Policy Search: Methods and Applications

Jan Peters
Gerhard Neumann
Motivation

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

**Today:**
- Industrial Robots

**Tomorrow:**
- Robot Assistants
  - http://news.softpedia.com/
- Nano-Robots
- Dangerous Env.
- Household
  - http://zackkanter.com/
- Robot Athletes
- Transportation
Most of these tasks can not be programmed by hand

**Easier:** Specifying a reward function \( \Rightarrow \) Markov Decision Processes

A Markov Decision Process (MDP) is defined by:

- its state space \( s \in \mathcal{S} \)
- its action space \( a \in \mathcal{A} \)
- its transition dynamics \( \mathcal{P}(s_{t+1}|s_t, a_t) \)
- its reward function \( r(s, a) \)
- and its initial state probabilities \( \mu_0(s) \)
Reinforcement Learning

Most of these tasks can not be programmed by hand

Easier: Specifying a reward function \(\Rightarrow\) Markov Decision Processes

A Markov Decision Process (MDP) is defined by:

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

Learning: Adapting the policy \(\pi(a|s)\) of the agent
Reinforcement Learning

**Objective:** Find policy that maximizes long term reward $J_\pi$

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

**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:
\[ Q(s, a) = r(s, a) + \gamma \mathbb{E}_{P} [V(s') | s, a] \]
- Global estimate for all reachable states
- Hard to scale to high-D
- Approximations might “destroy“ policy

Estimate global policy:
\[ \pi^*(s) = \arg \max_a Q(s, a) \]
- Greedy policy update for all states
- Policy update might get unstable

Explore the whole state space:
\[ \pi(a|s) = \frac{\exp(Q(s, a))}{\sum_{a'} \exp(Q(s, a'))} \]
- 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


Use parametrized policy
\( a \sim \pi(a|s; \theta), \theta \ldots \text{parameter vector} \)
- Compact parametrizations for high-D exists
- Encode prior knowledge

Locally optimal solutions
\( e.g.: \ \theta_{\text{new}} = \theta_{\text{old}} + \alpha \frac{dJ_\theta}{d\theta} \)
- Safe policy updates
- No global value function estimation

Correlated local exploration
\( e.g.: \ \theta_i \sim \mathcal{N}(\theta|\mu_\theta, \Sigma_\theta) \)
- 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., UCAI 2011], [Ghavamzadeh & Mahedevan, ICML 2003]
Used policy representations

Parametrized Trajectory Generators

- Returns a desired trajectory $\tau^*$
  \[ \tau^* = q_{1:T}^* = f(\theta) \]
- Compute controls $u_t$ 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], ...

Other Representations:
- Linear Controllers [Williams et. al., 1992]
- RBF-Networks [Deisenroth & Rasmussen., ICML 2011]
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 et al., 2010], Guided Policy Search by Trajectory Optimization [Levine & Koltun, 2010]
Taxonomy of Policy Search Algorithms

**Model-Free Policy Search**

Use samples

$$\mathcal{D} = \left\{ \left( s_{1:T}^{[i]}, a_{1:T-1}^{[i]}, r_{1:T}^{[i]} \right) \right\}$$

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

$$\mathcal{D} = \left\{ \left( s_{1:T}^{[i]}, a_{1:T-1}^{[i]} \right) \right\}$$

to estimate a model

**Properties:**

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

**model-free vs. model-based**
Taxonomy of Policy Search Algorithms

**Model-Free Policy Search**

Use samples

\[ \mathcal{D} = \left\{ \left( s_{1:T}^{[i]}, a_{1:T-1}^{[i]}, r_{1:T}^{[i]} \right) \right\} \]

to directly update the policy

**Optimization methods:**

- **Expectation Maximization** [Kober & Peters 2008, Vlassis & Toussaint 2009]
- **Path Integral Control** [Theodorou, Buchli & Schaal 2010, Stulp & Sigaud 2012]
- **Stochastic Search Methods** [Hansen 2012, Mannor 2004]

**Model-Based Policy Search**

Use samples

\[ \mathcal{D} = \left\{ \left( s_{1:T}^{[i]}, a_{1:T-1}^{[i]} \right) \right\} \]

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-Algorithmm: 3 basic steps

Repeat

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

\[ \theta_i \sim \pi(\theta; \omega) \]

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

**Evaluate:** quality of parameter vectors \( \theta_i \) by the returns \( R^{[i]} \)

\[ R^{[i]} = \sum_{t=1}^{T} r_t, \quad D = \{ \theta^{[i]}, R^{[i]} \} \]

#### Step-Based

**Explore:** in action-space at each time step

\[ a_t \sim \pi(a|s_t; \theta) \]

- stochastic control policy

**Evaluate:** quality of state-action pairs \((s_t^{[i]}, a_t^{[i]})\) by reward to come

\[ Q_t^{[i]} = \sum_{h=t}^{T} r_h, \quad D = \{ s_t^{[i]}, a_t^{[i]}, Q_t^{[i]} \} \]
## 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 $\theta_i$ 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**


**Step-Based**

- **Explore:** in action-space at each time step
- **Evaluate:** quality of state-action pairs $(s_i^{[i]}, a_i^{[i]})$ by reward to come $Q_i^{[i]}$

**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**

<table>
<thead>
<tr>
<th>Episode-based</th>
<th>Step-Based</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Explore:</strong> in parameter space at the beginning of an episode</td>
<td><strong>Explore:</strong> in action-space at each time step</td>
</tr>
<tr>
<td><strong>Evaluate:</strong> quality of parameter vectors $\theta_i$ by the returns $R[i]$</td>
<td><strong>Evaluate:</strong> quality of state-action pairs $(s_i, a_i)$ by reward to come $Q_i$</td>
</tr>
</tbody>
</table>

#### Algorithms:

**Episode-based**
- 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**
- 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 parameter space at the beginning of an episode
- **Evaluate:** quality of parameter vectors $\theta_i$ by the returns $R_i$

**Hybrid**
- **Explore:** in parameter space at each time step
- **Evaluate:** quality of state-action pairs $(s_t, a_t^i)$ by reward to come $Q_t^{[i]}$

**Properties:**
- State dependent exploration
- Can be reproduced by noise-free policy

**Algorithms:**
- Episodic REPS [CITE]
- PI2-CMA [CITE]
- CMA-ES [CITE]
- NES [CITE]
- PE-PG [CITE]
- Cross-Entropy Search [CITE]

**Step-based**
- **Explore:** in action space at each time step
- **Evaluate:** quality of state-action pairs $(s_t, a_t)$ by reward to come $Q_t$

**Properties:**
- State dependent exploration
- Can be reproduced by noise-free policy

**Algorithms:**
- Reinforce [CITE]
- Policy Gradient Theorem / GPOMDP [CITE]
- Episodic Natural Actor Critic [CITE]
- 2nd Order Policy Gradients [CITE]

More recent versions of these algorithms are episode-based

---

Episode-based vs. step-based algorithms are compared in detail, focusing on their exploration and evaluation strategies. The diagram illustrates the comparison between episode-based and step-based approaches, emphasizing their respective properties and algorithms. The text provides a comprehensive overview of the model-free policy search algorithms, including their exploration and evaluation methods, and highlights their properties and algorithms.
Model-Free Policy Updates

Use samples

\[ D_{ep} = \{ \theta[i], R[i] \} \quad \text{or} \quad D_{st} = \{ s_t[i], a_t[i], Q_t[i] \} \]

to directly update the policy

• **Different optimization methods**
  - 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]

• 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:
  \[
  \theta_i \sim \pi(\theta; \omega) \quad \text{Episode-based}
  \]
  \[
  a_t \sim \pi(a|s_t; \theta) \quad \text{Step-based}
  \]
  - Used for exploration!

- **Efficient Learning:** also update exploration rate!
- E.g. For Gaussian policies:
  \[
  \theta_i \sim \mathcal{N}(\theta|\mu, \Sigma)
  \]
  - Update mean and covariance!
  - Mean \(\mu\) : easy!
  - Covariance \(\Sigma\) : 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

Which policy update should we use?

**Conservative Update**
Small “step size”

**Moderate Update,**
Moderate “step size”

**Greedy update**
Large “step size”
Illustration of Policy Updates

Conservative

Iteration 1

small step-size $\rightarrow$ high exploration $\rightarrow$ slow convergence

Moderate

Iteration 1

step-size about right $\rightarrow$ moderate exploration $\rightarrow$ fast convergence

Greedy Update

Iteration 1

large step-size $\rightarrow$ exploration vanishes $\rightarrow$ premature convergence
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

\[ \mathcal{D}_{ep} = \{ \theta^{[i]}, R^{[i]} \} \quad \text{or} \quad \mathcal{D}_{st} = \{ s_t^{[i]}, a_t^{[i]}, Q_t^{[i]} \} \]

\[ \frac{\partial J_\theta}{\partial \omega} = \nabla_\omega J_\omega \quad \text{or} \quad \frac{\partial J_\theta}{\partial \theta} = \nabla_\theta J_\theta \]

- Update policy parameters in the direction of the gradient

\[ \omega_{k+1} = \omega_{k+1} + \alpha \nabla_\omega J_{\omega_k} \quad \text{or} \quad \theta_{k+1} = \theta_k + \alpha \nabla_\theta J_{\theta_k} \]

- \( \alpha \) ... learning rate
Likelihood Policy Gradients

**Episode-Based:** Policy $\theta \sim \pi(\theta; \omega)$

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:**

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

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

$$\approx \sum_{i=1}^{N} \nabla_\omega \log \pi(\theta_i; \omega) R_i$$

- Only needs samples!
- This gradient is called Parameter Exploring Policy Gradient (PGPE) [Rückstiess et al., 2009]
We can always **subtract a baseline** $b$ from the gradient...

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

**Why?**

- The gradient estimate can have a high variance
- Subtracting a baseline can reduce the variance
- It’s still unbiased...

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

**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

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

... 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 \( \nabla_\theta J \) for the low-level policy \( \pi(a \mid s; \theta) \)
Step-based Policy Gradient Methods

Plug in the **temporal structure** of the RL problem

- Trajectory distribution:
  \[
p(\tau; \theta) = p(s_1) \prod_{t=1}^{T} \pi(a_t|s_t; \theta)p(s_{t+1}|s_t, a_t)
\]

- Return for a single trajectory:
  \[
  R(\tau) = \sum_{t=1}^{T} r_t
  \]

⇒ Expected long term reward \( J_\theta \) can be written as expectation over the trajectory distribution

\[
J_\theta = \mathbb{E}_{p(\tau; \theta)}[R(\tau)] = \int p(\tau; \theta)R(\tau)d\tau
\]
Step-Based Likelihood Ratio Gradient

Using the log-ratio trick, we arrive at

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

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

$$p(\tau; \theta) = p(s_1) \prod_{t=1}^{T} \pi(a_t | s_t; \theta) p(s_{t+1} | s_t, a_t)$$

$$\log p(\tau; \theta) = \sum_{t=1}^{T} \log \pi(a_t | s_t; \theta) + \text{const}$$

Model-dependent terms cancel due to the derivative

$$\nabla_\theta \log p(\tau; \theta) = \sum_{t=1}^{T} \nabla_\theta \log \pi(a_t | s_t; \theta)$$
Step-Based Policy Gradients

Plug it back in...

\[ \nabla_{\theta} J = \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_{\theta} \log \pi(a_t^{[i]} | s_t^{[i]} ; \theta) R(\tau) \]

\[ = \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_{\theta} \log \pi(a_t^{[i]} | s_t^{[i]} ; \theta) \left( \sum_{t=1}^{T} r_t^{[i]} \right) \]

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

- Wait... we still use the returns \( R(\tau) \)
  
  ➤ 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

\[ \mathbb{E}_{p(\tau)}[r_t \log \pi(a_h|s_h)] = 0, \forall t < h \]

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

\[
\nabla_{\theta}^{PG} J = \sum_{i=1}^{N} \sum_{t=1}^{T-1} \nabla_{\theta} \log \pi(a_t^{[i]}|s_t^{[i]}; \theta) \left( \sum_{h=t}^{T-1} r_h^{[i]} + r_T^{[i]} \right) \\
= \sum_{i=1}^{N} \sum_{t=1}^{T-1} \nabla_{\theta} \log \pi(a_t^{[i]}|s_t^{[i]}; \theta) Q_h^{[i]}
\]

- 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 $\alpha$?

Metric used for policy gradients:

- Standard gradients use euclidian distance in parameter space as metric
- Episode-based: $L_2(\pi_{k+1}, \pi_{k}) = \|\omega_{k+1} - \omega_{k}\|$
- Step-based: $L_2(\pi_{k+1}, \pi_{k}) = \|\theta_{k+1} - \theta_{k}\|$

- Invariance to reward transformations
  - Choose learning rate, such that $L_2(\pi_{k+1}, \pi_{k}) \leq \epsilon$
  - Resulting learning rate: $\alpha_k = \frac{1}{\|\nabla J\|} \epsilon$

- No Invariance to parameter transformations
- Euclidian metric can not capture the greediness of the update
We need to find a better metric...

**Policies are probability distributions**

⇒ We can measure „distances“ of distributions

**Better Metric:** Relative Entropy or Kullback-Leibler divergence

\[
KL(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}
\]

- Information-theoretic „distance“ measure between distributions

- **Properties:**
  - Always larger 0: \( KL(p||q) \geq 0 \)
  - Only 0 iff both distributions are equal: \( KL(p||q) = 0 \iff p = q \)
  - Not symmetric, so not a real distance: \( KL(p||q) \neq KL(q||p) \)
Kullback-Leibler Divergences

2 types of KLs that can be minimized:

**Moment projection:**

\[ \argmin_p KL(q||p) = \argmin_p \sum_x q(x) \log \frac{q(x)}{p(x)} \]

- \( p \) is large wherever \( q \) 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:** \( \arg\min_p \text{KL}(p||q) = \arg\min_p \sum_x p(x) \log \frac{p(x)}{q(x)} \)

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

\[ \text{KL}(p_{\theta+\Delta\theta}||p_{\theta}) \approx \Delta\theta^T G(\theta) \Delta\theta \]

where \( G(\theta) \) is the Fisher information matrix (FIM)

\[ G(\theta) = E_p[\nabla_\theta \log p_\theta(x) \nabla_\theta \log p_\theta(x)^T] \]

- Captures information how a single parameter influences the distribution
The **natural gradient** [Amari 1998] uses the Fisher information matrix as metric

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

\[ \nabla_{\theta}^{NG} J = \arg\max_{\Delta \theta} \Delta \theta^T \nabla_{\theta} J \]

s.t.: \[ \text{KL}(p_{\theta + \Delta \theta} || p_\theta) \approx \Delta \theta^T G(\theta) \Delta \theta \leq \epsilon \]

The solution to this optimization problem is given as:

\[ \nabla_{\theta}^{NG} J \propto G(\theta)^{-1} \nabla_{\theta} J \]

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

Linear Quadratic Regulation

\[ x_{t+1} = Ax_t + Bu_t \]
\[ u_t \sim \pi(u|x_t) = N(u|kx_t, \sigma) \]
\[ r_t = -x_t^T Q x_t - u_t^T Ru_t \]

Two-State Problem

\[ u = 0, r = 0 \]
\[ u = 0 \]
\[ r = 0 \]

\[ u = 1 \]
\[ r = 1 \]
\[ u = 1 \]
\[ r = 2 \]

The standard gradient reduces the exploration too quickly!

[Peters et al. 2003, 2005]
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

\[ \nabla_{\theta}^{PG} J = \sum_{i=1}^{N} \sum_{t=1}^{T-1} \nabla_{\theta} \log \pi(a_t^{[i]} | s_t^{[i]} ; \theta) (Q_h^{[i]} - b_h(s)) \]

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

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

and use \[ \nabla_{\theta}^{FA} J = \sum_{i=1}^{N} \sum_{t=1}^{T-1} \nabla_{\theta} \log \pi(a_t^{[i]} | s_t^{[i]} ; \theta) f_w(s^{[i]}, a^{[i]}) \]

as gradient

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

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

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

The compatible function approximation is mean-zero!

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

- 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

\[ f_w(s, a) = \nabla_\theta \log \pi(a|s; \theta)^T w = Q^\pi(s, a) - V^\pi(s) \]
Can the Compatible FA be learned?

The compatible function approximation represents an advantage function \[ f_w(s, a) = Q^\pi(s, a) - V^\pi(s) = A^\pi(s, a) \]

The advantage function is very different from the value functions.

In order to learn \( f_w(s, a) \) we need to learn \( V^\pi(s) \).
Compatible Function Approximation

Gradient with Compatible Function Approximation:

\[
\nabla_{\theta}^{FA} J = \sum_{i=1}^{N} \sum_{t=1}^{T-1} \nabla_{\theta} \log \pi(a_t^{[i]} \, | \, s_t^{[i]} ; \theta) \nabla_{\theta} \log \pi(a_t^{[i]} \, | \, s_t^{[i]} ; \theta)^T w \\

\nabla_{\theta}^{FA} J = E_{p(\tau)} \left[ \nabla_{\theta} \log \pi(a_t^{[i]} \, | \, s_t^{[i]} ; \theta) \nabla_{\theta} \log \pi(a_t^{[i]} \, | \, s_t^{[i]} ; \theta)^T \right] w \\

\nabla_{\theta}^{FA} J = F(\theta) w \\
\]

It can be shown that [Peters & Schaal, 2008] :

\[
F(\theta) = E_{p(\tau)} \left[ \nabla_{\theta} \log \pi(a_t^{[i]} \, | \, s_t^{[i]} ; \theta) \nabla_{\theta} \log \pi(a_t^{[i]} \, | \, s_t^{[i]} ; \theta)^T \right] \\
= E_{p(\tau)} \left[ \nabla_{\theta} \log p(\tau ; \theta) \nabla_{\theta} \log p(\tau ; \theta)^T \right] = G(\theta) 
\]
Connection to V-Function approximation

Lets put the parts together:

- **Combatible Function Approximation:**

  \[ \nabla^\text{FA}_\theta J = F(\theta)w \]

- [Peters & Schaal, 2008] showed: \( F \) is the Fisher information matrix!

  \[ F(\theta) = G(\theta) \]

- That makes the natural gradient very simple!

  \[ \nabla^\text{NG}_\theta J = G(\theta)^{-1} \nabla^\text{FA}_\theta J = G(\theta)^{-1} F(\theta)w = w \]

So we just have to learn \( w \)
What about this additional FA?

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

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

$$Q^\pi_1(s^{[i]}_1, a^{[i]}_1) = r(s^{[i]}_1, a^{[i]}_1) + V^\pi_2(s^{[i]}_2)$$

$$V^\pi_1(s^{[i]}_1) + f_w(s^{[i]}_1, a^{[i]}_1) = r(s^{[i]}_1, a^{[i]}_1) + V^\pi_2(s^{[i]}_2)$$

$$V^\pi_1(s^{[i]}_1) + \nabla_\theta \log \pi(a^{[i]}_1 | s^{[i]}_1; \theta)w = r(s^{[i]}_1, a^{[i]}_1) + V^\pi_2(s^{[i]}_2)$$

for each time step

$$V^\pi_1(s^{[i]}_1) + \nabla_\theta \log \pi(a^{[i]}_1 | s^{[i]}_1; \theta)w = r(s^{[i]}_1, a^{[i]}_1) + V^\pi_2(s^{[i]}_2) \quad | + \text{ both sides}$$

$$V^\pi_2(s^{[i]}_2) + \nabla_\theta \log \pi(a^{[i]}_2 | s^{[i]}_2; \theta)w = r(s^{[i]}_2, a^{[i]}_2) + V^\pi_3(s^{[i]}_3) \quad | + \text{ both sides}$$

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

$$V^\pi_{T-1}(s^{[i]}_{T-1}) + \nabla_\theta \log \pi(a^{[i]}_{T-1} | s^{[i]}_{T-1}; \theta)w = r(s^{[i]}_{T-1}, a^{[i]}_{T-1}) + V^\pi_T(s^{[i]}_T)$$
What about this additional FA?

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

$$V^\pi(s_1^{[i]}) + \left( \sum_{t=1}^{T-1} \nabla_\theta \log \pi(a_t^{[i]}|s_t^{[i]}; \theta) \right) \phi_T^{[i]} w = \sum_{t=1}^T r(s_t^{[i]}, a_t^{[i]})$$

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 \( \mathbf{w} \) we can use linear regression

\[
V_\pi(s_1) + \left( \sum_{t=1}^{T-1} \nabla_\theta \log \pi(\mathbf{a}_t^{[i]} | s_t^{[i]}; \theta) \right) \mathbf{w} = \sum_{t=1}^{T} r(s_t^{[i]}, a_t^{[i]})
\]

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

\[
R = \begin{bmatrix} R_1, & R_2^T, & \ldots, & R_N^T \end{bmatrix}^T
\]

\[
\begin{bmatrix} \mathbf{w} \\ J \end{bmatrix} = \left( \Phi^T \Phi \right)^{-1} \Phi^T R
\]

Actor: Natural Policy Gradient Improvement

\[
\theta_{t+1} = \theta_t + \alpha_t \mathbf{w}_t.
\]

Critic: Episodic Evaluation
Results...

**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 $f(r)$

$$\pi_{new}(a|s) \propto f(r(s, a))\pi_{old}(a|s)$$
**Success Matching Principle**

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

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

Can be derived in many ways:

- Expectation maximization [Kober & Peters., 2008][Vlassis & Toussaint., 2009]
- Optimal Control [Theodorou, Buchli & Schaal, 2010]

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 Success Matching

Iterate:

Sample and evaluate parameters:

\[ \theta[i] \sim \pi(\theta; \omega_k) \]

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

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

\[ p_k(\theta[i]) \propto w[i] \pi(\theta[i]; \omega_k) \]

Fit new parametric policy \( \pi(\theta[i]; \omega_{k+1}) \) that best approximates \( p_k(\theta[i]) \)
So where are the weights $w^i = f(R^i)$ coming from?

Transform the returns in an improper probability distribution

**Exponential transformation** [Peters 2005]:

$$w^i = \exp(\beta(R^i - \max R^i))$$

- $\beta$ . . . 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 \( \pi(\theta; \omega_{k+1}) \) that best fits the distribution \( p(\theta^{[i]}_i) \propto w^{[i]}_i \pi(\theta^{[i]}_i; \omega_k) \),

We can do that by computing the M-projection of \( p(\theta^{[i]}_i) \):

\[
\omega_{k+1} = \text{argmin}_\omega \quad \text{KL}(p(\theta^{[i]}_i) || \pi(\theta^{[i]}_i; \omega))
\]

\[
= \text{argmin}_\omega \quad \int p(\theta) \log \frac{p(\theta)}{\pi(\theta; \omega)} d\theta
\]

\[
\approx \text{argmax}_\omega \quad \sum_i \frac{p(\theta^{[i]}_i)}{\pi(\theta^{[i]}_i; \omega_k)} \log \pi(\theta^{[i]}_i; \omega)
\]

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: $\pi(\theta; w) = \mathcal{N}(\theta | \mu, \Sigma)$

Weighted mean: $\mu = \frac{\sum_i w[i] \theta[i]}{\sum_i w[i]}$

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

• But more general: Also for mixture models, GPs and so on...
• Invariant to transformations of the parameters
Underactuated Swing-Up

swing heavy pendulum up

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

- 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 \cite{KoberPeters08}

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

Policy: DMPs

\[ d \]
Ball-in-a-Cup
Table Tennis [Mülling, Kober, Krömer & Peters, 2013]

Initial Policy after Imitation Learning

Success Rate 69 %
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 $\beta$
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:

\[
\max_{\pi} \sum_{i} \pi(\theta[i]) R(\theta[i]) \quad \text{Maximize Reward}
\]

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

\[
\sum_{i} \pi(\theta[i]) = 1 \quad \text{It’s a distribution}
\]

- Stay close to the data
- Epsilon directly controls the exploration-exploitation trade-off
  - \( \epsilon = 0 \ldots \) continue to explore with policy \( q(\theta) \)
  - \( \epsilon \to \infty \ldots \) greedily jump to best sample
Relative Entropy Policy Search

Which has the following analytic solution:

$$\pi(\theta) \propto q(\theta) \exp \left( \frac{R_\theta}{\eta} \right)$$

- That’s exactly success matching with exponential transformation!
- **Scaling factor** $\eta = 1/\beta$:
  - Automatically chosen from optimization (Lagrange Multiplier)
  - Specified by KL-bound $\epsilon$
- How to compute $\eta$?
  - 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 $x$ describes objectives of the task (fixed before task execution)
- E.g.: Target location to throw a ball
- Adapt the control policy parameters $\theta$ to the target location $x$
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 $\theta$ to the target location $x$
- Learn an upper level policy $\pi(\theta|x; \omega)$

Objective:

$$J_\pi = \int \int \mu_0(x)\pi(\theta|x)R_{x\theta} dx d\theta$$

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

Dataset for policy update:

$$D_{ep} = \{\theta[i], x[i], R[i]\}$$

- Also contains context vectors
Contextual Policy Search with REPS
[Kupscik, Deisenroth, Peters & Neumann, 2013]

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

- Otherwise independent optimization problems for each context

We get the following optimization problem [CITE]:

\[
\max_p \sum_{x, \theta} p(x, \theta) R(x, \theta) \quad \text{maximize rewards}
\]

\[
\text{s.t.: } \sum_{x, \theta} p(x, \theta) = 1 \quad \text{it's a distribution}
\]

\[
\text{KL}(p(x, \theta) \| q(x, \theta)) \leq \epsilon \quad \text{stay close to the data}
\]

\[
\forall x \quad p(x) = \sum_\theta p(x, \theta) = \mu_0(x) \quad \text{reproduce given context distribution } \mu_0(x)
\]
Closed form solution:

\[ p(x, \theta) \propto q(x, \theta) \exp \left( \frac{R x \theta - V(x)}{\eta} \right) \]

• We automatically get a **baseline** \( V(x) \) for the returns

• **Function approximation** for \( V(x) \) achieved by matching feature averages instead of distributions

\[
\sum_x p(x) \phi(x) = \hat{\phi} \quad \implies \quad V(x) = \phi^T(x)v
\]

• \( v \ldots \) given by Lagrangian multipliers

• Obtain \( v \) again by optimizing the dual

Policy \( \pi(\theta|x; \omega_{k+1}) \) 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

<table>
<thead>
<tr>
<th>Player</th>
<th>Hit rate</th>
<th>Matches won</th>
<th>Total score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analytical</td>
<td>71%</td>
<td>6/25</td>
<td>8</td>
</tr>
<tr>
<td>Learned</td>
<td>85%</td>
<td>19/25</td>
<td>38</td>
</tr>
</tbody>
</table>

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 $\pi(\theta|x)$ as hierarchical policy

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

$$\pi(\theta|x) = \sum_o \pi(o|x)\pi(\theta|x, o)$$
Learning versatile Sub-Policies

Sub-Policies should represent distinct solutions.

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

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

Entropy
Hierarchical REPS

Bounding the overlap of sub-policies:

Learning of versatile, distinct solutions due to separation of sub-policies.
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
\[ \mathcal{D} = \left\{ \left( s_1^{[i]}, a_1^{[i]} \right) \right\} \rightarrow \hat{P}(s'|s,a) \approx P(s'|s,a) \]

+ 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]

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

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

- 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]
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 $p(s_1|\theta) \ldots p(s_T|\theta)$
  $p(s_t|\theta) = \int \widehat{P}(s_t|s_{t-1}, \pi(s; \theta)) p(s_{t-1}|\theta) ds_{t-1}$
  $\text{GP prediction} \quad N(\mu_t, \Sigma_t)$
• Moment matching: deterministic approximate inference

Policy Update:
• Analytically compute expected return and its gradient
• Greedily Optimize with BFGS

\[ J_{\theta, \hat{\rho}} = \sum_{t=1}^{T} \int p(x_t|\theta)r(x_t)dx_t \]
\[ \theta_{\text{new}} = \arg\min_{\theta} J_{\theta, \hat{\rho}} \]
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)

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

\[
\arg\max_{\pi} \mathbb{E}_{\hat{P},\pi}\left[ \sum_{t=1}^{T} r(s_t, a_t) \right], \quad \text{s.t.}: \ KL(\pi \parallel q) \leq \epsilon
\]

⇒ **Ideas from model-free PS directly transfer**

• **Exploration:** Step-size of the policy update is bounded
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 $\mathcal{P}(s'|s, \alpha)$ following policy $\pi(s; \theta)$
- Generate a high number of trajectories for different parameter vectors $\theta$ and context vectors $x$

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

$$\arg \max_\pi \quad \mathbb{E}_{\hat{P},\pi} [R_x \theta],$$
$$\text{s.t.: } \text{KL}(\pi(\theta|x)\|q(\theta|x)) \leq \epsilon$$
Table tennis experiment
[Kupcsik, Deisenroth, Peters & Neumann et al. 2015]

19 Policy Parameters (DMPs)
5 context variables (initial ball velocities, desired target location)
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]
Infinite Horizon Formulation

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

$$\max_\pi \int \int \mu^\pi(s)\pi(a|s)r(s, a)dsda$$

Maximize average reward

s.t.: $\epsilon \geq KL(\mu^\pi(s)\pi(a|s)||q(s, a))$

KL should be bounded to old state action distribution

$$1 = \int \int \pi(a|s)\mu^\pi(s)dsda$$

It’s a distribution

$\forall s', \mu^\pi(s') = \int \int \mu^\pi(s)\pi(a|s)\mathcal{P}(s'|s, a)dsda$

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

Closed form solution:

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

- We automatically get a softmax over the advantage function

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

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

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
Bounded Trajectory Optimization

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

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

Maximize average reward

s.t.: $\epsilon \geq KL(p^\pi(\tau)||q(\tau))$  

KL should be bounded to old trajectory distribution

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

It’s a distribution
Bounded Trajectory Optimization

Plugging in the factorization of the trajectory distribution:

\[ \max_{\pi} \int \int \mu^t_1(s) \pi_t(a \mid s) r_t(s, a) ds da \]

Maximize average reward

s.t.: \( \forall t : \epsilon \geq \mathbb{E}_{\mu^t_1} \left[ \text{KL}(\pi_t(a \mid s) \| q_t(a \mid s)) \right] \)

KL on the policies should be bounded at each time step

\( \forall t \forall s : 1 = \int \pi_t(a \mid s) da \)

It’s a distribution

\( \forall s' \forall t : \mu^t_{t+1}(s') = \int \int \mu^t_1(s) \pi_t(a \mid s) \mathcal{P}_t(s' \mid s, a) ds da \)

Time-dependent state distributions need to be consistent

\( \forall s : \mu^t_1(s) = \mu_1(s), \forall s \)

Initial distribution is given
Infinite Horizon Formulation

Closed form solution:

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

- \(V(s)\)... Lagrangian multiplier,
  - can be computed by dynamic programming

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

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

\[
\pi_t(a|s) = \mathcal{N}(a|K_t s + k_t, \Sigma_t)
\]

- 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