Policy Search for Robotics and Multi-Agent Systems

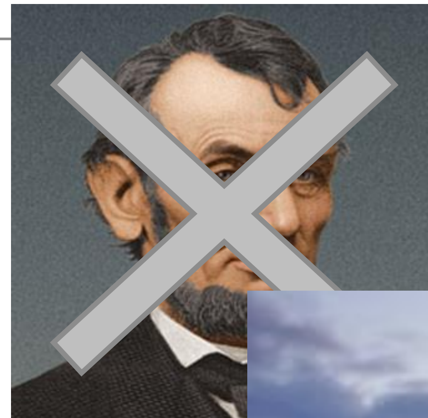
Gerhard Neumann, University of Lincoln

UNIVERSITY OF LINCOLN

Horizon 2020

Some geography…

➡Lincoln?



Motivation

In the next few years, we will see a dramatic increase of (multi-)robot applications

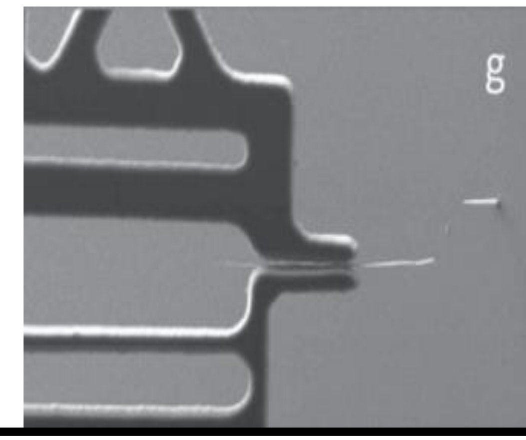
Today:

Indus

http://www.

Household

Tomorrow:



gerous Env.

ackkanter.com/

Programming such tasks seems to be infeasable.

Can a robot learn such tasks by trial and error?

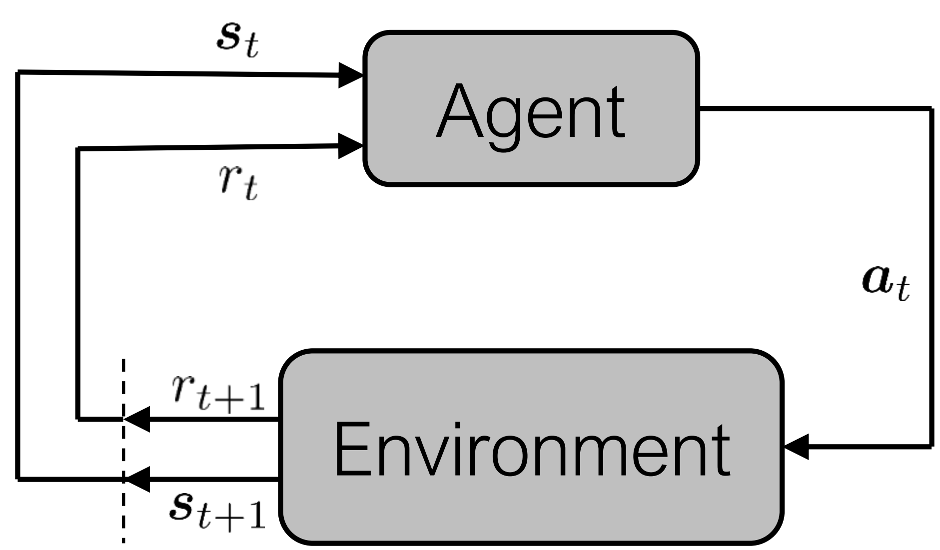
Household

Agriculture

Transportation

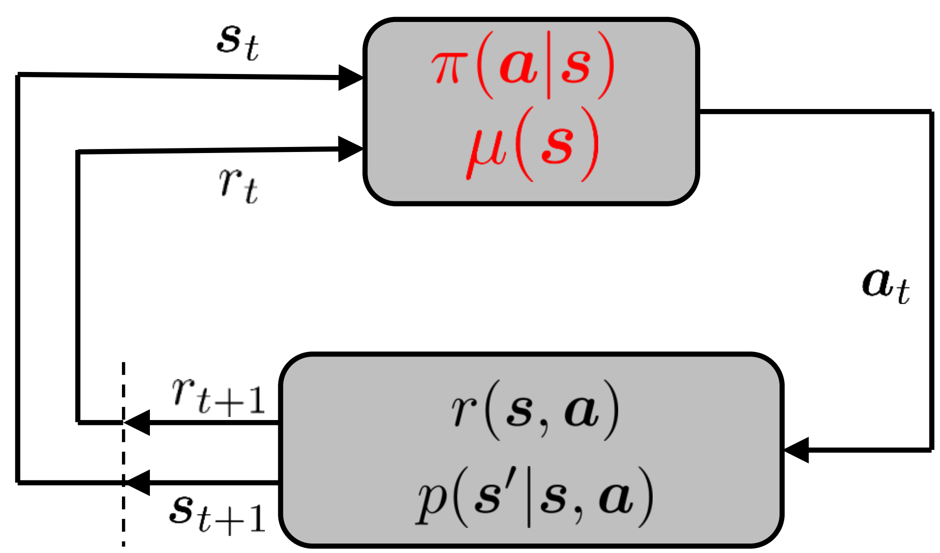
Reinforcement Learning (RL)

Markov Decision Processes (MDPs):



Reinforcement Learning (RL)

Markov Decision Processes (MDPs):



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

transition dynamics
$p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$

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

- implicit exploration

Deterministic Policy $\mu(\boldsymbol{s})$

- explicit exploration needed in addition

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

Reinforcement Learning

Objective: Find policy that maximizes long term reward J_π

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

Infinite Horizon MDP:

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

- Discount factor γ

Tasks:

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

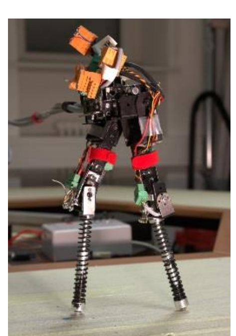
Finite Horizon MDP:

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

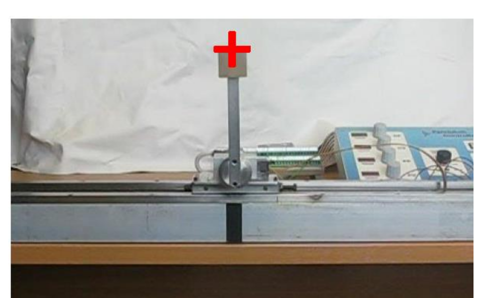
Tasks:

- 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.

Reinforcement Learning

Important Functions:

- **V-Function:** Quality of state ***s*** when following policy π

Infinite Horizon MDP:

$$V^{\pi}(\boldsymbol{s}) = \mathbb{E}_{\pi}\left[\sum_{h=0}^{\infty} \gamma^h r_h(\boldsymbol{s}_h, \boldsymbol{a}_h) \middle| \boldsymbol{s}_t = \boldsymbol{s}\right]$$

Finite Horizon MDP:

$$V_t^{\pi}(\boldsymbol{s}) = \mathbb{E}_{\pi}\left[\sum_{h=t}^{T} r_h(\boldsymbol{s}_h, \boldsymbol{a}_h) \middle| \boldsymbol{s}_t = \boldsymbol{s}\right]$$

- **Q-Function:** Quality of state ***s*** when taking action ***a*** and following policy π afterwards

Infinite Horizon MDP:

$$Q^{\pi}(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_{\pi}\left[\sum_{h=0}^{\infty} \gamma^h r_h(\boldsymbol{s}_h, \boldsymbol{a}_h) \middle| \boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a}\right]$$

Finite Horizon MDP:

$$Q_t^{\pi}(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_{\pi}\left[\sum_{h=t}^{T} r_h(\boldsymbol{s}_h, \boldsymbol{a}_h) \middle| \boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a}\right]$$

Robot Reinforcement Learning

Challenges:

Dimensionality:

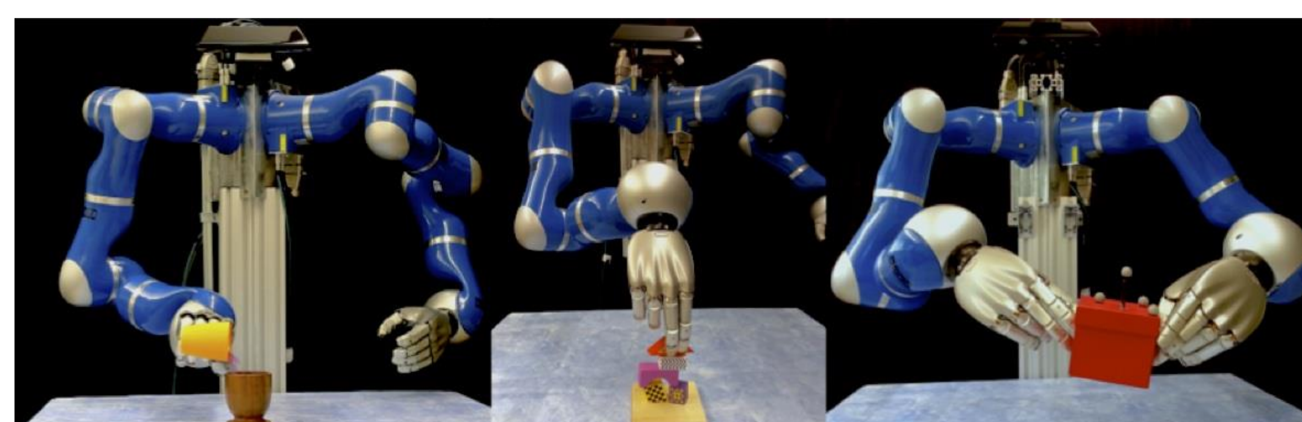
- High-dimensional continuous state and action space
- Huge variety of tasks

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

- 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

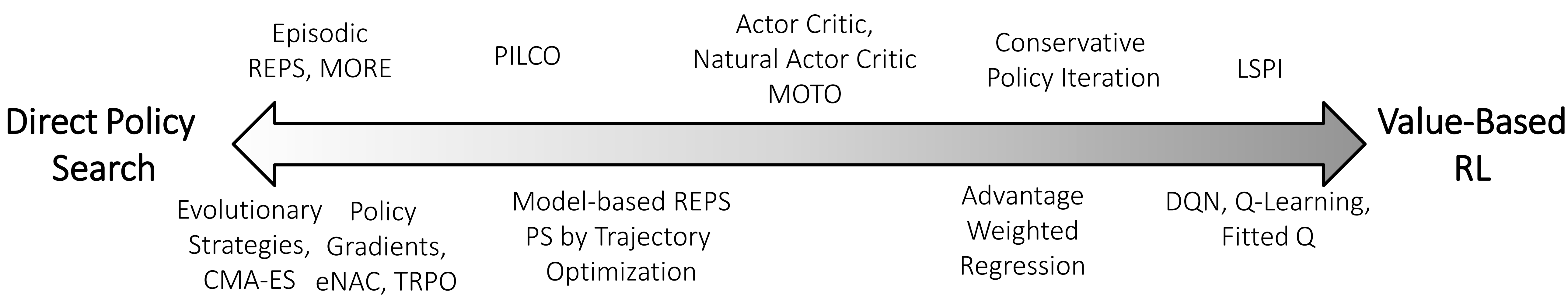
Correlated local exploration

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

- 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]

Outline

Taxonomy of Policy Search Algorithms

Model-Free Policy Search Methods

- Policy Gradients
- Natural Gradients
- Exact Information Geometric Updates
- Success Matching

Policy Search for Multi-Agent Systems

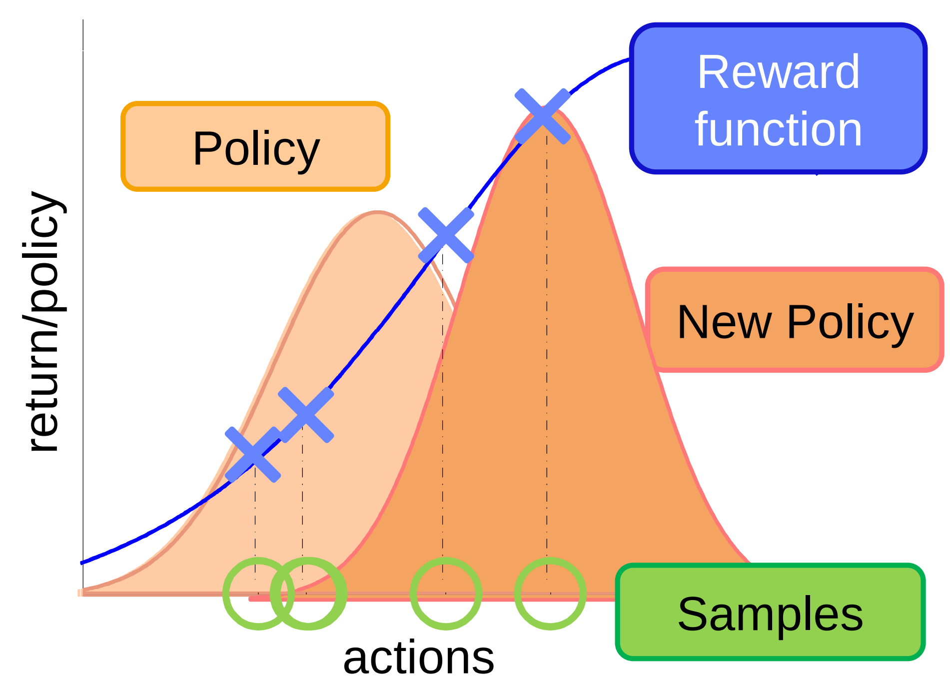
Policy Search Pseudo Algorithm

Three basic steps:

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

Evaluate: Assess quality of trajectory or actions

Update: Compute new policy π_{k+1}



Taxonomy of Policy Search Algorithms

Trajectory-based:

Use trajectories and parameters interchangeably

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

Explore: in parameter space at the beginning of an episode

- Search distribution $\pi(\boldsymbol{\theta};\boldsymbol{\omega})$
- $\boldsymbol{\omega} \ldots$ parameters of search distribution
- $\boldsymbol{a} = \mu(\boldsymbol{s};\boldsymbol{\theta})$... deterministic policy

Evaluate: quality of trajectories $\boldsymbol{\tau}_i$ by the returns $R^{[i]}$

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

Action-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 Policy Search Algorithms

Trajectory-based

Properties:

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

Structure-less optimization

➡ „Black-Box Optimizer“

Action-based

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 Policy Search Algorithms

Trajectory-based

Algorithms:

- Evolutionary Strategies
- PE-PG [Rückstiess, Sehnke, et al.2008]
- MORE [Abdolmaleki, et al.2015]
- Episodic REPS [Daniel, Neumann & Peters, 2012]
- PI2-CMA [Stulp & Sigaud, 2012]
- CMA-ES [Hansen et al., 2003]
- Natural Evolution Strategy [Wiestra, Schaul , Peters & Schmidhuber, 2008]
- Cross-Entropy Search [Mannor et al. 2004]

Action-based

Algorithms:

- Natural Actor Critic [Peters & Schaal 2003]
- Trust Region Policy Optimization [Schulman et al., 2015]
- MOTO [Akrour et al., 2016]
- Policy Gradient Theorem / GPOMDP [Baxter & Bartlett , 2001]
- 2nd Order Policy Gradients [Furmston & Barber 2011]
- Deterministic Policy Gradients [Silver, Lever et al, 2014]

Trajectory-based policy representations

Parametrized Trajectory Generators

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

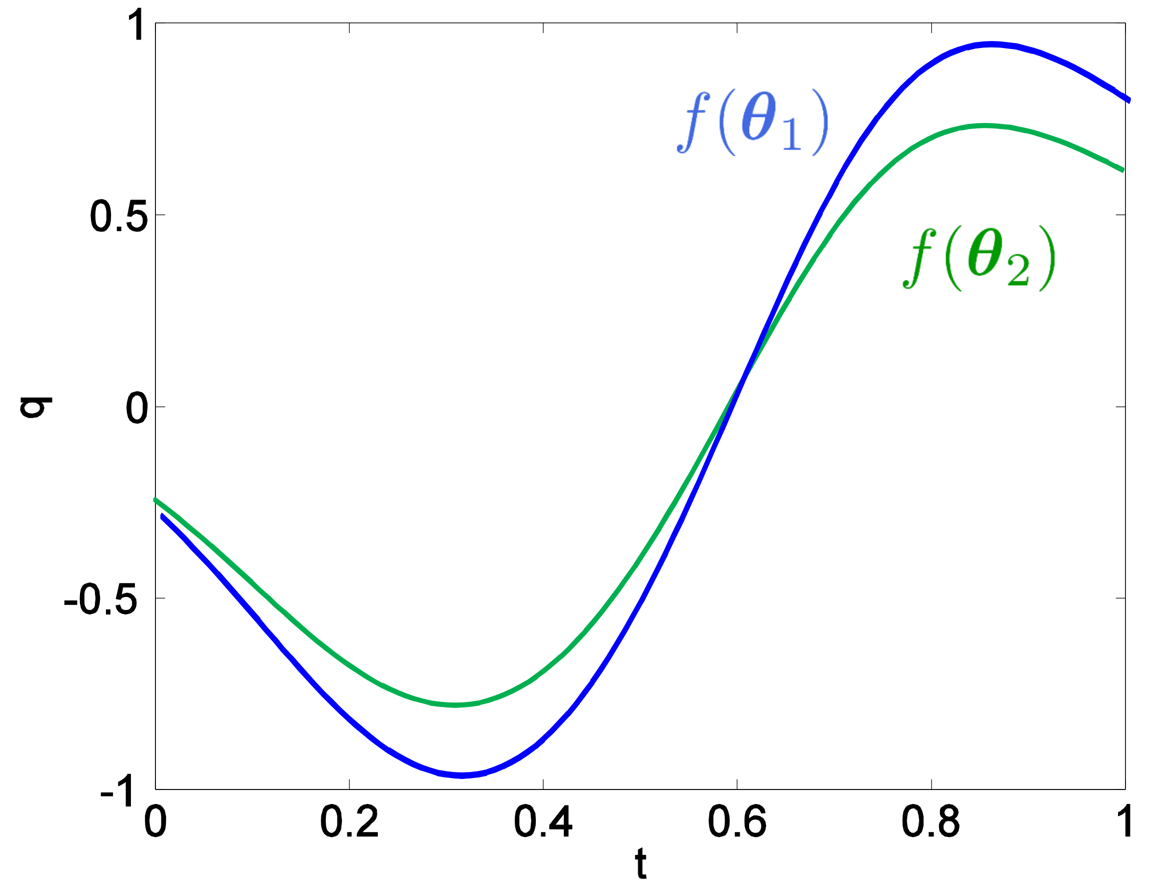
$$\boldsymbol{\tau}^* = \boldsymbol{q}^*_{1:T} = f(\boldsymbol{\theta})$$

- Compute controls $\boldsymbol{u}_t$ by the use of trajectory tracking controllers

✓ Low number of parameters

✓ Sample efficient to learn

× No sensory feedback



Examples:

- Splines, Bezier Curves [Miyamoto et al., Neural Networks 1996],[Kohl & Stone., ICRA 2004], ...
- Movement Primitives [Peters & Schaal, IROS 2006], [Kober & Peters., NIPS 2008], [Kormushev et al., IROS 2010], [Kober & Peters, IJCAI 2011] [Theodorou, Buchli & Schaal., JMLR 2010]

Action-based policy representations

Deep Neural Networks:

- Directly computes control output

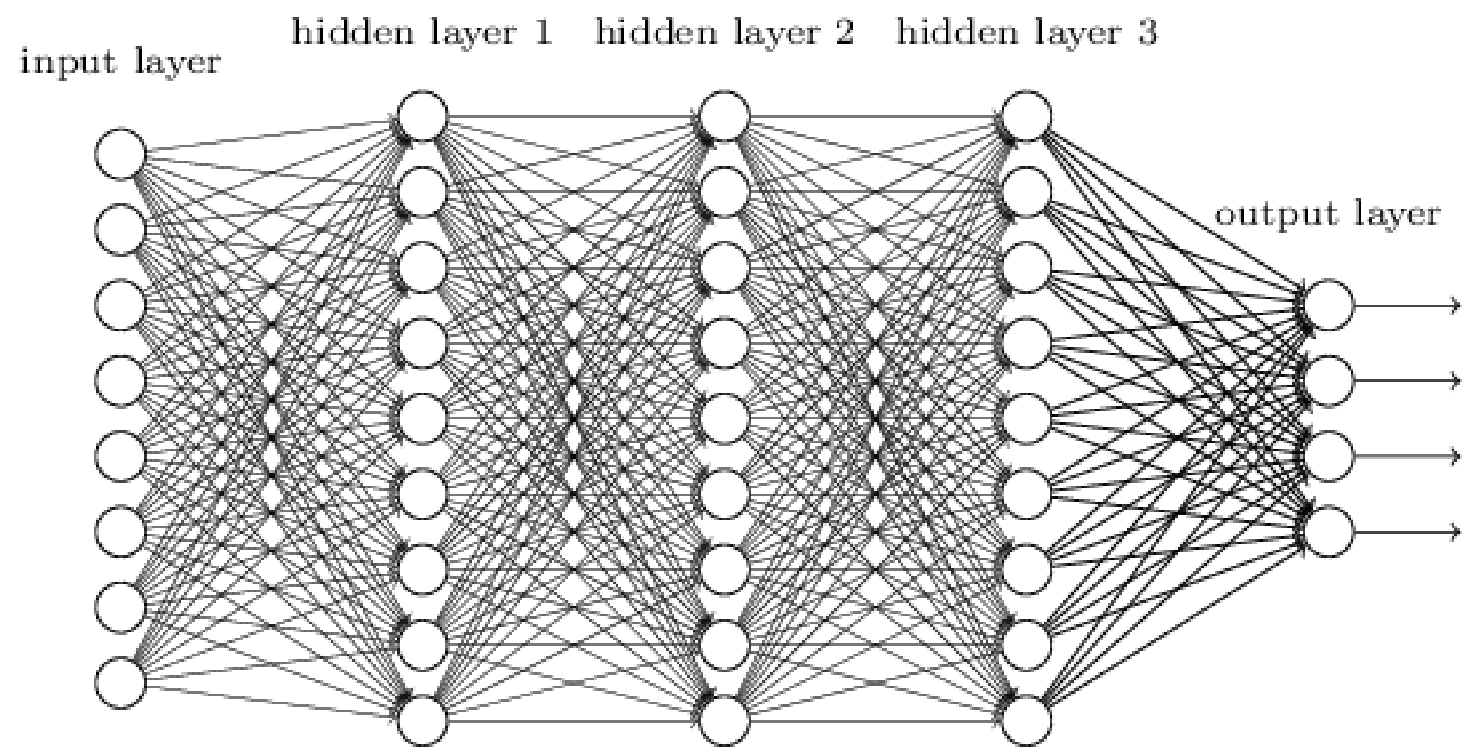
$$\pi(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{\theta}) = \mathcal{N}(\underbrace{\boldsymbol{\mu}(\boldsymbol{s})}_{\text{Deep NN}}, \boldsymbol{\Sigma})$$

✓ Less feature engineering

✓ Incorporate high-dimensional feedback (vision, tactile)

× Large number of parameters

× Needs a lot of training data



Examples: TRPO [Schulman 2015], DDPG [Silver 2015]

Other Representations:

- Linear Controllers [Williams et. al., 1992]
- RBF-Networks [Atkeson & Morimoto, NIPS 2002][Deisenroth & Rasmussen., ICML 2011]

Outline

Taxonomy of Policy Search Algorithms

Model-Free Policy Search Methods

- Policy Gradients
- Natural Gradients
- Exact Information Geometric Updates
- Success Matching

Policy Search for Multi-Agent Systems

Model-Free Policy Updates

Use samples

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

to directly update the policy

- Learn stochastic policies:

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

Parameter exploration　　Action exploration

- E.g. Gaussian policies:

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

  - Mean $\boldsymbol{\mu}$: location of the maximum
  - Covariance $\boldsymbol{\Sigma}$: which directions to explore (simplification: $\boldsymbol{\Sigma} = \mathrm{diag}(\boldsymbol{\sigma})$)
  - Update mean and covariance!

Model-Free Policy Updates

Different optimization methods ...

- Policy Gradients
- Natural Policy Gradients
- Exact Information-Geometric Updates
- Success Matching

... use different metrics to define step-size

- Euclidean distance
- Approximate KL
- Exact Information-KL
- Exact Moment-KL

Can be used for action-based and trajectory-based policy search

Policy Gradients

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-Ratio Policy Gradients

Trajectory-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:

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

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

$$\approx \sum_{i=1}^{N} \nabla_{\boldsymbol{\omega}} \log \pi(\boldsymbol{\theta}_i; \boldsymbol{\omega}) R^{[i]}$$

- **Policy gradients** with parameter-based exploration (PGPE) [Rückstiess 2008]

Likelihood-Ratio Policy Gradients

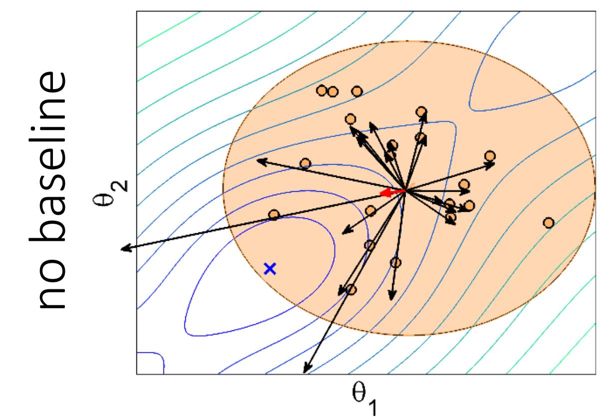
Problem: The likelihood-ratio gradient is a high variance estimator

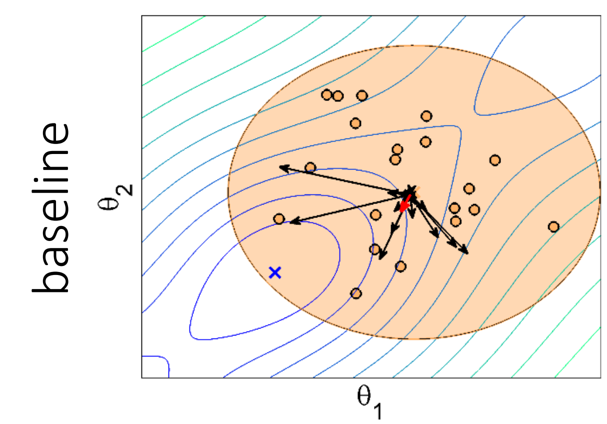
- Subtract a minimum variance-baseline
- High variance in the returns – use rewards to come

Baselines…

We can always **subtract a baseline *b*** from the returns…

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





Why?

- Subtracting a baseline can reduce the variance
- Its still unbiased…

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

Good baselines:

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

Action-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 $\boldsymbol{J_\theta}$ can be written as expectation over the trajectory distribution

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

Action-Based Likelihood Ratio Gradient

Using the log-ratio trick, we arrive at

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

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

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

Action-Based Policy Gradients

Plug it back in...

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

This algorithm is called the REINFORCE [Williams 1992]

Action-Based Policy Gradient Methods

The returns have a lot of variance

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

... as they are the sum over T random variables

There is less variance in the rewards to come:

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

- ... as we sum over less time steps

Using the rewards to come…

Simple Observation: Rewards in the past are not correlated with actions in the future

$$\mathbb{E}_{p(\boldsymbol{\tau})}\left[r_h \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_t|\boldsymbol{s}_t)\right] = 0, \forall h < t$$

This observation leads to the Policy Gradient Theorem [Sutton 1999]

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}}^{\mathrm{PG}} J &= \sum_{i=1}^{N} \sum_{t=1}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_t^{[i]}|\boldsymbol{s}_t^{[i]};\boldsymbol{\theta}) \left(\sum_{h=0}^{T} r_h^{[i]} \right) \\
&= \sum_{i=1}^{N} \sum_{t=1}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_t^{[i]}|\boldsymbol{s}_t^{[i]};\boldsymbol{\theta}) \left(\sum_{h=t}^{T} r_h^{[i]} \right) \\
&= \sum_{i=1}^{N} \sum_{t=1}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}_t^{[i]}|\boldsymbol{s}_t^{[i]};\boldsymbol{\theta}) Q_t^{[i]}
\end{aligned}$$

- This algorithm is also called GPOMDP [Baxter 2001]

Using the rewards to come

Essentially, the policy gradient theorem is equivalent to the following objective:

Finite Horizon MDP:

$$J_{\mathrm{PG}} = \sum_{t=1}^{T-1} \int p_t^{\pi_{\mathrm{old}}}(\boldsymbol{s}_t)\pi(\boldsymbol{a}_t|\boldsymbol{s}_t;\boldsymbol{\theta})Q_t^{\pi_{\mathrm{old}}}(\boldsymbol{s}_t,\boldsymbol{a}_t)d\boldsymbol{s}_t d\boldsymbol{a}_t$$

Infinite Horizon MDP:

$$J_{\mathrm{PG}} = \int p^{\pi_{\mathrm{old}}}(\boldsymbol{s})\pi(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{\theta})Q^{\pi_{\mathrm{old}}}(\boldsymbol{s},\boldsymbol{a})d\boldsymbol{s}d\boldsymbol{a}$$

- $p^{\pi_{\mathrm{old}}}(\boldsymbol{s})$ ... state distribution of old policy
- $Q^{\pi_{\mathrm{old}}}(\boldsymbol{s},\boldsymbol{a})$ .... Q-Function of old policy

Assumption:

- Policy does not change a lot
- I.e., we can neglect change in state distribution and Q-function

Baselines…

We can again use a baseline

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

- Baseline is now **state dependent** and **time dependent**

Good Baselines:

- Value function: $b_t(\boldsymbol{s}) = V_t^{\pi_{\mathrm{old}}}(\boldsymbol{s})$
- There is also a minimal variance baseline

Outline

Taxonomy of Policy Search Algorithms

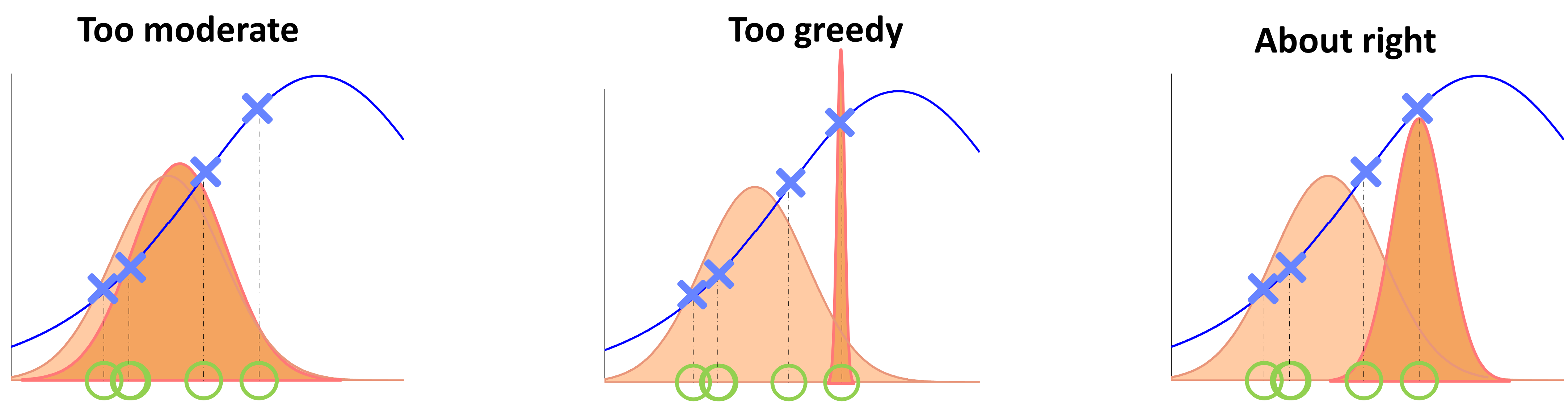
Model-Free Policy Search Methods

- Policy Gradients
- Natural Gradients
- Information Geometric Updates
- Success Matching

Metric in standard gradients

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

How can we choose the step size α ?



Aggressiveness of the policy update:

- Exploration-Exploitation tradeoff
- Robustness: Stay close to validity region of your data
- immediate vs. long-term performance

Metric in policy gradients

Define a bound/trust region to specify aggressiveness:

$$M(\pi, \pi_{\text{old}}) \leq \epsilon$$

- ϵ defines the distance in the metric space

Which metric *M* can we use?

- E.g, euclidian distance

Trajectory-based

$$L_2(\pi_{k+1}, \pi_k) = ||\boldsymbol{\omega}_{k+1} - \boldsymbol{\omega}_k||$$

Action-based

$$L_2(\pi_{k+1}, \pi_k) = ||\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k||$$

- Resulting step-size:

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

- **However:** Euclidean distance does not capture the change in the distribution!

Information-geometric constraints

Better Metric from information geometry: 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-geometric „distance“ measure between distributions
- „Most natural similarity measure for probability distributions“

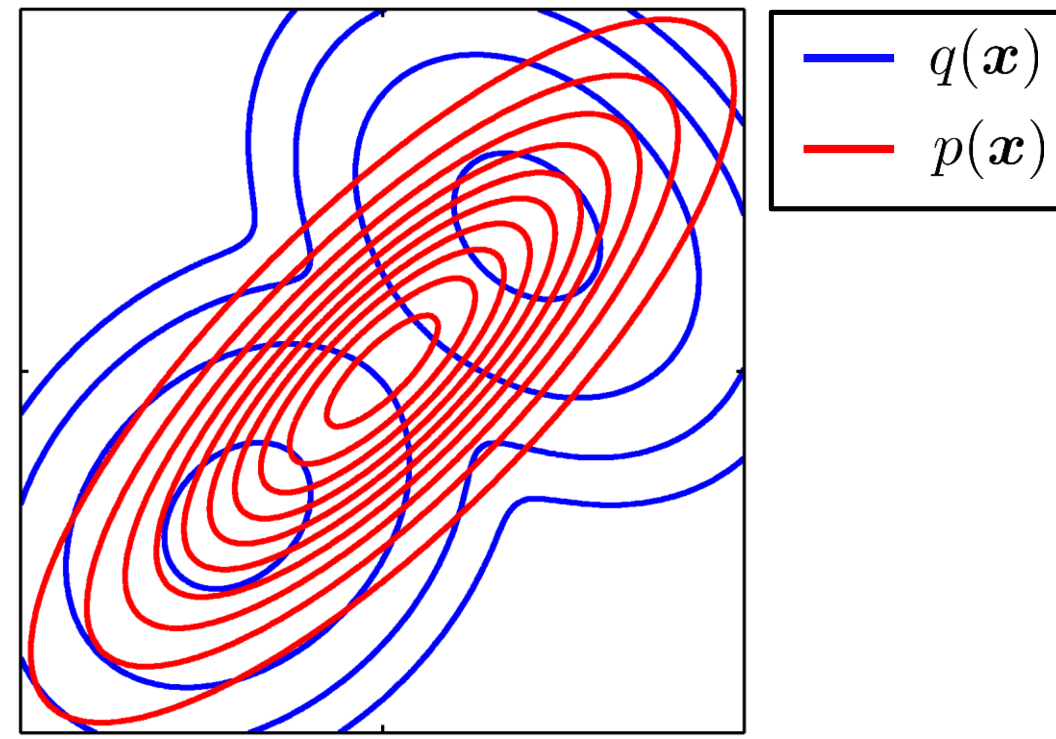
Properties:

- 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

Moment projection: $\mathrm{argmin}_p \mathrm{KL}(q||p)$

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



KL-Bound: $\mathrm{KL}(\pi_{\mathrm{old}}||\pi) \leq \epsilon$

- Limits the difference in the moments of both policies

Bishop, 2006

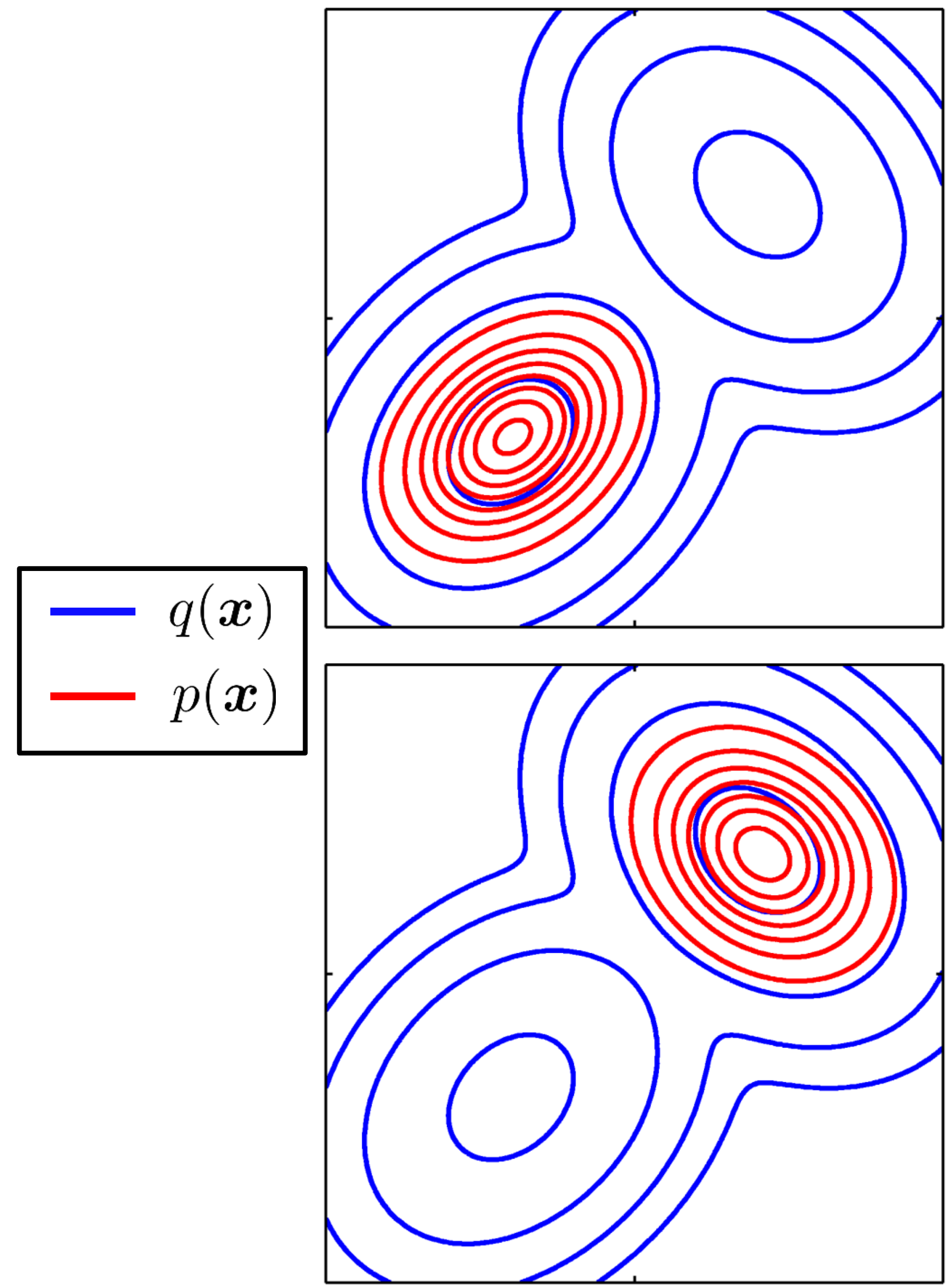
Kullback-Leibler Divergence

Information projection: $\mathrm{argmin}_p \mathrm{KL}(p||q)$

- *p* is zero wherever *q* is zero (zero forcing)
- not unique for most distributions
- Contains the entropy of *p*

KL-Bound: $\mathrm{KL}(\pi_{\mathrm{old}}||\pi) \leq \epsilon$

- Limits the information gain of the policy update



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)**

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

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

$$G(\boldsymbol{\theta}) = \mathbb{E}_\pi \left[\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{x}; \boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{x}; \boldsymbol{\theta})^T \right]$$

➡ Captures information how the **parameters influence the distribution**

Natural Gradients

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

- Linearized objective: Find direction $\Delta\omega$ maximally correlated with gradient
- Quadratized KL constraint

$$\nabla_{\boldsymbol{\theta}}^{\mathrm{NG}} J = \arg\max_{\Delta\boldsymbol{\theta}} \Delta\boldsymbol{\theta}^T \nabla_{\boldsymbol{\theta}} J$$

$$\text{s.t.} \quad \Delta\boldsymbol{\theta}^T \boldsymbol{G}(\boldsymbol{\theta}) \Delta\boldsymbol{\theta} \leq \epsilon$$

Note: The 2nd order Taylor approximation is symetric:

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

- For approximate information-geometric trust regions, it does not matter which KL we take

Natural Gradients

The solution to this optimization problem is given as:

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

- Inverse of the FIM: every parameter has the same influence!
- Invariant to linear transformations of the parameter space!
- We can optimize for η in closed form (Lagrangian multiplier)
- Can be directly applied to the trajectory-based policy gradient:
  - Natural Evolutionary Strategy (NES) [Wiestra, Sun, Peters & Schmidhuber 2008]

Natural Policy Gradients

Action-based policy gradient:

- We need to compute Fisher information matrix over trajectories

$$\boldsymbol{G}(\boldsymbol{\theta}) = \mathbb{E}_{p(\boldsymbol{\tau};\boldsymbol{\theta})}\left[\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau};\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau};\boldsymbol{\theta})^T\right]$$

  - Trajectory distribution not known, hard to compute

- It can be shown that we can compute the **all action matrix instead** [Peters & Schaal, 2003]

$$\boldsymbol{F}(\boldsymbol{\theta}) = \sum_{t=1}^{T} \mathbb{E}_{p^{\pi}(\boldsymbol{s})\pi(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{\theta})}\left[\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{\theta})^T\right] = \boldsymbol{G}(\boldsymbol{\theta})$$

  - Easier to compute

Result: Action-based natural gradient

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

Computing the FIM

Two ways to compute the FIM

- Closed form solution
- Compatible function approximation

Closed form FIM computation

Closed-form solution:

$$\boldsymbol{F}(\boldsymbol{\theta}) \approx \sum_{t=1}^{T} 1/N \sum_{i} \underbrace{\mathbb{E}_{\pi(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{\theta})}\left[\nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} \log \pi(\boldsymbol{a}|\boldsymbol{s};\boldsymbol{\theta})^T \middle| \boldsymbol{s} = \boldsymbol{s}^{[i]}\right]}_{\boldsymbol{F}(\boldsymbol{\theta},\boldsymbol{s}^{[i]})}$$

- Average the state FIM $\boldsymbol{F}(\boldsymbol{\theta}, \boldsymbol{s})$ over the state samples
- For most policies, the inner term can be computed in closed form
- **E.g.: Gaussian distributions**

Algorithms:

- **Trajectory-based:** Natural Evolutionary Strategy (NES) [Wiestra, Schaul, Peters & Schmidhuber, 2008]
- **Action-based:** Trust Region Policy Optimization (TRPO) [Schulman et al, 2015]

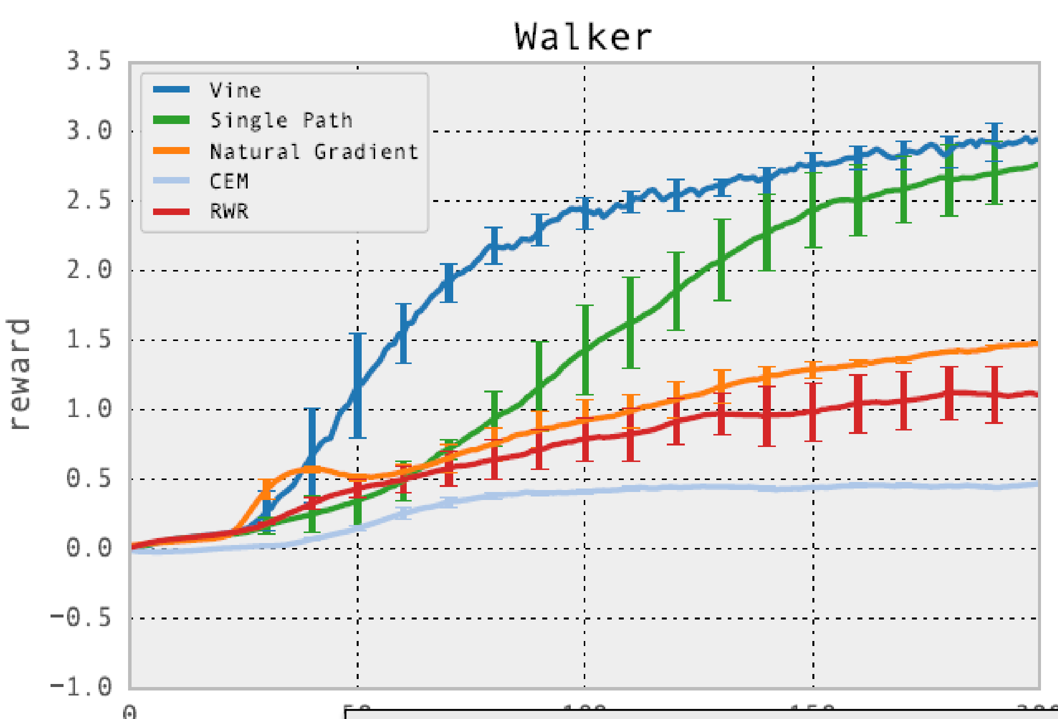
TRPO for Deep Reinforcement Learning

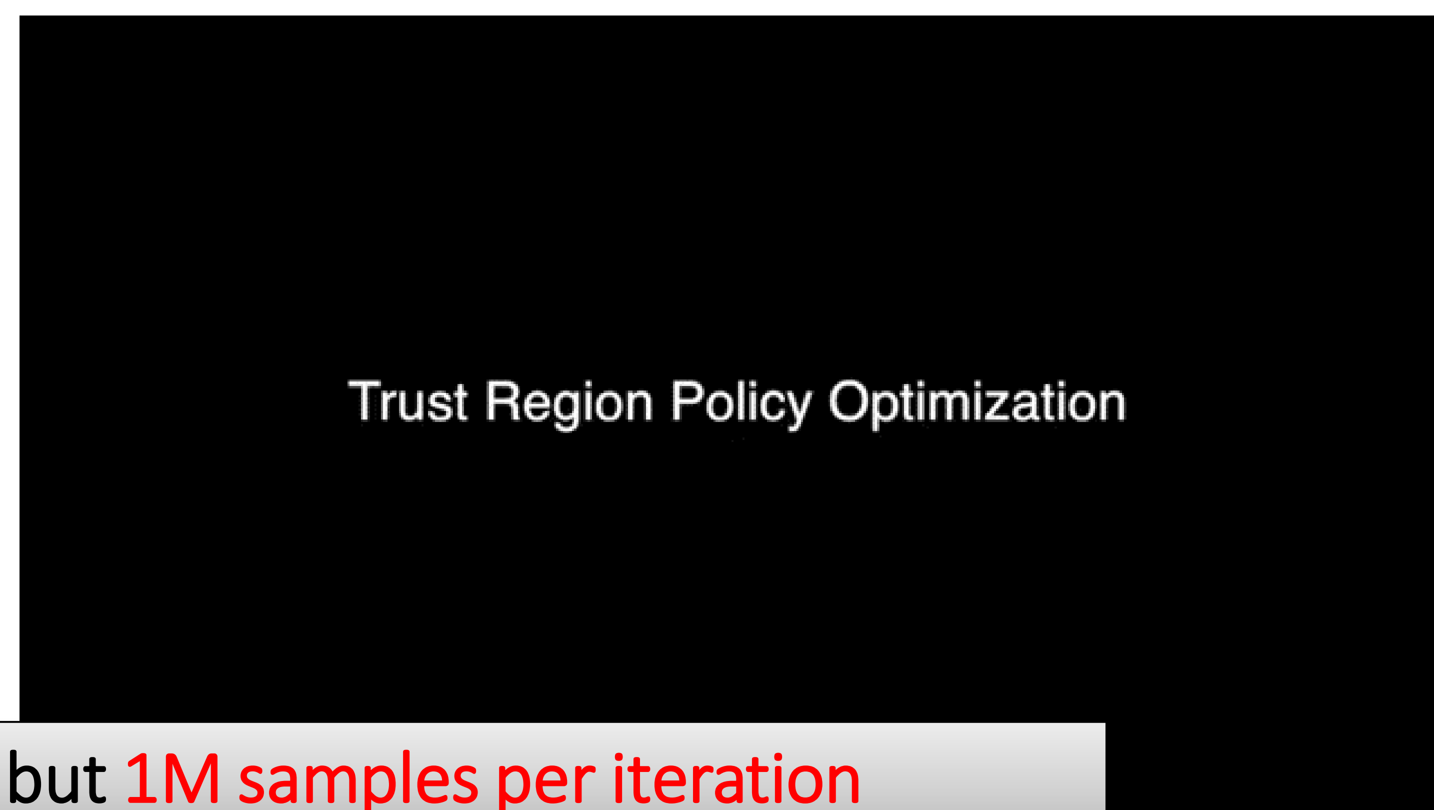
Trust Region Policy Optimization (TRPO):

- State of the art for optimizing deep neural networks
- **Problem:** FIM gets huge

Use conjugate gradient as approximation

- FIM never explicitely represented, only FIM times gradie
- No need to invert FIM
- Line search to find step-size on exact KL constraint





Works very well… but 1M samples per iteration

[Schulman, Levine, Moritz, Jordon & Abbeel, Trust Regoin Policy Optimization, ICML 2015]

What we have seen from the policy gradients

- Policy gradients dominated policy search for a long time and solidly working methods exist.
- They need a lot of samples
- Approximate information-geometric constraints can be easily implemented
- Learning the exploration rate / variance is still difficult

Outline

Taxonomy of Policy Search Algorithms

Model-Free Policy Search Methods

- Policy Gradients
- Natural Gradients
- Exact Information Geometric Updates
- Success Matching

Policy Search Methods for Multi-Agent Systems

Exact Information Geometric Constraints

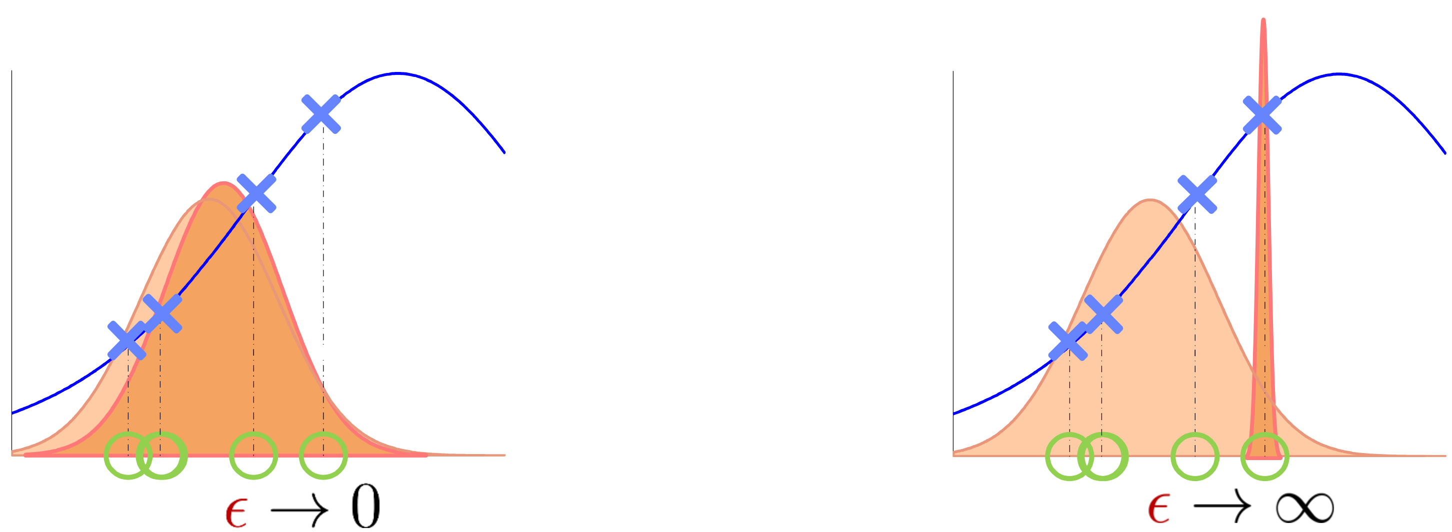
Exact information-theoretic policy update (trajectory-based):

1. Maximize return $$\arg\max_{\pi} \int \pi(\boldsymbol{\theta}) R(\boldsymbol{\theta}) d\boldsymbol{\theta}$$

2. Bound information gain [Peters et al, 2011] $$\text{s.t.} \quad \mathrm{KL}\big(\pi || \pi_{\text{old}}\big) \leq \epsilon$$

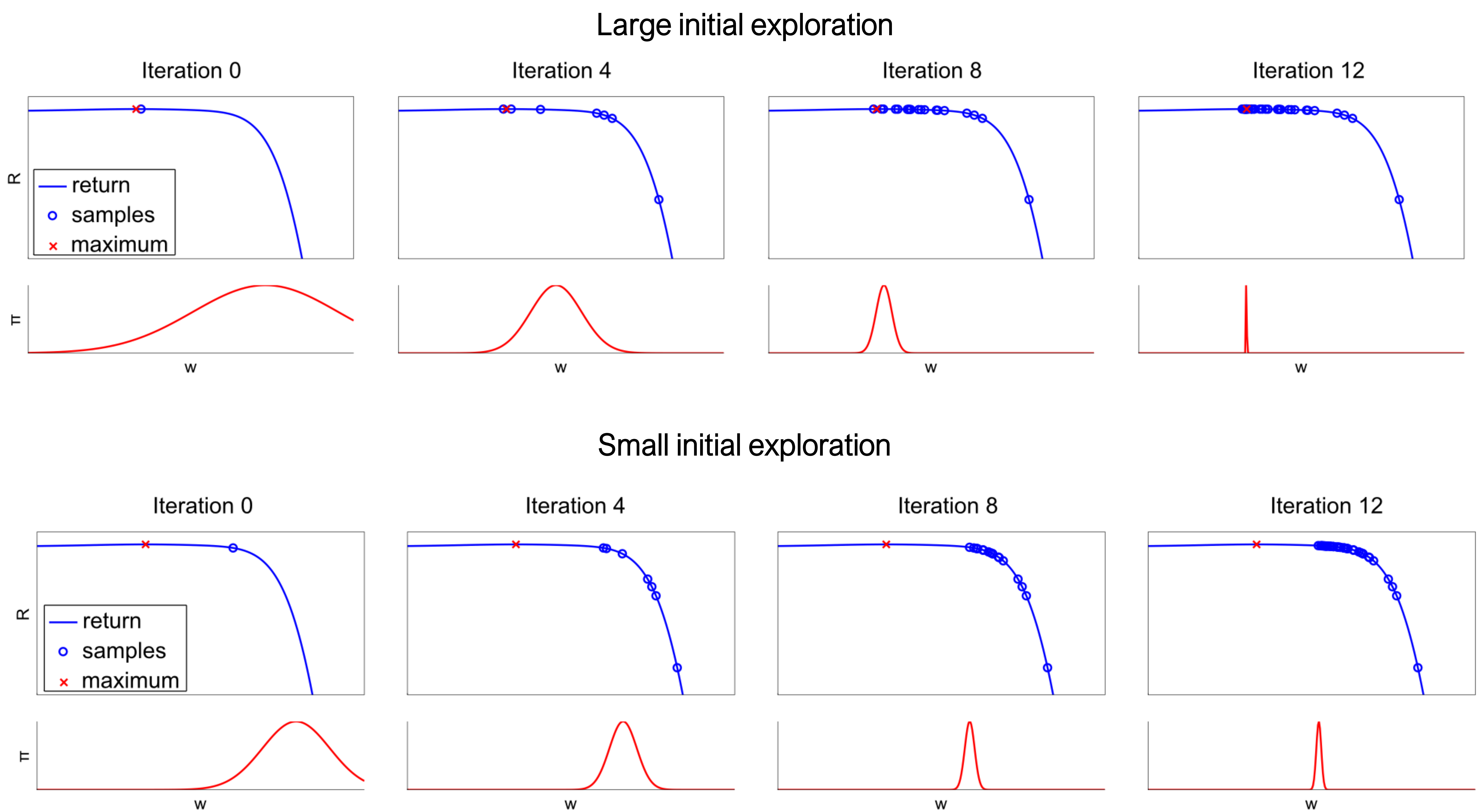
   Controls step-size for mean and covariance

Algorithm is called Relative Entropy Policy Search (REPS) [Peters et al., 2011]



J. Peters et al., *Relative Entropy Policy Search,* Association for the Advancement of Artificial Intelligence (AAAI), 2011

Illustration: Distribution Update



Information-Theoretic Policy Update

Information-theoretic policy update: incorporate information from new samples

1. Maximize return $$\arg\max_{\pi} \int \pi(\boldsymbol{\theta}) R(\boldsymbol{\theta}) d\boldsymbol{\theta}$$
2. Bound information gain [Peters 2011] $$\text{s.t.} \quad \mathrm{KL}\big(\pi || \pi_{\text{old}}\big) \leq \epsilon$$
3. Bound entropy loss [Abdolmaleki 2015] $$\underbrace{H(\pi_{\text{old}}) - H(\pi)}_{\text{loss in entropy}} \leq \gamma$$

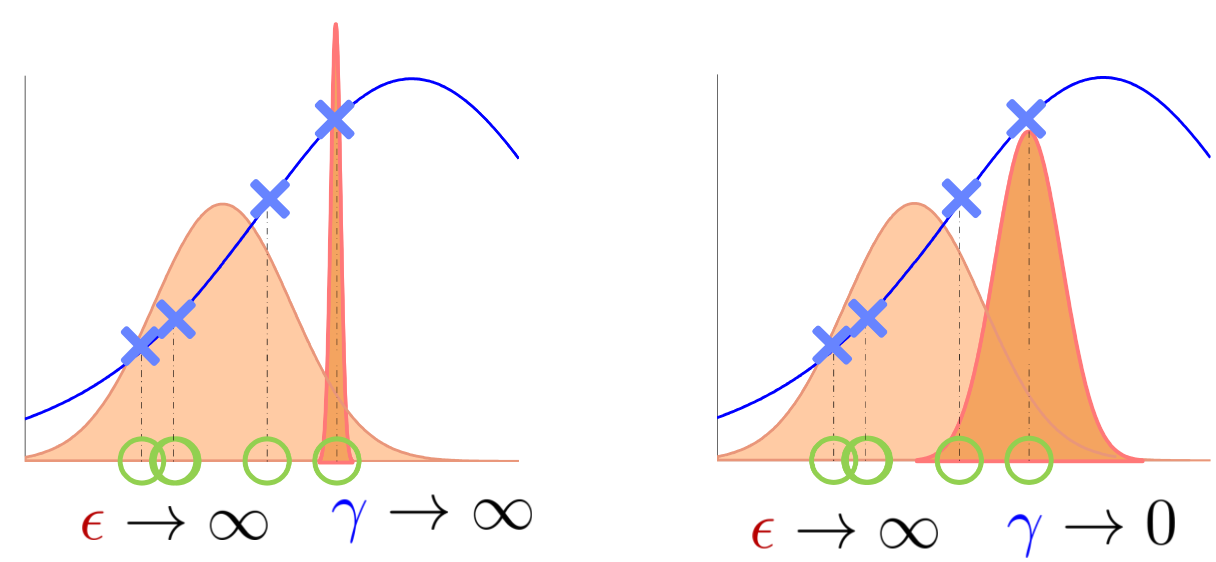
Reduces variance too quickly

Exploration Parameters

Entropy:

$$\mathrm{H}(p) = -\int p(\boldsymbol{w}) \log p(\boldsymbol{w}) d\boldsymbol{w}$$

- Measure for uncertainty

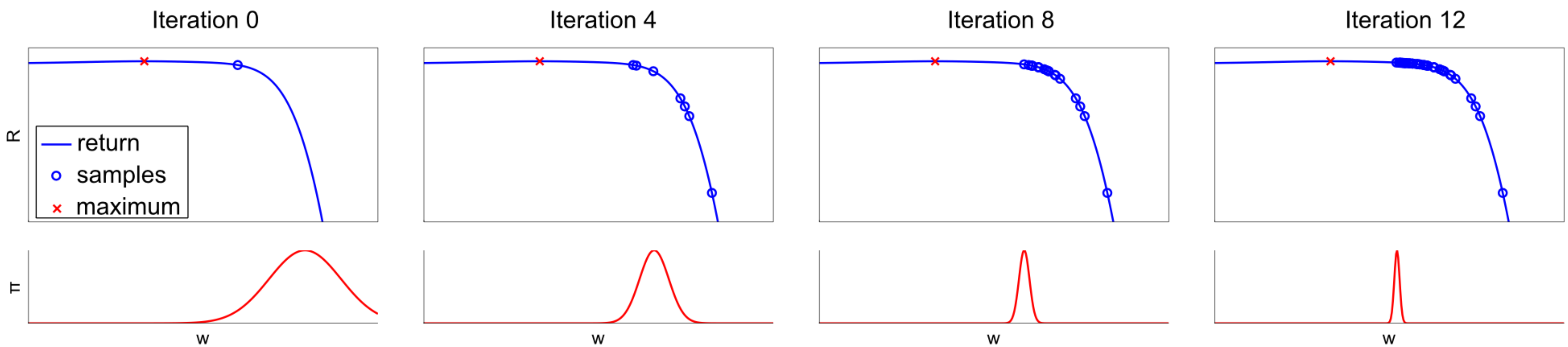


J. Peters et al., *Relative Entropy Policy Search,* Association for the Advancement of Artificial Intelligence (AAAI), 2011

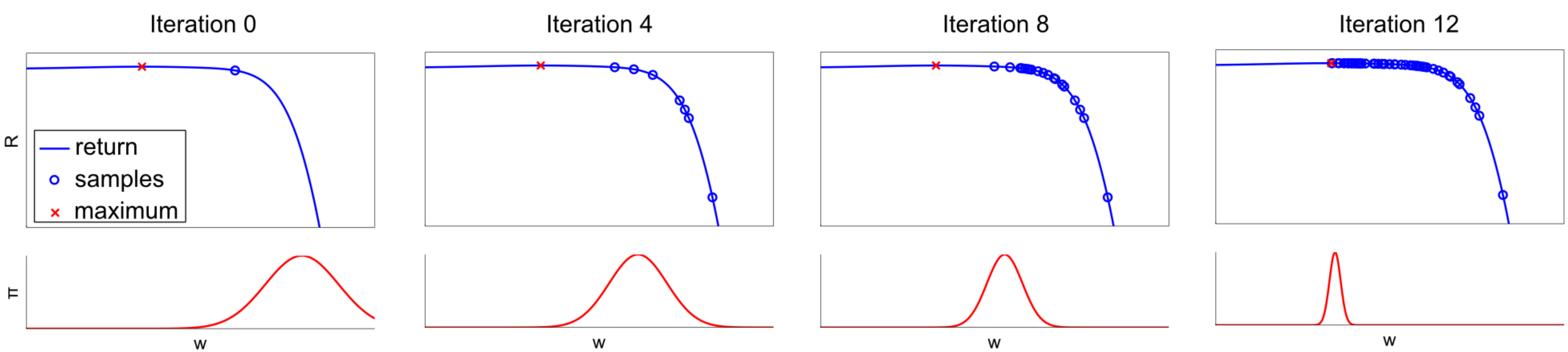
A. Abdolmaleki, …, **G. Neumann**, *Model-Based Relative Entropy Stochastic Search,* NIPS 2015

Illustration: Distribution Update

No entropy loss bound



With bounded entropy loss



Solution for Search Distribution

Solution for unconstrained distribution: $\pi(\boldsymbol{w}) \propto \pi_{\text{old}}(\boldsymbol{w})^{\frac{\eta}{\eta+\omega}} \exp\left(\frac{R(\boldsymbol{w})}{\eta+\omega}\right)$

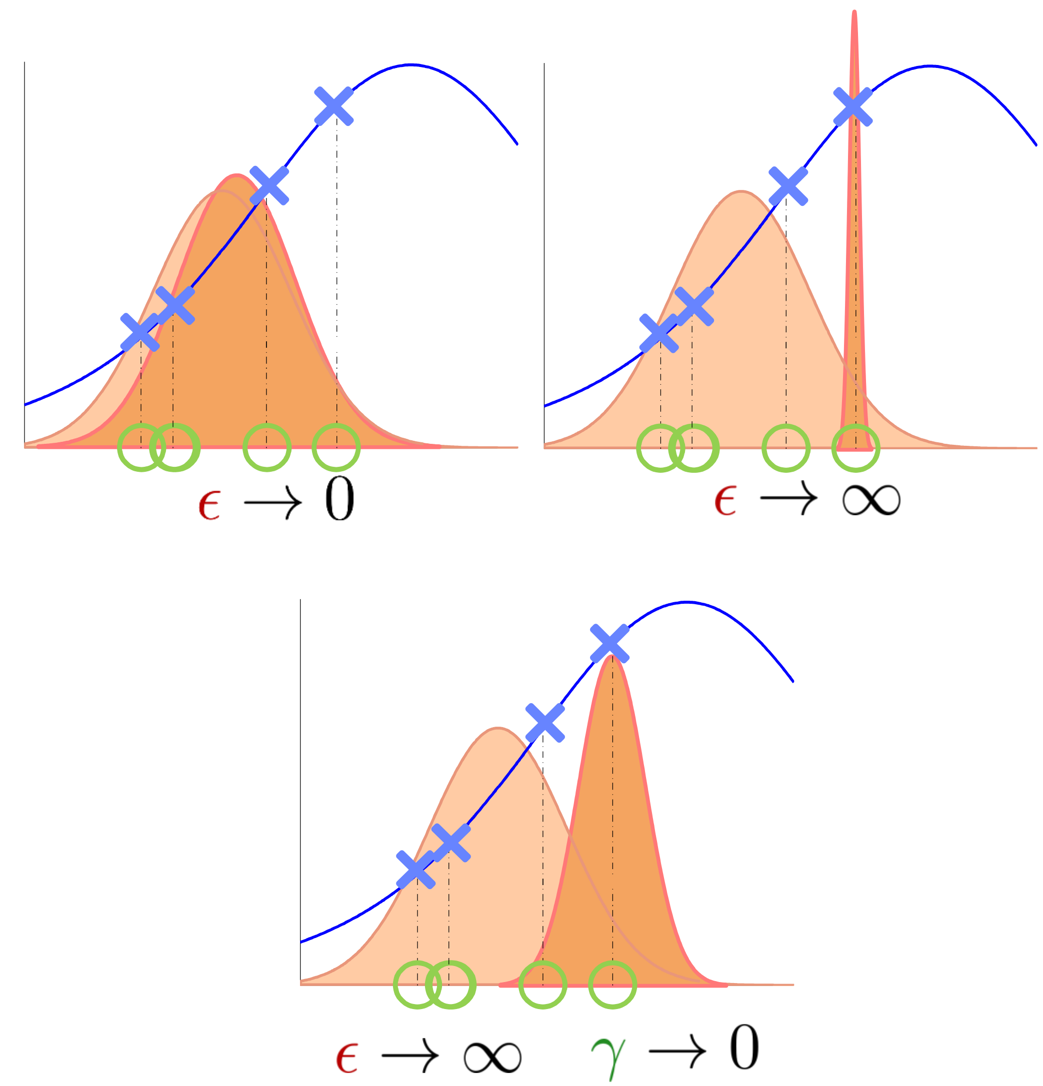
- η ... Lagrangian multiplier for: $\mathrm{KL}(\pi || \pi_{\text{old}}) \leq \epsilon$

  $\epsilon \rightarrow 0 \Rightarrow \eta \rightarrow \infty \Rightarrow \pi \rightarrow \pi_{\text{old}}$

  $\epsilon \rightarrow \infty \Rightarrow \eta \rightarrow 0 \Rightarrow \pi \rightarrow$ greedy

- ω ... Lagrangian multiplier for: $H(\pi_{\text{old}}) - H(\pi) \leq \gamma$

  $\gamma \rightarrow 0 \Rightarrow \omega \gg 0 \Rightarrow \pi \rightarrow$ more uniform



Gaussianity needs to be „enforced“ !

- Fit new policy on samples (REPS, [Daniel2012, Kupcsik2014, Neumann2014])
- Fit return function on samples (MORE, [Abdolmaleki2015])

A. Abdolmaleki, ..., J. Peters, G. Neumann, *Model-Based Relative Entropy Stochastic Search,* NIPS 2015

Fit Return Function

Use compatible function approximation:

- Gaussian distribution: $\mathcal{N}[\boldsymbol{\theta}|\boldsymbol{m}, \boldsymbol{\Lambda}] \propto \exp\left(\underbrace{-\frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\Lambda}\boldsymbol{\theta}}_{\text{quadratic}} + \underbrace{\boldsymbol{\theta}^T\boldsymbol{m}}_{\text{linear}} + \underbrace{\text{const}}_{\text{const}}\right)$
- Gaussian in cannonical form (log linear)
- Precision $\boldsymbol{\Lambda}$ and linear part $\boldsymbol{m}$
- **Compatible basis:**

$$\nabla_{\boldsymbol{\Lambda}} \log \pi(\boldsymbol{\theta};\boldsymbol{\omega}) = \boldsymbol{\theta}\boldsymbol{\theta}^T, \quad \nabla_{\boldsymbol{m}} \log \pi(\boldsymbol{\theta};\boldsymbol{\omega}) = \boldsymbol{\theta}$$

Match functional form:

$$\tilde{R}(\boldsymbol{\theta}) = \overbrace{\boldsymbol{\theta}^T\boldsymbol{A}\boldsymbol{\theta}} + \overbrace{\boldsymbol{a}^T\boldsymbol{\theta}} + \overbrace{a_0} \approx R(\boldsymbol{\theta})$$

- Quadratic in $\boldsymbol{\theta}$, but linear in parameters: $\boldsymbol{w} = \{\boldsymbol{A}, \boldsymbol{a}, a_0\}$
- $\boldsymbol{w}$ obtained by **linear regression** on current set of samples

A. Abdolmaleki, …, **G. Neumann**, *Model-Based Relative Entropy Stochastic Search,* NIPS 2015

Fit Return Function

Model-Based Relative Entropy Stochastic Search (MORE) : [Abdolmaleki 2015]

1. **Evaluation:** Fit local surrogate $\tilde{R}(\boldsymbol{\theta}) \approx \boldsymbol{\theta}^T \boldsymbol{A}\boldsymbol{\theta} + \boldsymbol{a}^T\boldsymbol{\theta} + a_0$

2. **Update:** $\pi(\boldsymbol{\theta}) \propto \underbrace{\pi_{\text{old}}(\boldsymbol{\theta})^{\frac{\eta}{\eta+\omega}}}_{\text{prior}} \underbrace{\exp\left(\frac{\tilde{R}(\boldsymbol{\theta})}{\eta+\omega}\right)}_{\text{likelihood}} \Rightarrow \pi(\boldsymbol{\theta}) = \underbrace{\mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^*)}_{\text{posterior}}$

Linear Term: $\boldsymbol{m}^* = \eta \boldsymbol{m}_{\text{old}} + \boldsymbol{a}$

Precision: $\boldsymbol{\Lambda}^* = \dfrac{\eta \boldsymbol{\Lambda}_{\text{old}} - 2\boldsymbol{A}}{\eta + \omega}$

} Obtain mean and covariance

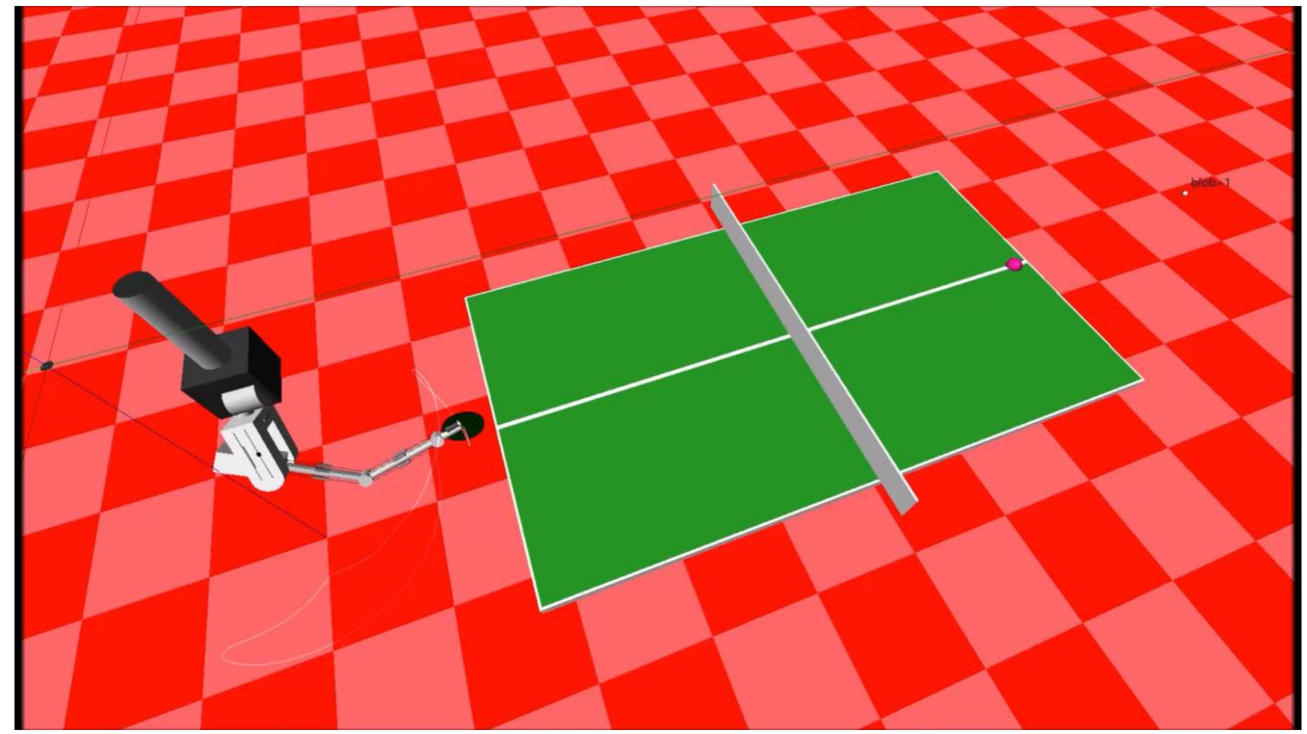
⇨ Interpolates in the natural parameter space (log linear parameters)

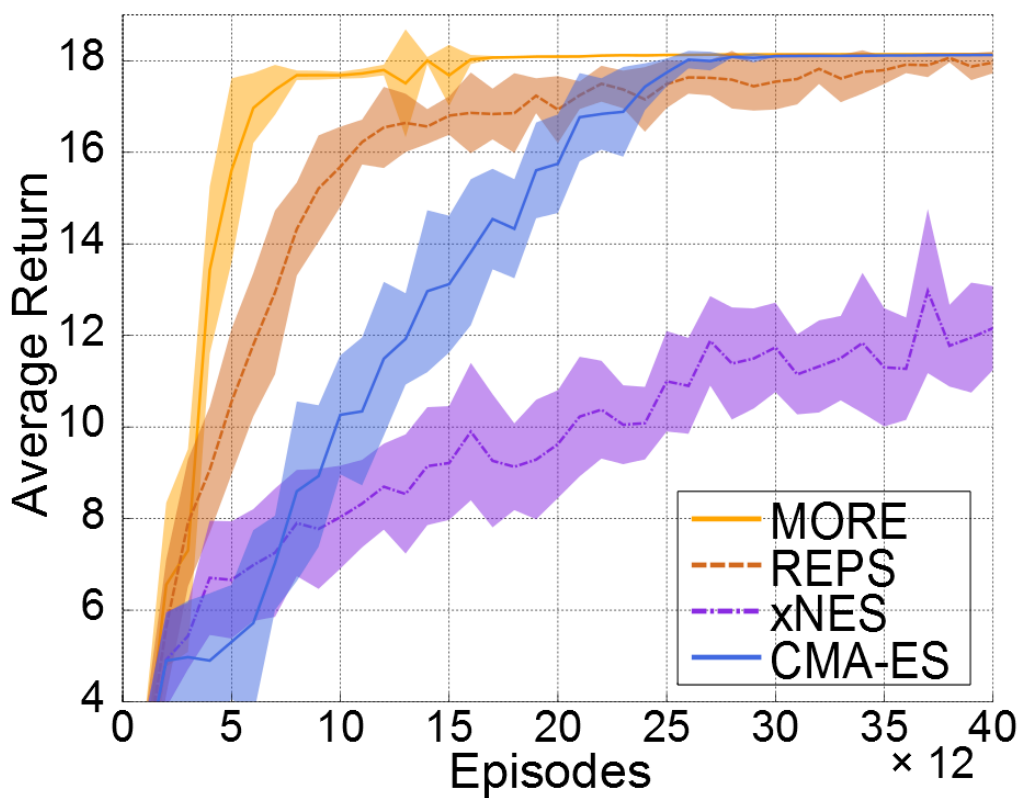
A. Abdolmaleki, …, **G. Neumann**, *Model-Based Relative Entropy Stochastic Search,* NIPS 2015

Skill Improvement: Table Tennis

Setup:

- Single ball configuration
- 17 movement primitive parameters (DMPs)

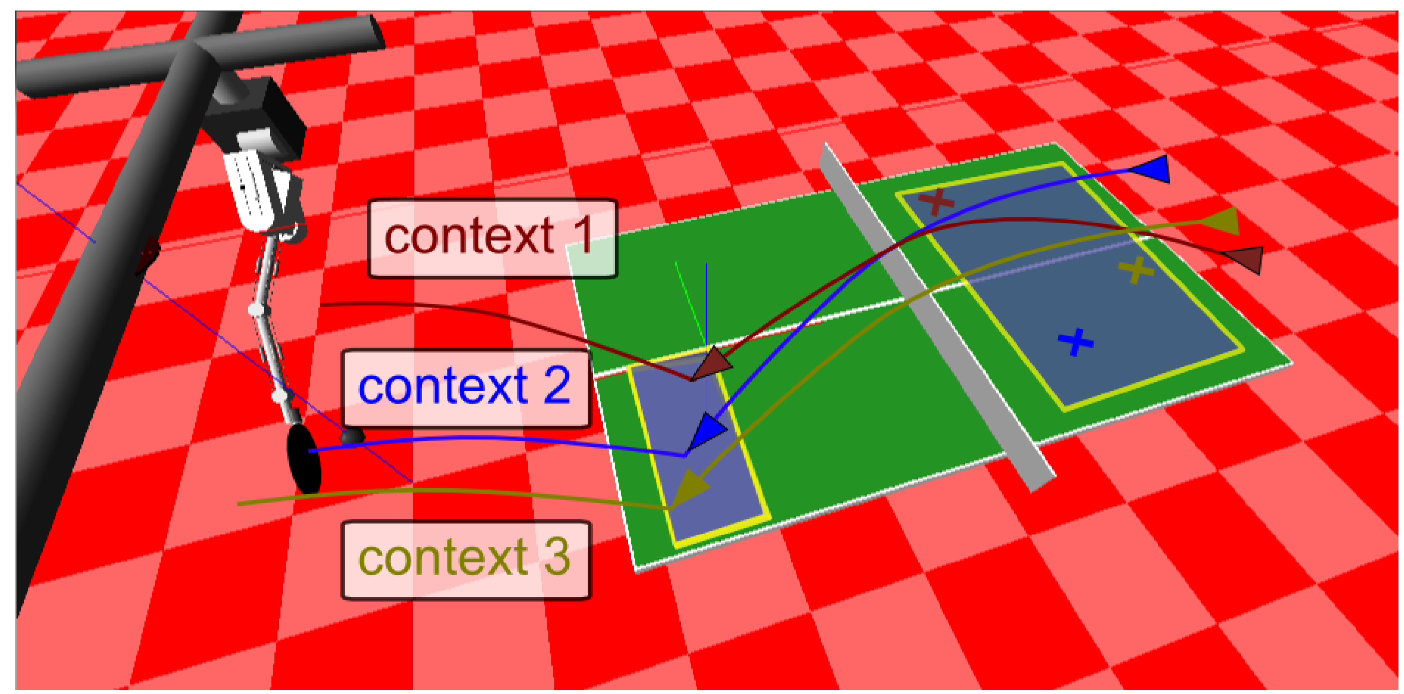




Adaptation of Skills

Goal: Adapt parameters $\boldsymbol{\theta}$ to different situations

- Different ball trajectories
- Different target locations



Introduce context vector $\boldsymbol{c}$

- Continuous valued vector
- Characterizes environment and objectives of agent
- Individual context per task execution

$$\boldsymbol{c} \sim p(\boldsymbol{c})$$

Use **contextual search distribution:**

$$\pi(\boldsymbol{\theta}|\boldsymbol{c}) = \mathcal{N}\left(\boldsymbol{\theta}|\boldsymbol{M}\boldsymbol{\phi}(\boldsymbol{c}), \boldsymbol{\Sigma}\right)$$

Abdolmaleki, …, **Neumann**, *Model-Based Relative Entropy Stochastic Search,* NIPS 2015
Kupcsik, …, **Neumann**, *Model-based Contextual Policy Search for Data-Efficient Generalization of Robot Skills, Artificial Intelligence, 2015*
Kupcsik, …, **Neumann**, *Data-Efficient Generalization of Robot Skills with Contextual Policy Search,* AAAI *2013*

Adaptation of Skills

Contextual distribution update:

1. Maximize **expected** return

$$\arg\max_{\pi} \mathbb{E}_{p(\boldsymbol{c})}\left[\int \pi(\boldsymbol{\theta}|\boldsymbol{c})R(\boldsymbol{c},\boldsymbol{\theta})d\boldsymbol{\theta}\right]$$

2. Bound **expected** information loss

$$\text{s.t.:} \quad \mathbb{E}_{p(\boldsymbol{c})}\left[\mathrm{KL}\big(\pi(\cdot|\boldsymbol{c})||\pi_{\text{old}}(\cdot|\boldsymbol{c})\big)\right] \le \epsilon$$

3. Bound entropy loss

$$\underbrace{H(\pi_{\text{old}}) - H(\pi)}_{\text{loss in entropy}} \le \gamma$$

Contextual MORE: [Tangaratt 2017]

1. **Evaluation:** Fit local surrogate $\tilde{R}(\boldsymbol{c},\boldsymbol{\theta}) \approx \boldsymbol{\theta}^T\boldsymbol{A}\boldsymbol{\theta} + \boldsymbol{\theta}^T\boldsymbol{B}\boldsymbol{\phi}(\boldsymbol{c}) + \boldsymbol{a}^T\boldsymbol{\theta} + a_0$

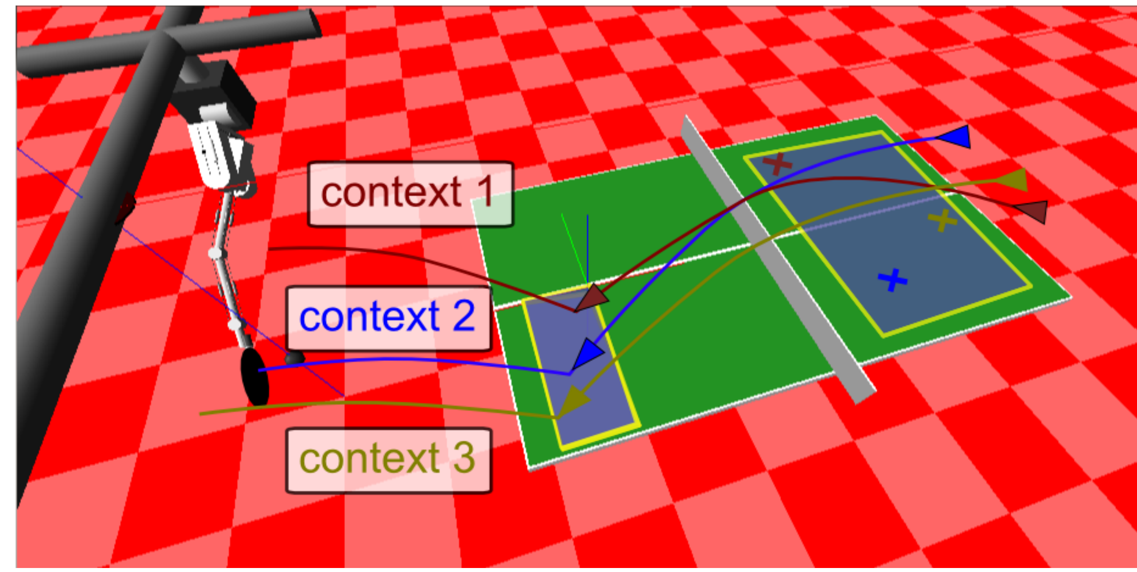
2. **Update:** $\pi(\boldsymbol{\theta}|\boldsymbol{c}) \propto \underbrace{\pi_{\text{old}}(\boldsymbol{\theta}|\boldsymbol{c})^{\frac{\eta}{\eta+\omega}}}_{\text{prior}} \underbrace{\exp\left(\frac{\tilde{R}(\boldsymbol{c},\boldsymbol{\theta})}{\eta+\omega}\right)}_{\text{likelihood}} \Rightarrow \pi(\boldsymbol{\theta}|\boldsymbol{c}) = \underbrace{\mathcal{N}\left(\boldsymbol{\theta}|\boldsymbol{M}^*\boldsymbol{\phi}(\boldsymbol{c}),\boldsymbol{\Sigma}^*\right)}_{\text{posterior}}$

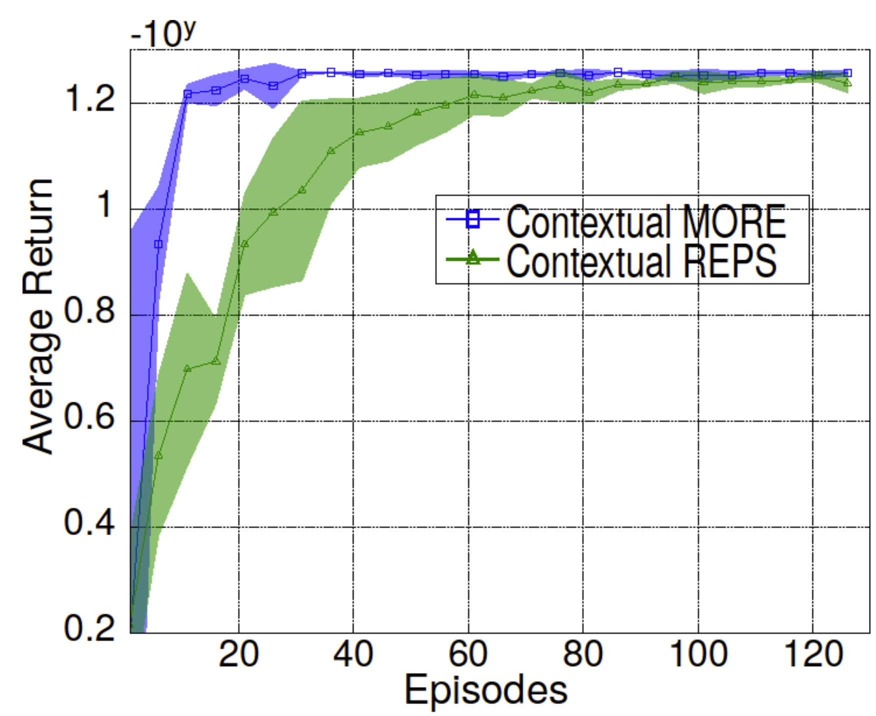
A. Abdolmaleki, …, **G. Neumann**, *Model-Based Relative Entropy Stochastic Search,* NIPS 2015

Adaptation of Skills: Table Tennis

Contextual Policy Search:

- Context: Initial ball velocity (in 3 dimensions)
- Successfully return 100% of the balls



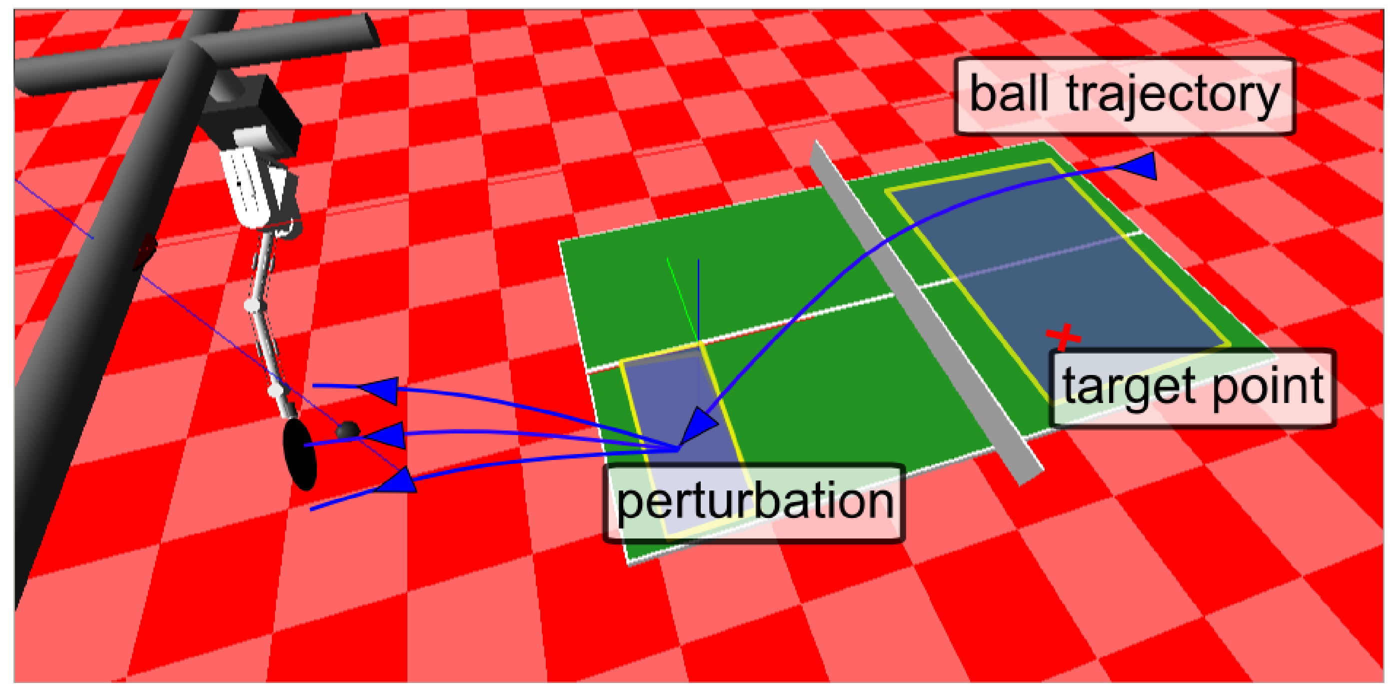


Action-based KL-constraints: Reactive Skills

Goal: React to unforeseen events

- Adaptation during execution of the movement
- Add perceptual variables to state representation
- E.g.: ball position + velocity

Example: Perturbation at impact (spin)



Use action-based stochastic policy:

- Time dependent linear feedback controllers

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

Policy Evaluation

Compatible Value Function Approximation:

- **V-Function (baseline):**

  Quality of state *s* when following policy

$$V_t^{\pi}(\boldsymbol{s}) = \mathbb{E}_{\pi}\left[\sum_{h=t}^{T} r_h(\boldsymbol{s}_h, \boldsymbol{a}_h)\middle|\boldsymbol{s}_t = \boldsymbol{s}\right] \approx \boldsymbol{s}^T\boldsymbol{V}_t\boldsymbol{s} + \boldsymbol{s}^T\boldsymbol{v}_t + v_{0,t}$$

- **Q-Function (compatible approximation):**

  Quality of state *s* when taking action *a* and following policy afterwards

$$Q_t^{\pi}(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_{\pi}\left[\sum_{h=t}^{T} r_h(\boldsymbol{s}_h, \boldsymbol{u}_h)\middle|\boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{a}_t = \boldsymbol{a}\right] \approx \boldsymbol{a}^T\boldsymbol{Q}_t\boldsymbol{a} + \boldsymbol{a}^T\boldsymbol{B}_t\boldsymbol{s} + \boldsymbol{a}^T\boldsymbol{q}_t + q_{0,t} + f_t(\boldsymbol{s})$$

  - Quadratic in actions, linear in state

- Baseline and Q-function are time dependent
- Estimated by LSTD

Policy Improvement

Policy Improvement per Time-Step:

1. Maximize Q-Function
$$\arg\max_{\pi_t} \mathbb{E}_{p_t(\boldsymbol{s})}\left[\int \pi_t(\boldsymbol{a}|\boldsymbol{s}) Q_t^{\pi_{\text{old}}}(\boldsymbol{s},\boldsymbol{a}) d\boldsymbol{a}\right]$$
2. Bound expected information loss
$$\text{s.t.} \quad \mathbb{E}_{p_t(\boldsymbol{s})}\left[\text{KL}\big(\pi_t(\cdot|\boldsymbol{s})||\pi_{t,\text{old}}(\cdot|\boldsymbol{s})\big)\right] \leq \epsilon$$
3. Bound entropy loss
$$H(\pi_{t,\text{old}}) - H(\pi_t) \leq \gamma$$

Model-free Trajectory Optimization (MOTO): [Akrour 2016]

1. **Evaluation:** Fit local Q-Function
$$\tilde{Q}^{\pi_{old,t}}(\boldsymbol{s},\boldsymbol{a}) \approx \boldsymbol{a}^T \boldsymbol{Q}_t \boldsymbol{a} + \boldsymbol{a}^T \boldsymbol{B}_t \boldsymbol{s} + \boldsymbol{a}^T \boldsymbol{q}_t + q_{0,t} + f_t(\boldsymbol{s})$$
2. **Update:**
$$\pi_t(\boldsymbol{a}|\boldsymbol{s}) \propto \pi_{\text{old},t}(\boldsymbol{a}|\boldsymbol{s})^{\frac{\eta}{\eta+\omega}} \exp\left(\frac{\tilde{Q}_t^{\pi_{\text{old}}}(\boldsymbol{s},\boldsymbol{a})}{\eta+\omega}\right)$$

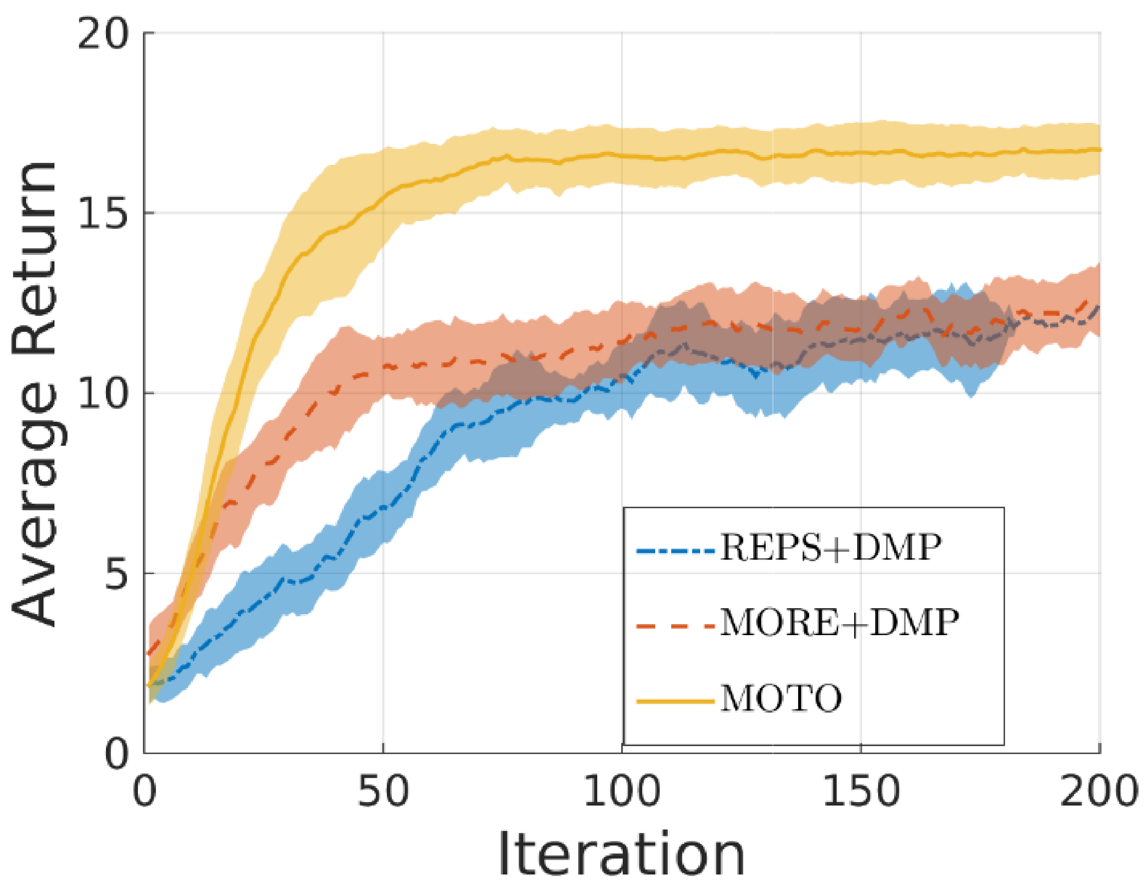
$$\Rightarrow \quad \pi_t(\boldsymbol{a}|\boldsymbol{s}) = \mathcal{N}\big(\boldsymbol{a}|\boldsymbol{K}_t^* \boldsymbol{s} + \boldsymbol{k}_t^*, \boldsymbol{\Sigma}_t^*\big)$$

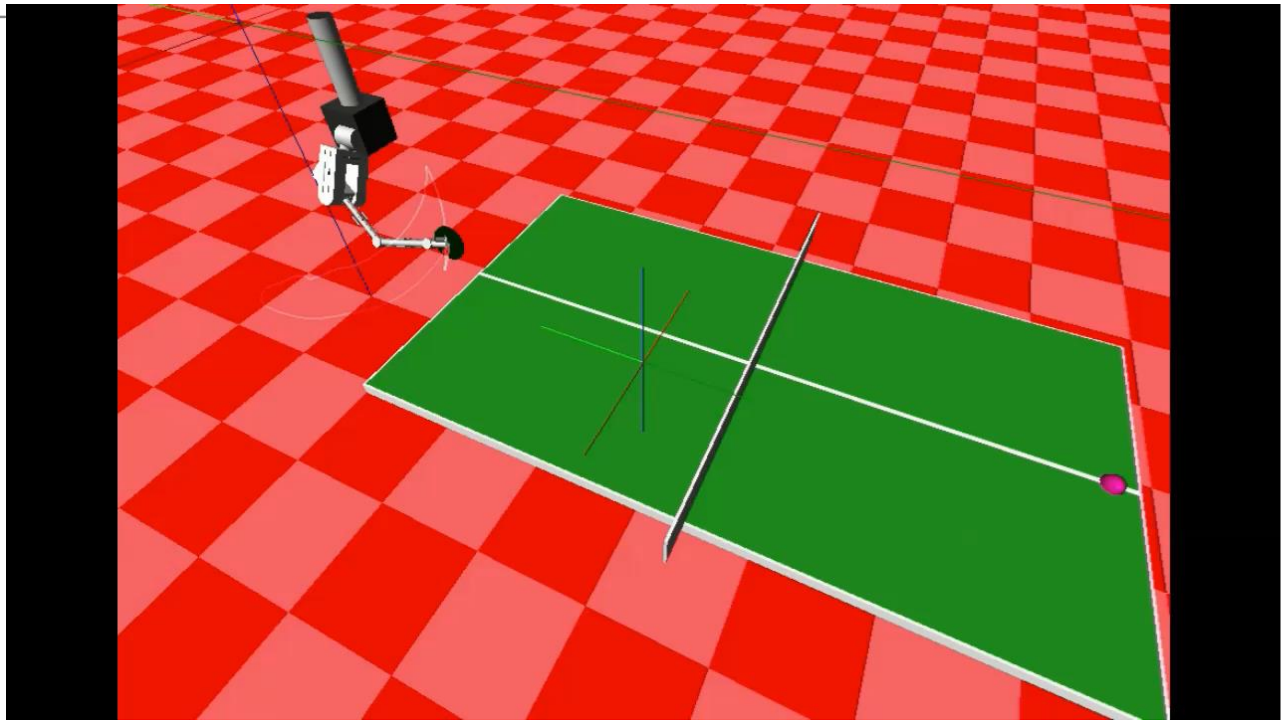
R. Akrour, …, G. Neumann, *Model-Free Trajectory Optimization for Reinforcement Learning of Motor Skills,* ICML 2016

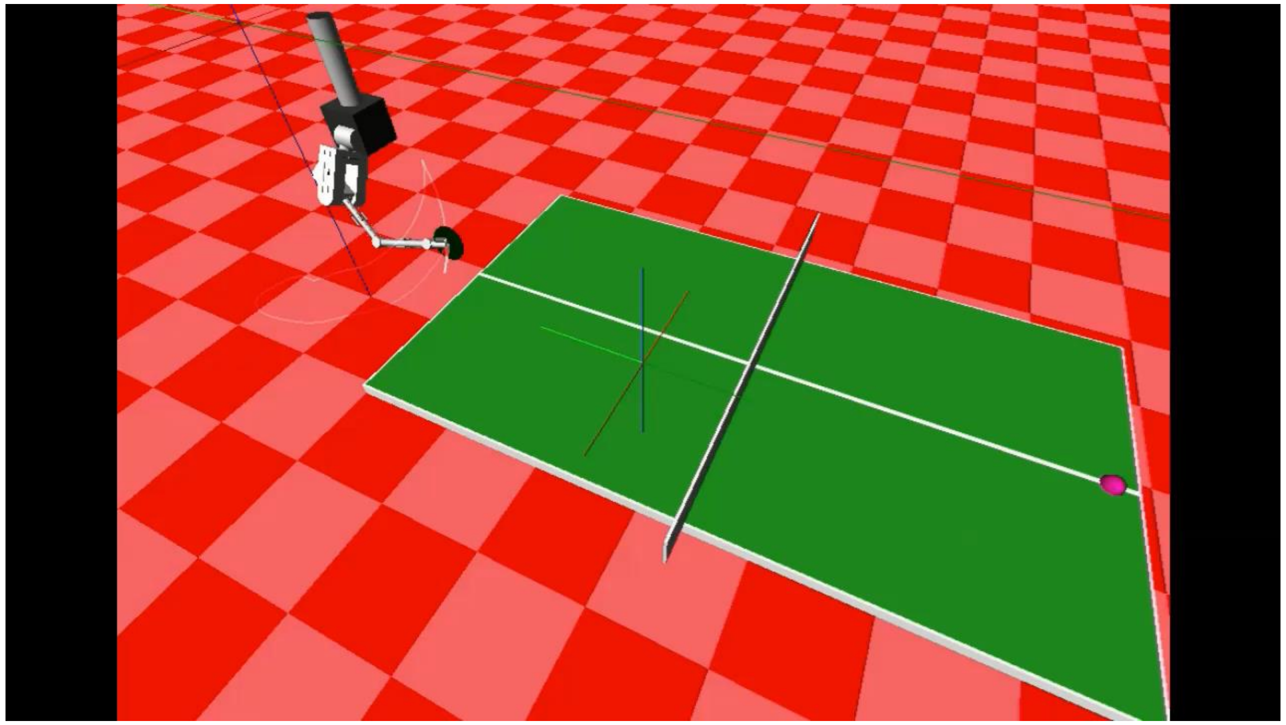
Reactive Skills: Table Tennis

Reactive Skills:

- Returns ball 100% of the times
- Not possible with desired trajectories







Wrap-up for exact information constraints

Exact information-geometric constraints:

- Efficient computation of the full-covariance matrix
- Can be used in trajectory-based and action-based formulation
- We can use entropy-loss regularization to prevent premature convergence

There is a tight connection between natural gradients and REPS

- If we use the natural parametrization (log-linear), REPS and natural gradients are equivalent
- I.e., only in this case the natural gradient solution is exact

Outline

Taxonomy of Policy Search Algorithms

Model-Free Policy Search Methods

- Policy Gradients
- Natural Gradients
- Exact Information Geometric Updates
- Success Matching

Policy Search Methods for Multi-Agent Systems

Success Matching Principle

Optimizing the average return is difficult:

- Non-linear, non-convex optimization problem
- Can we optimize a simpler, convex function instead?

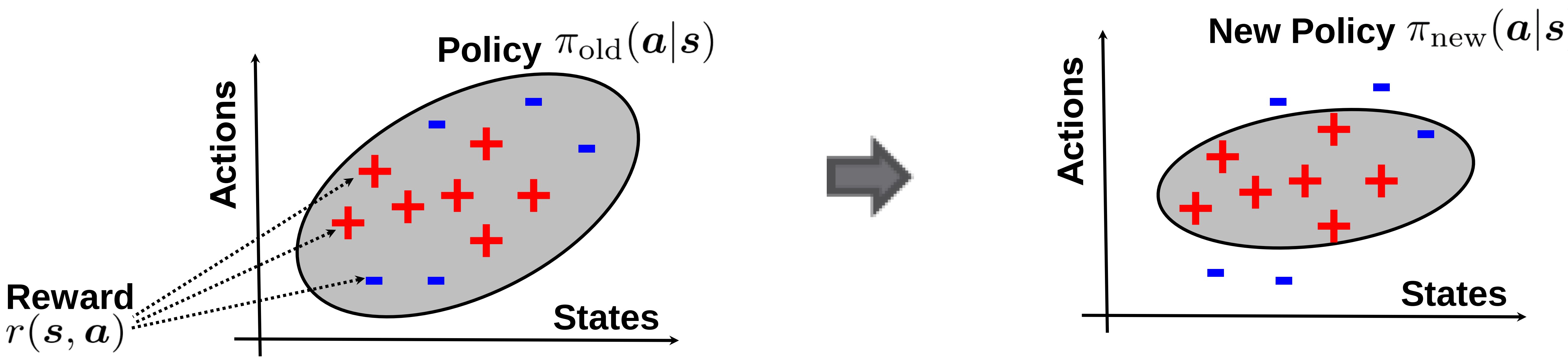
Success Matching Principle

> "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: reweighting by success probability $p(R=1|\boldsymbol{\tau})$

$$p^{\pi_{\text{new}}}(\boldsymbol{\tau}) \propto p(R|\boldsymbol{\tau})p^{\pi_{\text{old}}}(\boldsymbol{\tau})$$

- Binary reward event R = 1



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

Success Matching Principle

Success-Matching: policy reweighting by success probability $p(R=1|\boldsymbol{\tau})$

$$p^{\pi_{\text{new}}}(\boldsymbol{\tau}) \propto p(R|\boldsymbol{\tau})p^{\pi_{\text{old}}}(\boldsymbol{\tau})$$

Most common success distribution

- Exponential reweighting:

$$p(R=1|\boldsymbol{\tau}) \propto \exp(\eta R(\boldsymbol{\tau}))$$

Can be derived in many ways:

- Expectation maximization [Kober & Peters., 2008][Vlassis & Toussaint., 2009][Neumann, 2011]
- Optimal Control [Theodorou, Buchli & Schaal, 2010]
- Information Geometry [Peters et al, 2010, Daniel, Neumann & Peters, 2012]

Success Matching via Expectation Maximization

We want to maximize the average success probability

$$p(R;\boldsymbol{\theta}) = \int p(R|\boldsymbol{\tau})p(\boldsymbol{\tau};\boldsymbol{\theta})d\boldsymbol{\tau}$$

- This is a latent variable model.
- Trajectories that have high success are unknown

Success Matching via Expectation Maximization

Using the EM-decomposition [Bishop 2006], it is easy to show that

$$\log p(R; \boldsymbol{\theta}) = \mathcal{L}(q(\boldsymbol{\tau}), \boldsymbol{\theta}) + \mathrm{KL}\left(q(\boldsymbol{\tau}) || p(\boldsymbol{\tau} | R, \boldsymbol{\theta})\right)$$

- For any variational distribution $q(\boldsymbol{\tau})$

Lower Bound: $$\mathcal{L}(q(\boldsymbol{\tau}), \boldsymbol{\theta}) = \int q(\boldsymbol{\tau}) \log \frac{p(R|\boldsymbol{\tau}) p(\boldsymbol{\tau}; \boldsymbol{\theta})}{q(\boldsymbol{\tau})}$$

Posterior: $$p(\boldsymbol{\tau} | R, \boldsymbol{\theta}) = \frac{p(R|\boldsymbol{\tau}) p(\boldsymbol{\tau}; \boldsymbol{\theta})}{p(R; \boldsymbol{\theta})}$$

Success matching via Expectation Maximization

E-step: $\mathrm{argmin}_{q(\boldsymbol{\tau})} \mathrm{KL}\left(q(\boldsymbol{\tau}) || p(\boldsymbol{\tau}|R, \boldsymbol{\theta})\right)$

- Solution: $q(\boldsymbol{\tau}) = p(\boldsymbol{\tau}|R, \boldsymbol{\theta})$
- Lower Bound is tight after the E-step

$$\log p(R; \boldsymbol{\theta}) = \mathcal{L}(q(\boldsymbol{\tau}), \boldsymbol{\theta}) + \underbrace{\mathrm{KL}\left(q(\boldsymbol{\tau}) || p(\boldsymbol{\tau}|R, \boldsymbol{\theta})\right)}_{=0}$$

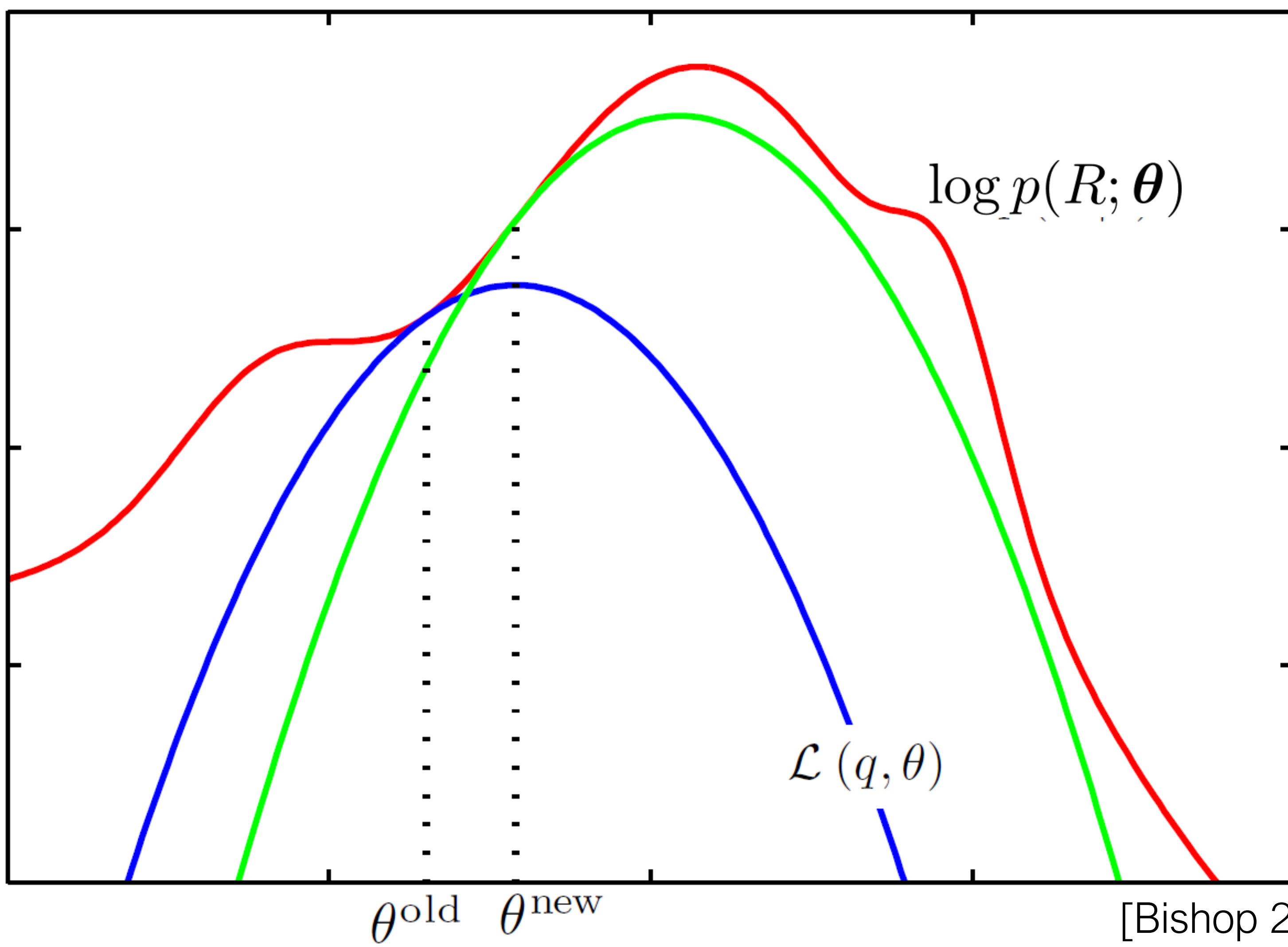
M-step:

$$\begin{aligned}
\boldsymbol{\theta}_{\mathrm{new}} &= \mathrm{argmax}_{\boldsymbol{\theta}}\, \mathcal{L}(q(\boldsymbol{\tau}), \boldsymbol{\theta}) = \mathrm{argmax}_{\boldsymbol{\theta}} \int q(\boldsymbol{\tau}) \log \frac{p(R|\boldsymbol{\tau}) p(\boldsymbol{\tau}; \boldsymbol{\theta})}{q(\boldsymbol{\tau})} d\boldsymbol{\tau} \\
&= \mathrm{argmax}_{\boldsymbol{\theta}} \int p(R|\boldsymbol{\tau}) p(\boldsymbol{\tau}; \boldsymbol{\theta}_{\mathrm{old}}) \log p(\boldsymbol{\tau}; \boldsymbol{\theta}) d\boldsymbol{\tau} \\
&\approx \mathrm{argmax}_{\boldsymbol{\theta}} \sum_{\boldsymbol{\tau}^{[i]} \sim p(\boldsymbol{\tau}; \boldsymbol{\theta}_{\mathrm{old}})} p(R|\boldsymbol{\tau}^{[i]}) \log p(\boldsymbol{\tau}^{[i]}; \boldsymbol{\theta})
\end{aligned}$$

- This is a weighted maximum log likelihood objective

Weighted ML objective

Lower bound is easier to optimize than the expected reward



[Bishop 2006]

- Closed form solution exist for many distributions

Weighted Maximum Likelihood Solutions…

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

Weighted mean:

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

Weighted covariance:

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

- with $w^{[i]} = p(R|\boldsymbol{\tau}^{[i]})$
- **But more general:** Also for mixture models, GPs and so on…
- Matches moments of $p(\boldsymbol{\theta}|R)$ and $\pi(\boldsymbol{\theta};\boldsymbol{w})$

Comparison to policy gradients

Weighted Maximum Likelihood Objective:

$$\boldsymbol{J}_{\mathrm{ML}}(\boldsymbol{\theta}) = \int p(\boldsymbol{\tau}|\boldsymbol{\theta}_{\mathrm{old}})p(R|\boldsymbol{\tau}) \log p(\boldsymbol{\tau};\boldsymbol{\theta})d\boldsymbol{\tau}$$

- Derivative (Weighted ML Solution):

$$\nabla_{\boldsymbol{\theta}}\boldsymbol{J}_{\mathrm{ML}} = \int p(\boldsymbol{\tau}|\boldsymbol{\theta}_{\mathrm{old}})p(R|\boldsymbol{\tau})\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau};\boldsymbol{\theta})d\boldsymbol{\tau}$$

$$\approx 1/N \sum_i \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau};\boldsymbol{\theta}^{[i]})p(R|\boldsymbol{\tau}^{[i]}) = 0$$

Average return objective:

$$\boldsymbol{J}(\boldsymbol{\theta}) = \int p(\boldsymbol{\tau}|\boldsymbol{\theta})R(\boldsymbol{\tau})d\boldsymbol{\tau}$$

- Derivative (Policy Gradient):

$$\nabla_{\boldsymbol{\theta}}\boldsymbol{J} = \int p(\boldsymbol{\tau}|\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau};\boldsymbol{\theta})R(\boldsymbol{\tau})d\boldsymbol{\tau}$$

$$\approx 1/N \sum_i \nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\tau};\boldsymbol{\theta}^{[i]})R(\boldsymbol{\tau}^{[i]})$$

Difference: reward transformation

Metric in Success Matching

Maximum Likelihood is inherently greedy

- How can we control the aggressiveness?
- What about overfitting?
  - In particular for the covariance matrix estimate

Limit change in moments:

$$\underbrace{\mathrm{argmax}_p \sum_i p(R|\boldsymbol{\tau}^{[i]}) \log p(\boldsymbol{\tau}^{[i]};\boldsymbol{\theta}) d\boldsymbol{\tau}}_{\text{weighted ML} = \text{Moment Matching}}, \quad \text{s.t.} \quad \underbrace{\mathrm{KL}\big(p_{\boldsymbol{\theta}_{\text{old}}}(\boldsymbol{\tau})||p_{\boldsymbol{\theta}}(\boldsymbol{\tau})\big) \le \epsilon}_{\text{Limit change in moments}}$$

- Reversed KL in comparison to REPS
- New distribution on the right
- Weighted maximum likelihood corresponds to **moment projection**

CMA-ES

The Covariance Matrix Adaptation - Evolutionary Strategy (CMA-ES) [Hansen 2003] is one of the most successful stochastic optimizers

- Developed from well established heuristics
- Theoretical background for most CMA-ES update rules is missing

Gaussian Search Distribution: $\pi(\boldsymbol{\theta};\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\theta};\boldsymbol{\mu},\sigma\boldsymbol{\Sigma})$

- Update rules for:
  - Mean $\boldsymbol{\mu}$
  - Covariance $\boldsymbol{\Sigma}$
  - Stepsize $\boldsymbol{\sigma}$

Inconsistent update rules that are not fully understood

Deriving and improving CMA-ES

CMA-ES can be **derived and improved using moment-KL bounds** [Abdolmaleki 2017]

- Algorithm called Trust Region CMA-ES

Trajectory/Parameter-based formulation:

$$\sum_i p(R|\boldsymbol{\theta}^{[i]}) \log \pi(\boldsymbol{\theta}^{[i]}; \boldsymbol{\omega}), \quad \text{s.t. } \mathrm{KL}\big(\pi_{\boldsymbol{\omega}_{\text{old}}}(\boldsymbol{\theta}) || \pi_{\boldsymbol{\omega}}(\boldsymbol{\theta})\big) \leq \epsilon$$

- Optimize for each parameter (mean, covariance, stepsize) independently
- Can retrieve similar structure then CMA-ES updates

- **Mean:** $$\boldsymbol{\mu}_{\text{new}} = \frac{\eta_\mu \boldsymbol{\mu}_{\text{old}} + \sum_i w^{[i]} \boldsymbol{\theta}^{[i]}}{\eta_\mu + \sum_i w^{[i]}}$$
- **Covarariance:** $$\boldsymbol{\Sigma}_{\text{new}} = \frac{\eta_\Sigma \boldsymbol{\Sigma}_{\text{old}} + \sum_i w^{[i]} \boldsymbol{S}}{\eta_\Sigma + \sum_i w^{[i]}} \qquad \boldsymbol{S} = \underbrace{\frac{\sum_i w^{[i]} (\boldsymbol{\theta}^{[i]} - \boldsymbol{\mu}_{\text{old}})(\boldsymbol{\theta}^{[i]} - \boldsymbol{\mu}_{\text{old}})^T}{\sum_i w^{[i]}}}_{\text{weighted sample covariance}}$$

Update interpolates moments of weighted sample distribution and old distribution!

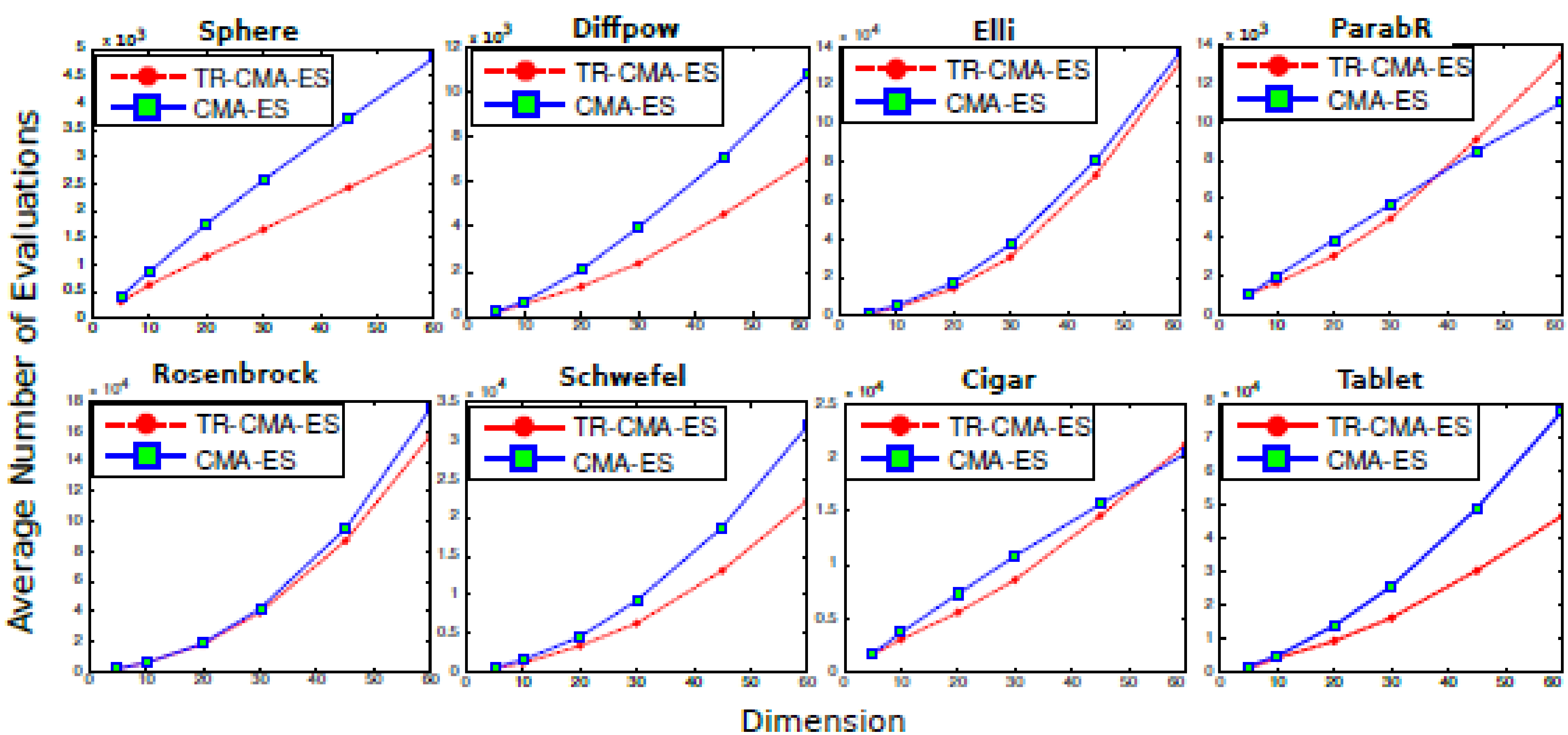
A. Abdolmaleki, B. Price, N. Lau, P. Reis, G. Neumann, Deriving and Improving CMA-ES with information-geometric trust regions, Gecco 2017

Comparison to original CMA-ES

Difference to CMA-ES:

- CMA-ES does not use bound but KL-regularizer
- CMA-ES only uses KL regularizer for covariance
- Mean is just weighted ML, stepsize is based on heuristics

Evaluation on optimization functions

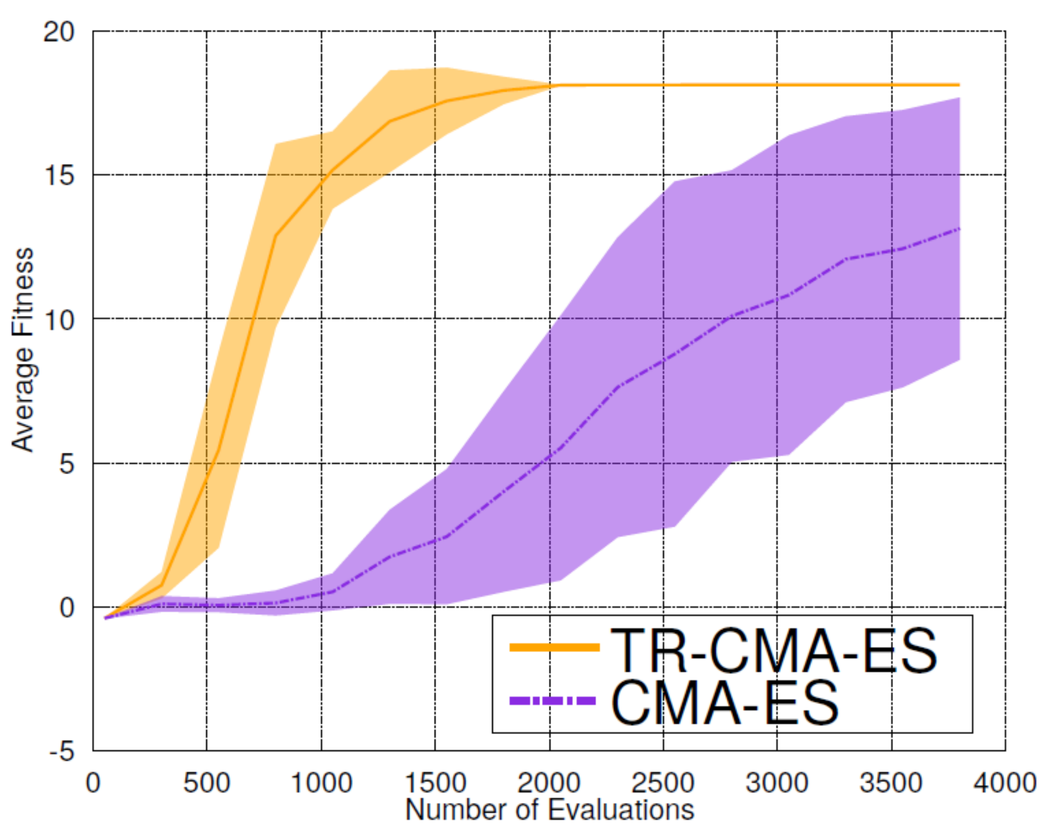


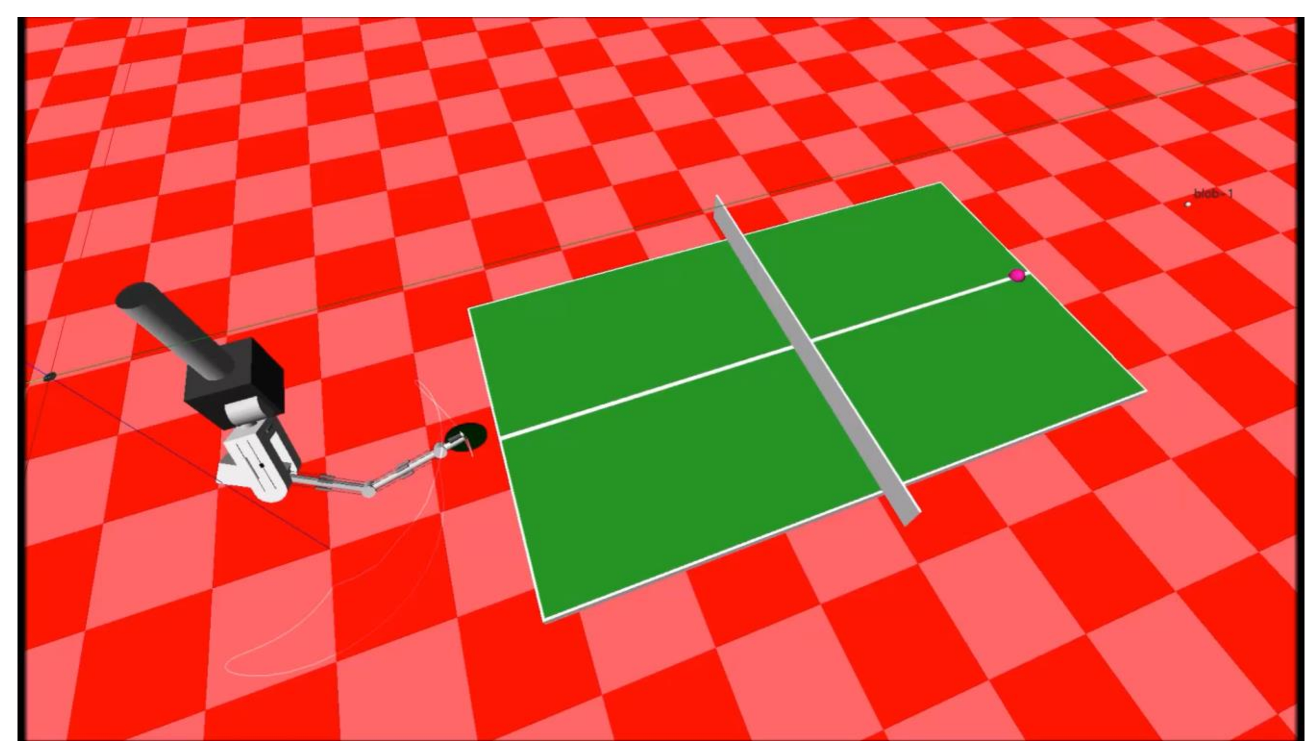
Comparison to original CMA-ES

Difference to CMA-ES:

- Bound is essential for non-continuous performance function

Evaluation on table tennis:





Wrap-up: **Two different objectives**

Average Reward:

- Exact information-gain bound works well
- Can use compatible function approximation

Weighted Log-Likelihood:

- Convex surrogate for average reward
- Exact moment-bound works well

Relations (and combinations) of both still need to be understood

- In the approximate case, both bound formulations are equivalent

Outlook & further reading

Survey papers:

- [Deisenroth, Neumann & Peters: A survey on policy search for robotics, FNT, 2013]
- [Kober, Bagnell & Peters: Reinforcement Learning for Robotics: A survey, IJJR 2013]

Sample-efficient learning from high-dimensional sensory data

- Tactile and vision data [van Hoof 2015][Levine et al. 2016]
- Transfer from simulation to real robots [Russo et al. 2016, Levine et al. 2016a]
- Deep kernel-based methods [Wilson et al. 2016]

Hierarchical Policy Search

- Identify set of re-useable skills [Daniel et al 2016, Bacon et al 2016]
- Learn to select, adapt, sequence and combine these skills [Daniel 2016b, Neumann 2014]
- Deep hierarchical policy search [Bacon et al 2016]

Incorporate human feedback

- Inverse RL and Preference Learning [Finn 2016][Akrour et al. 2013][Wirth et al. 2016,]
- Adverserial imitation learning [Ermon 2016]

Outline

Taxonomy of Policy Search Algorithms

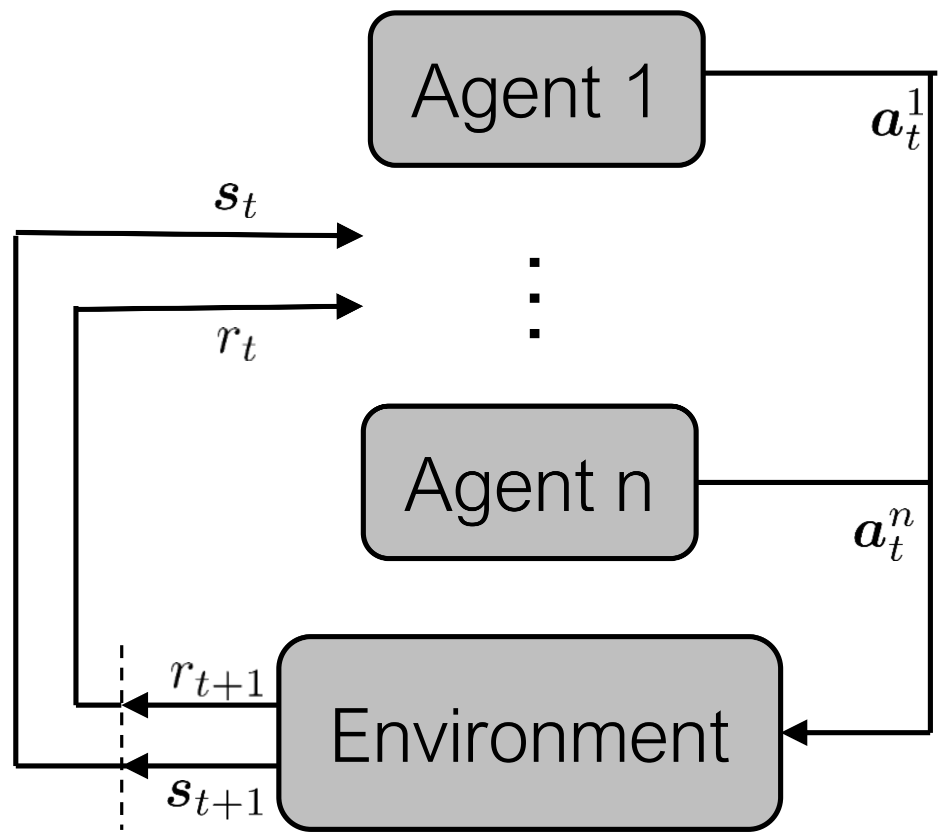
Model-Free Policy Search Methods

- Policy Gradients
- Natural Gradients
- Exact Information Geometric Updates
- Success Matching

Policy Search Methods for Multi-Agent Systems

Reinforcement Learning for Multi-Agent Systems

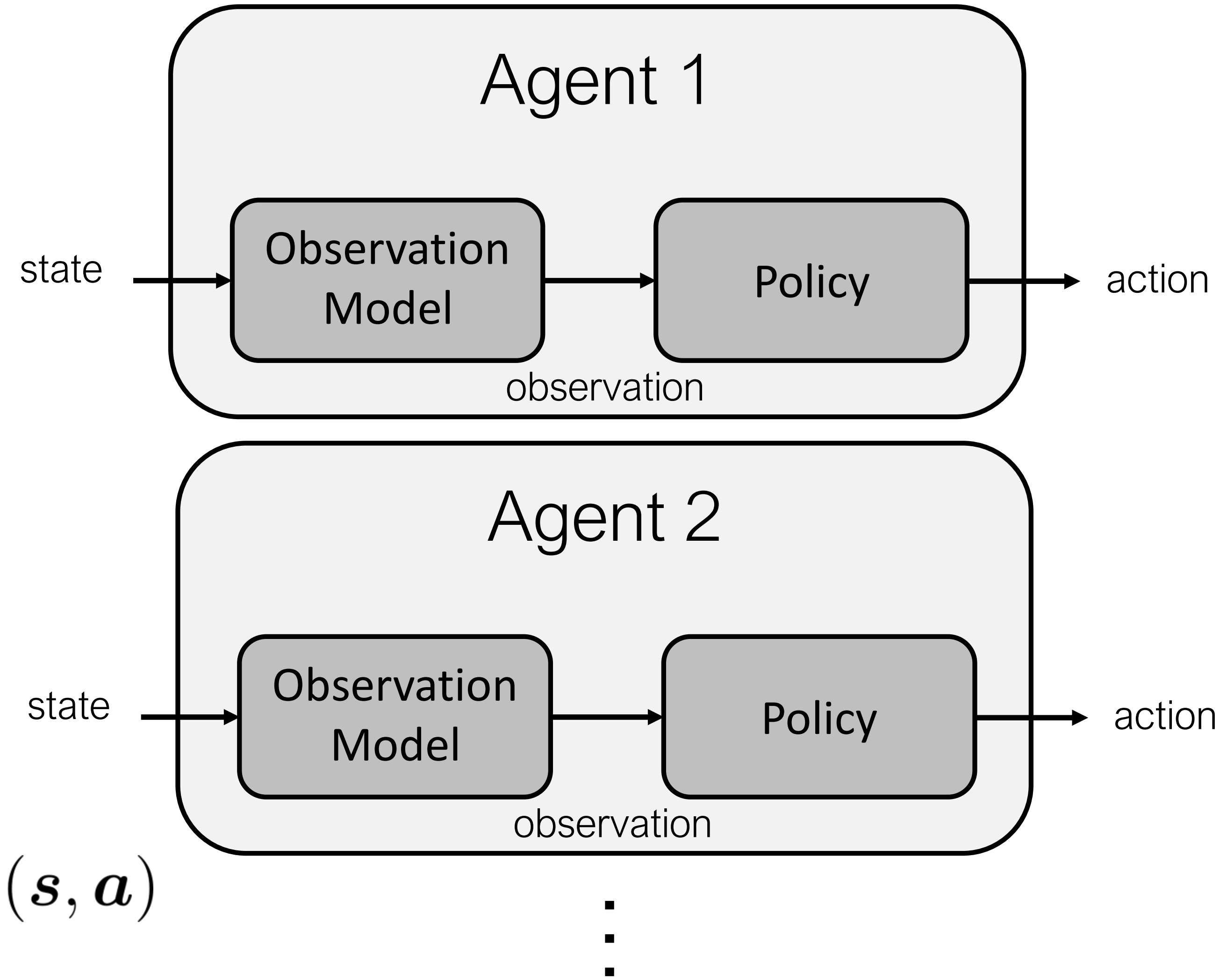
How can we scale such approaches to **multiple agents**?



Decentralized-POMDPs

A Dec-POMDP is defined by:

- its state space $\boldsymbol{s} \in \mathcal{S}$
- An action space $\mathcal{A}_i$ for agent *i*
- An observation space O_i for agent *i*
- its transition dynamics $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$
- observation model per agent $p_i(\boldsymbol{o}|\boldsymbol{s})$
- A shared reward function for all agents $r(\boldsymbol{s}, \boldsymbol{a})$
- and its initial state probabilities $\mu_0(\boldsymbol{s})$



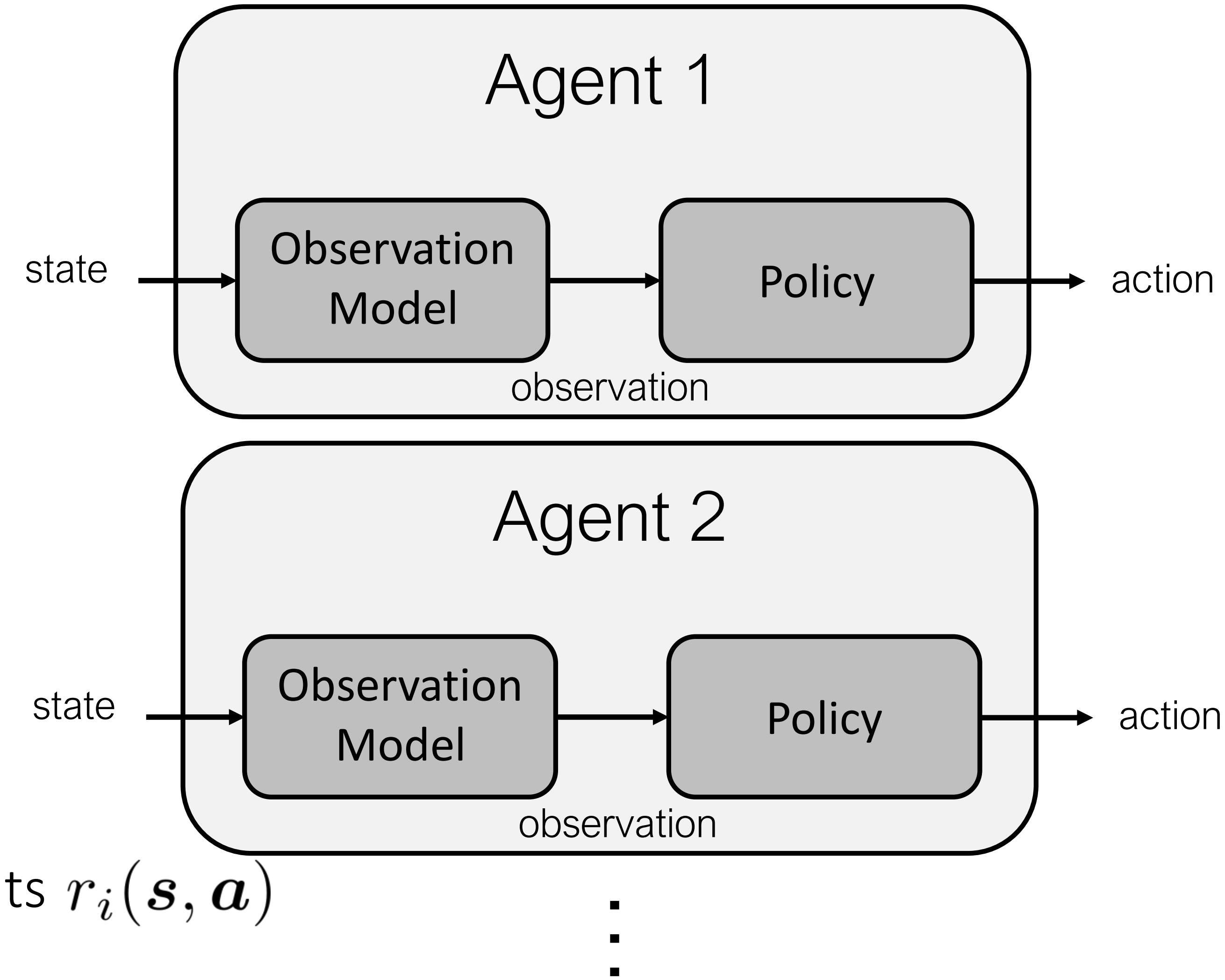
There is a **common goal** (reward): collaborative agents

We do not know what the other agents observed

Partially Observable Stochastic Games (POSG)

A POSG is defined by:

- its state space $\boldsymbol{s} \in \mathcal{S}$
- An action space $\mathcal{A}_i$ for agent *i*
- An observation space O_i for agent *i*
- its transition dynamics $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$
- observation model per agent $p_i(\boldsymbol{o}|\boldsymbol{s})$
- An individual reward function for all agents $r_i(\boldsymbol{s}, \boldsymbol{a})$
- and its initial state probabilities $\mu_0(\boldsymbol{s})$



Competitive agents -> That's the hardest case!

Collaborative vs. Competitive Learning

Collaborative Agents:

- **Increased dimensionality**
- Each agent is only **controlling a subset** of the total action space
- Actions of other agents are **perceived as noise** in the transitions
- Typically **heterogenous**: Agents share the same policy
- **Common goal:** Each agent will find similar policy updates
- **Stable learning** can be achieved

Competitive Agents:

- **Simultan[illegible]s moves:** [illegible]ents do not see moves of [illegible]er age[illegible]mmediatly
- If I change [illegible] poli[illegible] how will **competing agents react**?
- We can use **so[illegible]n concepts from game theory** (e.g. Na[illegible]quilibrium) to get a stable solutio[illegible]
- Computatio[illegible]y **v[illegible] demanding**
- **Inherently [illegible]stable** [illegible]andard reinforce[illegible]nt learni[illegible]s used

Partial observability

How do we deal with local observations?

- For optimal decisions, just the current observation is not enough

Two alternative state representations:

➡ **Belief state:**

Probability distribution over states, given past observations

✓ Compact representation of the agent's knowledge (sufficient statistics)

× Complex to compute, needs a model

➡ **Information state:**

Information state incorporates whole history

✓ Simple

× Very high dimensional ⇨ ✓ Deep Neural Networks

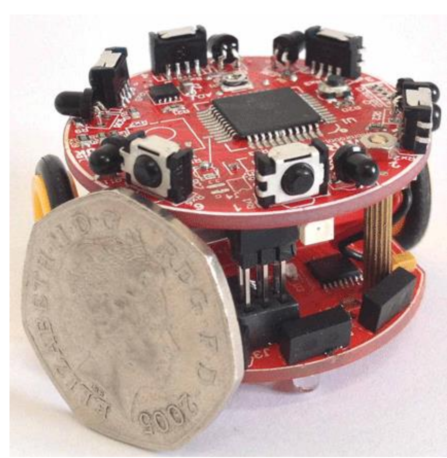
Approximation: Cut history at certain length

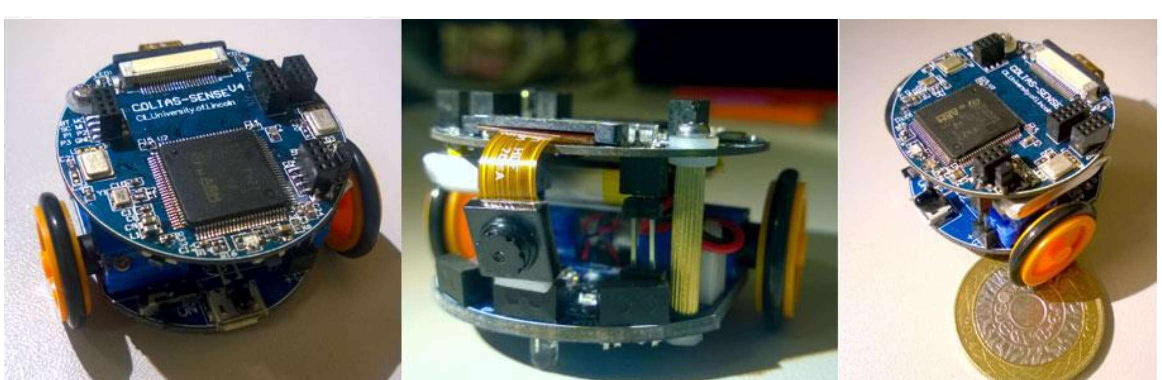
Policy Search for Robot Swarms

Many agents with only local observations

- Ability to accomplish sophisticated tasks (inspired by natural swarms)
- Local observations
- Decentralized decision making
- Learning in swarm systems is very difficult

Robot Platform:





Colias

Deep RL Algorithms

Adaptations for Multi-Agent Learning with Homogeneous Agents

- Policies are shared across agents
- The policy gets the local observation-history as inputs
- **Trust Region Policy Optimization (TRPO):**
  - Use transitions from all agents to estimate gradient
  - Scales well to Deep Neural Networks

Tasks

- Simulations use Box2D for physically correct collision and movement
- Hand-coded communication model includes histograms of distance and bearing to neighbouring agents

Three different tasks:

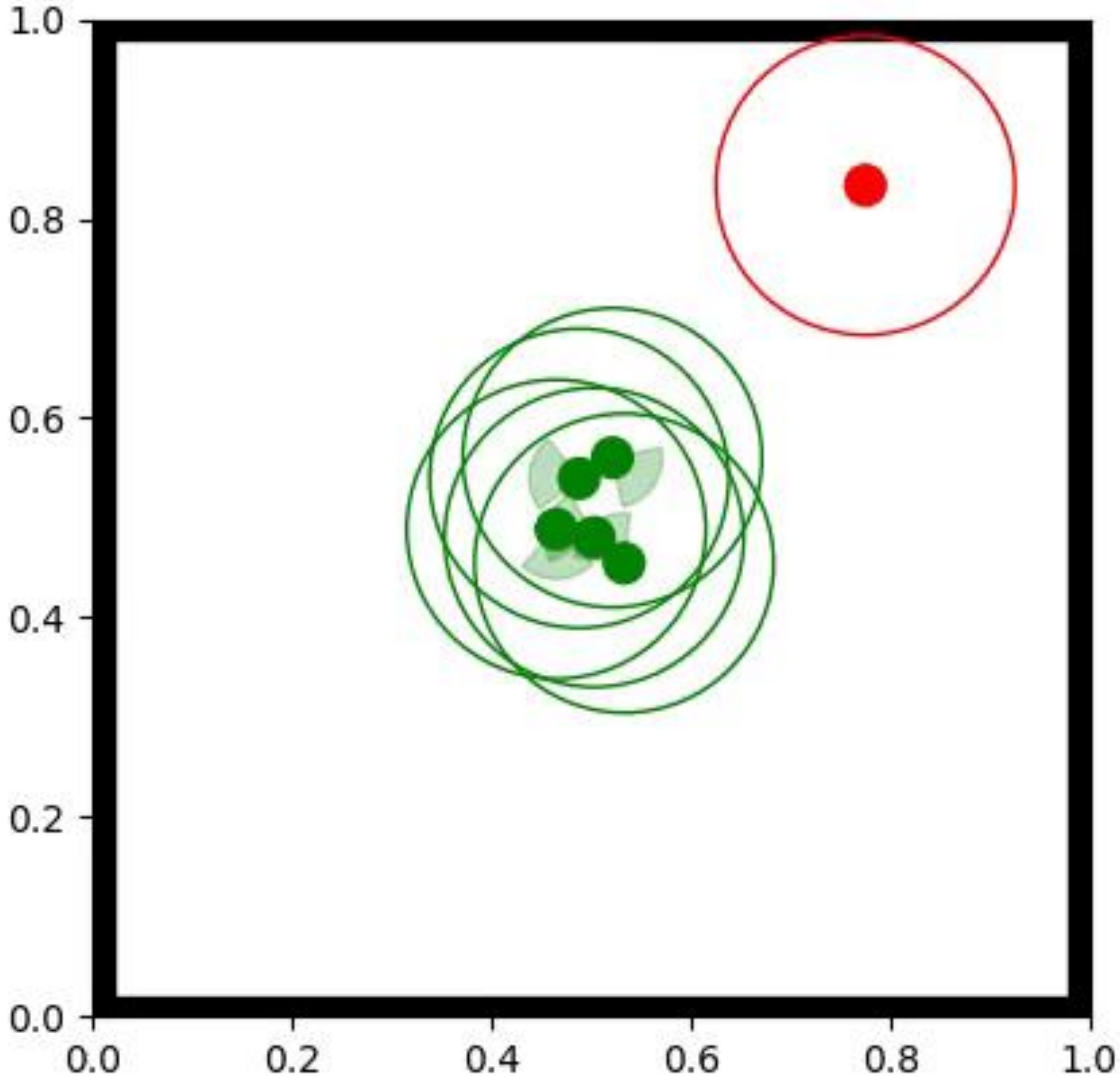
- **Push:** Agents need to learn how to push an intruder away from a simulated light source, added information about intruder
- **Edge:** Agents shall find a constellation to stay within a certain range to each other while avoiding collisions
- **Chain:** Agents shall bridge two points (e.g. a food source and a nest) and keep up the connections, added information about shortest paths

Results: Push Task

- Red agent uses hand-coded phototaxis behaviour to reach center of the world
- Green agents execute learned policy to push red agent as far as possible away from center

Observations:

  - 3 bump sensors for short range collision avoidance
  - distance to red agent if in range
  - Histogram over distances of green agents in range

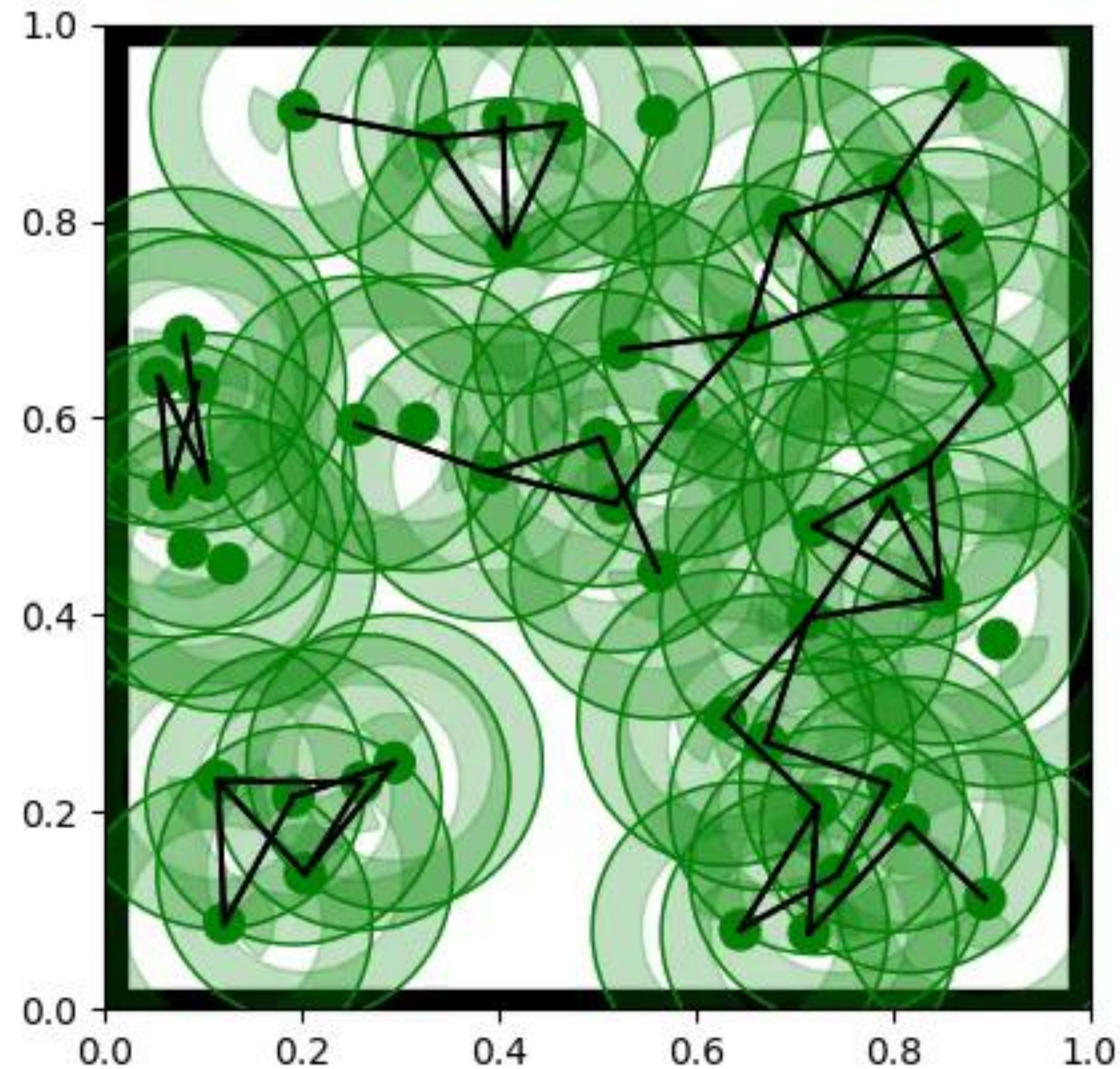


Results: Edge Task

- Agents receive positive reward for each edge they form
- an edge forms if two agents are within the bright green bands
- negative reward for being too close to each other

Observations:

- 3 bump sensors
- 2D histogram over distance/bearing to other agents in range

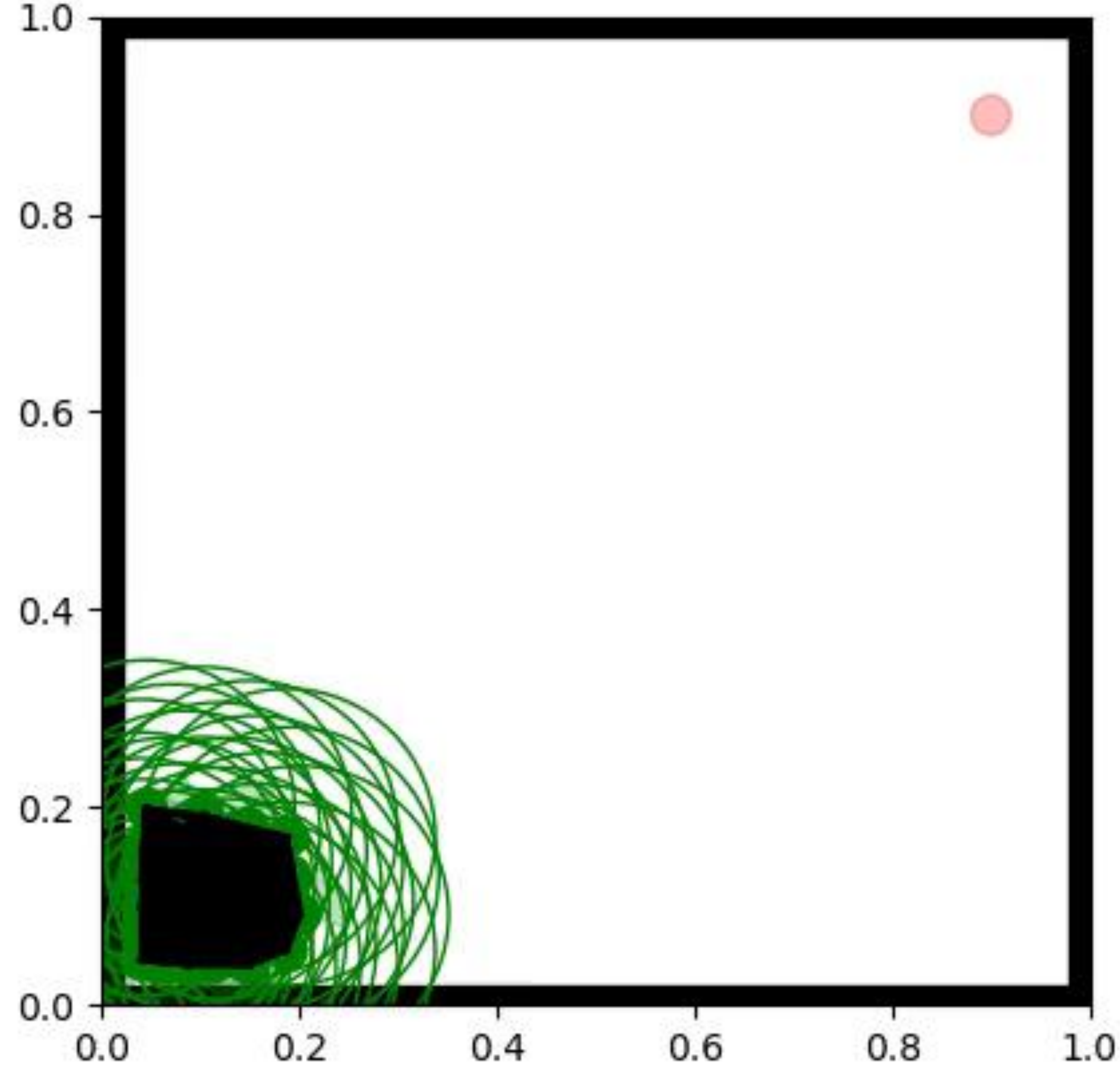


Results: Chain Task

- Agents start at a source and try to find and maintain a link to a sink of some sort

Observations include:

- 3 bump sensors
- Two 2D histograms over distance/bearing to other agents within range
  - 1: Agents seeing source
  - 2: agents seeing sink



Conclusion

Policy Search Methods have made a tremendous development !

Trajectory-based:

- Data efficient learning of rather simple policies
- No feedback
- „Robot-friendly“ exploration

Action-based:

- can learn deep policies
- not sample efficient
- Uncorrelated exploration

Finding the right metric is the key to efficient and robust exploration!

- **Approximate KL bounds:** symmetric, but loose information
- **Information KL bounds:** Suitable for average return formulation
- **Moment KL bounds:** Suiteable for maximum likelihood formulation

Conclusion

Policy Search Methods for Multi-Agent Systems

- Learn complex policies using observation histories
- Deep RL algorithms scale well to the multi-agent case
- They do need millions of examples

Open Problems:

- Learning Communication
- Internal memory
- Specialization of Agents
- Physical Interaction
- Learning with real robots