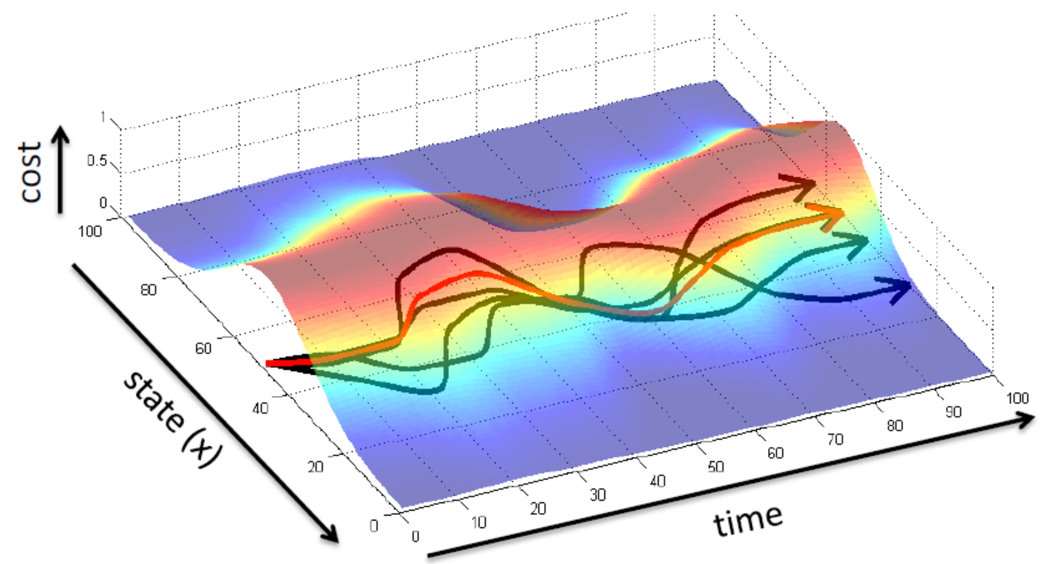
Learn pendulum swing-up based on image data [Von Hoof, Neumann & Peters, 2015]

- Policy is a GP defined on images
- Policy is obtained via weighted ML





Trajectory-based formulation



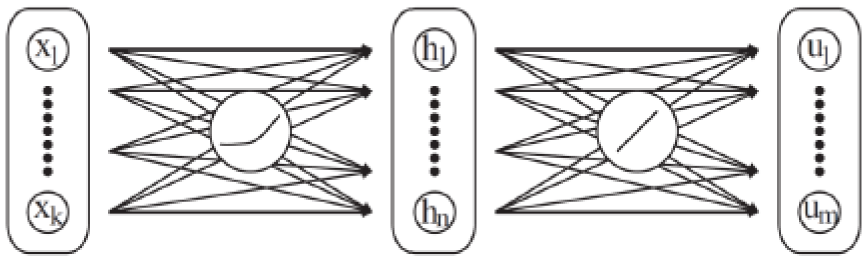
Levine et. al

Guided Policy Search via Trajectory Optimization [Levine & Koltun, 2014]

- Use trajectory optimization to learn local policies
- Policy is a time-varying stochastic feedback controller
- Time-varying linear model is learned
- Bounded policy update critical for the stability of the algorithm

Use learned local policies to train global, complex policy

- Deep Neural Nets
- **"Guidance"**:
  - Local policy might have more information on the current situation than the global one
  - Joint values versus camera image [Levine 2015]
  - Global policy learns to infer which situation we are in



Levine et. al

Bounded Trajectory Optimization

Bound the change in the resulting trajectory distribution $p^{\pi}(\boldsymbol{\tau})$

$$\max_{\pi} \quad \int p^{\pi}(\tau) R(\boldsymbol{\tau}) d\boldsymbol{\tau}$$

Maximize average reward

$$\text{s.t.:} \quad \epsilon \geq \mathrm{KL}\big(p^{\pi}(\boldsymbol{\tau}) || q(\boldsymbol{\tau})\big)$$

KL should be bounded to old trajectory distribution

$$\forall t, \quad 1 = \int \pi_t(\boldsymbol{a}|\boldsymbol{s}) d\boldsymbol{a}$$

It's a distribution

Bounded Trajectory Optimization

Plugging in the factorization of the trajectory distribution:

$$\max_{\pi} \quad \iint \mu_t^{\pi}(\boldsymbol{s})\pi_t(\boldsymbol{a}|\boldsymbol{s})r_t(\boldsymbol{s},\boldsymbol{a})d\boldsymbol{s}d\boldsymbol{a}$$

Maximize average reward

$$\text{s.t.:} \quad {\color{red}\forall t} : \epsilon \geq \mathbb{E}_{\mu_t^{\pi}} \left[\text{KL}\big(\pi_t(\boldsymbol{a}|\boldsymbol{s})||q_t(\boldsymbol{a}|\boldsymbol{s})\big)\right]$$

KL on the policies should be bounded at each time step

$$\forall t \forall \boldsymbol{s} : 1 = \int \pi_t(\boldsymbol{a}|\boldsymbol{s})d\boldsymbol{a}$$

It's a distribution

$$\forall \boldsymbol{s}' \forall t : \mu_{t+1}^{\pi}(\boldsymbol{s}') = \iint \mu_t^{\pi}(\boldsymbol{s})\pi_t(\boldsymbol{a}|\boldsymbol{s})\mathcal{P}_t(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a})d\boldsymbol{s}d\boldsymbol{a}$$

Time-dependent state distributions need to be consistent

$$\forall \boldsymbol{s} : \mu_1^{\pi}(\boldsymbol{s}) = \mu_1(\boldsymbol{s}), \forall \boldsymbol{s}$$

Initial distribution is given

Infinite Horizon Formulation

Closed form solution:

$$\pi_t(\boldsymbol{a}|\boldsymbol{s}) \propto q_t(\boldsymbol{a}|\boldsymbol{s}) \exp\left(\frac{r_t(\boldsymbol{s},\boldsymbol{a}) + \mathbb{E}_{\hat{\mathcal{P}}}[V_{t+1}(\boldsymbol{s}')|\boldsymbol{s},\boldsymbol{a}]}{\eta_t}\right)$$

- *V(s)*… Lagrangian multiplier,
  - can be computed by dynamic programming

$$V_t(\boldsymbol{s}) = \log \int q(\boldsymbol{a}|\boldsymbol{s}) \exp\left(\frac{r(\boldsymbol{s},\boldsymbol{a}) + \mathbb{E}[V_{t+1}(\boldsymbol{s}')]}{\eta_t}\right) d\boldsymbol{a}$$

- Time-dependent temperature η_t
- Linear systems, quadratic costs and Gaussian noise:
  - Standard LQR equations, solved by dynamic programming
  - The policy is a (stochastic) linear feed back controller

$$\pi_t(\boldsymbol{a}|\boldsymbol{s}) = \mathcal{N}(\boldsymbol{a}|\boldsymbol{K}_t\boldsymbol{s} + \boldsymbol{k}_t, \boldsymbol{\Sigma}_t)$$

  - Implements exploration
  - Similar to iLQG [Todorov & Li, 2005], but more stable due to KL-bound

Time-varying linear models

Linear models:

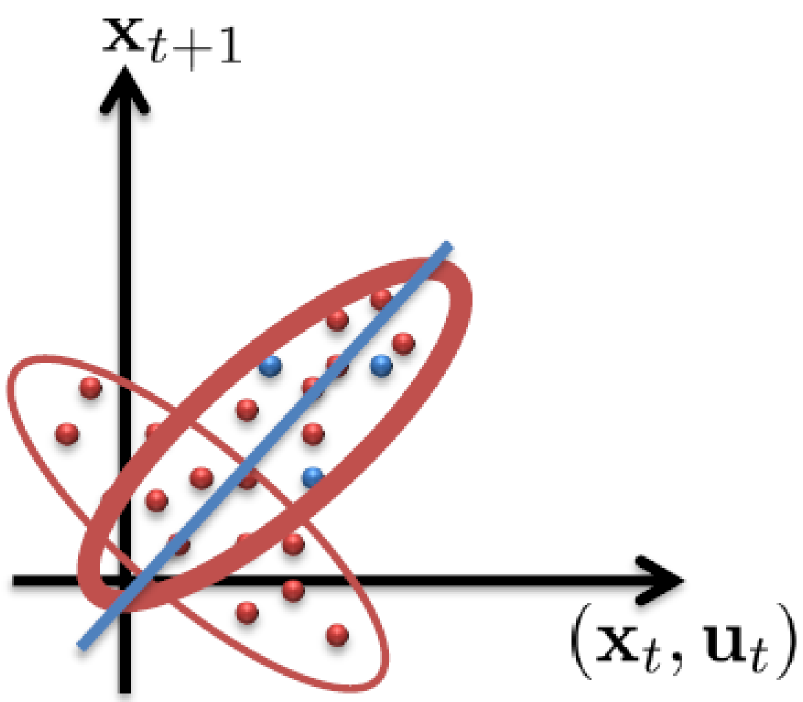
- Generalize well locally
- Scale well

Time-varying:

- Enforces locality
- At the same time step, the robot will be in similar states in different trials

Learning time-varying linear models:

- Learn a GMM of linear models
- Fit an own model for each time step
- Use GMM as prior

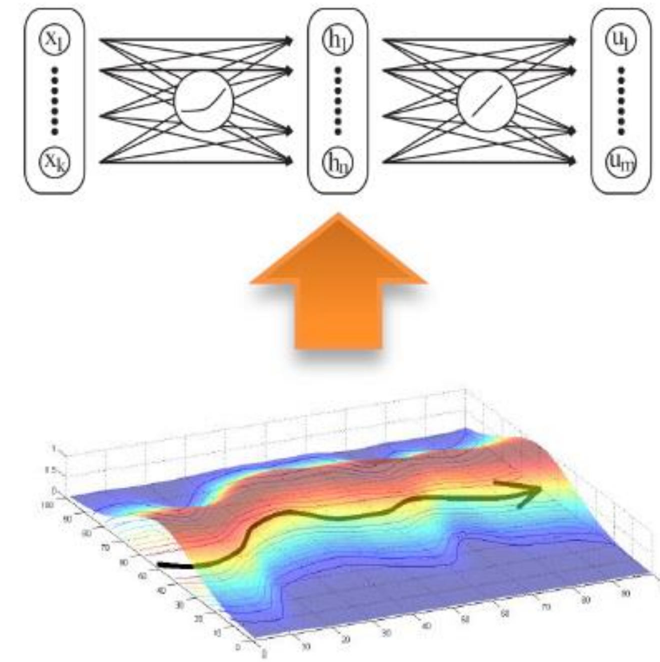


Levine et. al

Constrained Guided Policy Search [Levine 2014]

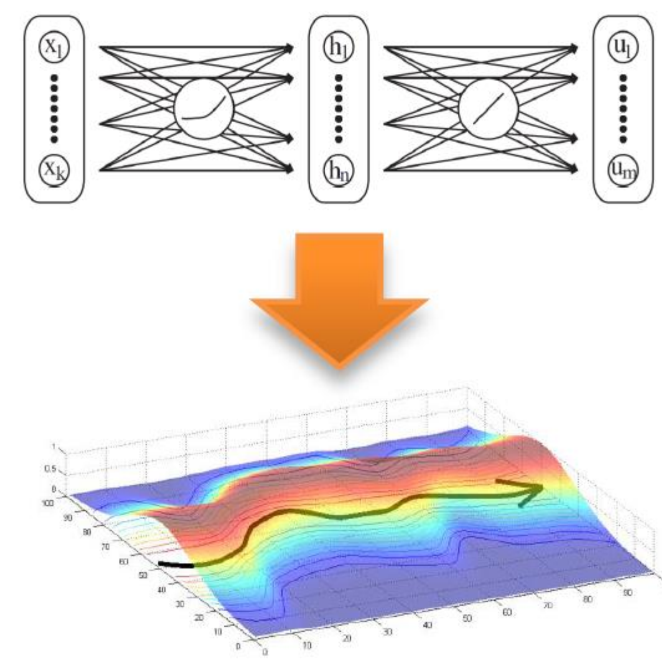
Train Deep Neural Net:

- Supervised learning: reproduce the optimized trajectories
- Linearization of the neural net should be close to linear feedback controller
- Can train several thousand parameters



Trajectory optimization:

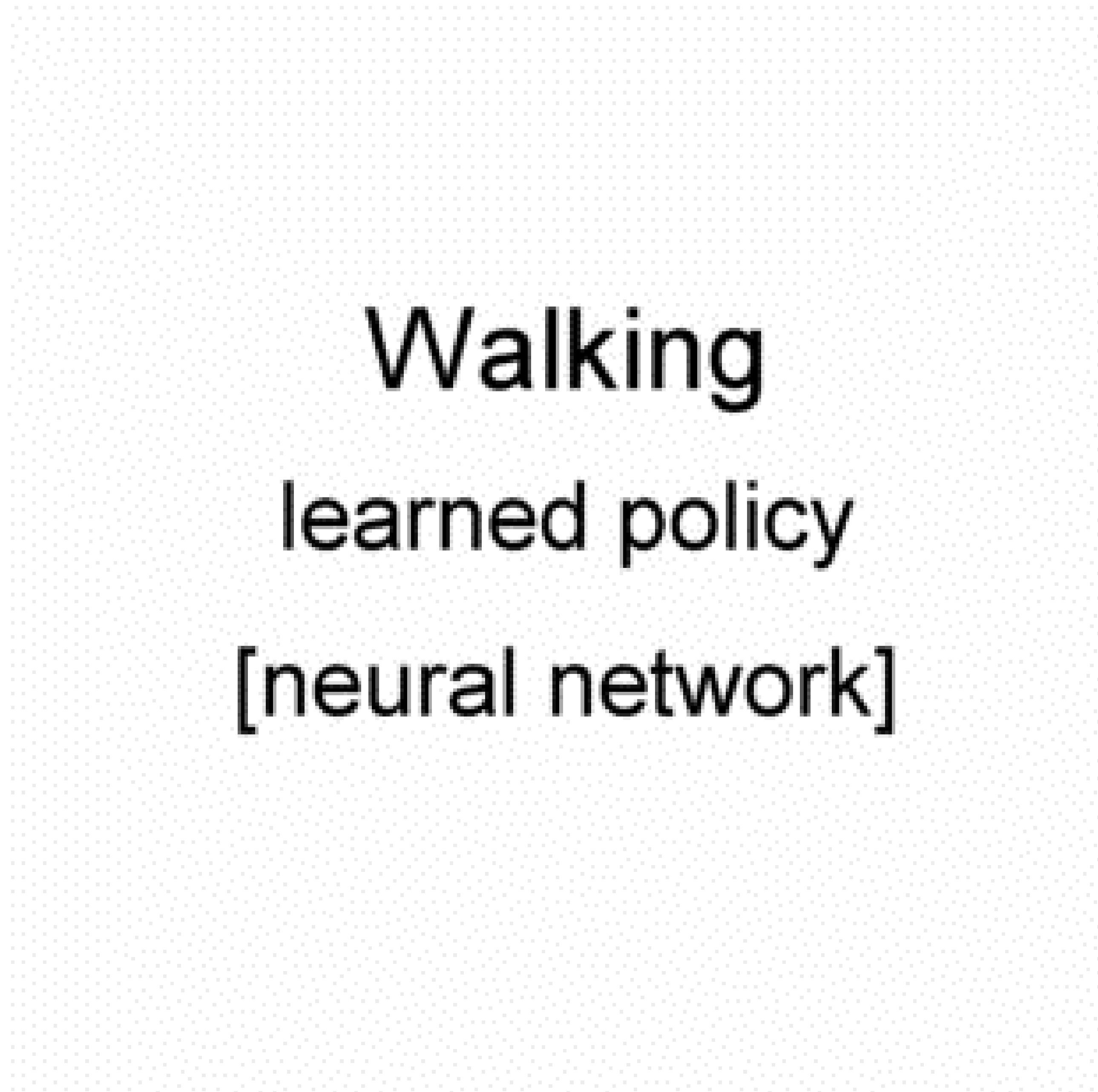
- Trajectories should stay close to trajectories generated by neural net
- No time dependence in the neural net



Simulated Results

Learning walking gaits [Levine & Koltun, 2014]:

- Simulator: Mojoco
- Planar walking robot



Real Robot Results

Learning different manipulation tasks [Levine 2015]:

Outlook

Learning from high-dimensional sensory data

- Tactile and vision data
- Deep Learning
- Kernel-based methods

Hierarchical Policy Search

- Identify set of re-useable skills
- Learn to select, adapt, sequence and combine these skills
- Deep hierarchical policy search?

Incorporate human feedback

- Inverse RL and Preference Learning
- Autonomous learning from imitation

POMDPs and Multi-Agent Policy Search

Conclusion

Policy Search Methods have made a tremendous development

- Model free methods can learn trajectory-based policies for complex skills
  - Trajectory-based representations provide an compact representation of a skill but lack flexibility
  - Step-based vs episode-based formulation
  - Different optimization methods with different policy metrics
- Complex policies with thousands of parameters can be learned with model-based methods
  - But might be less appropriate for execution on a real robot

Robot-RL is still a challenging problem

- Learning efficient exploration policies is a major challenge
  - Exploration-Exploitation tradeoff can be controlled by bounding the relative entropy
  - Bounded policy updates are useful for model-free and model-based methods
- We can solve mainly monolithic problems
  - Hierarchical policy search methods should help