



Reinforcement Learning

Lecture 18: Policy Optimization

24 Oct 2025

Instructor: Raunak P. Bhattacharyya



ScAI

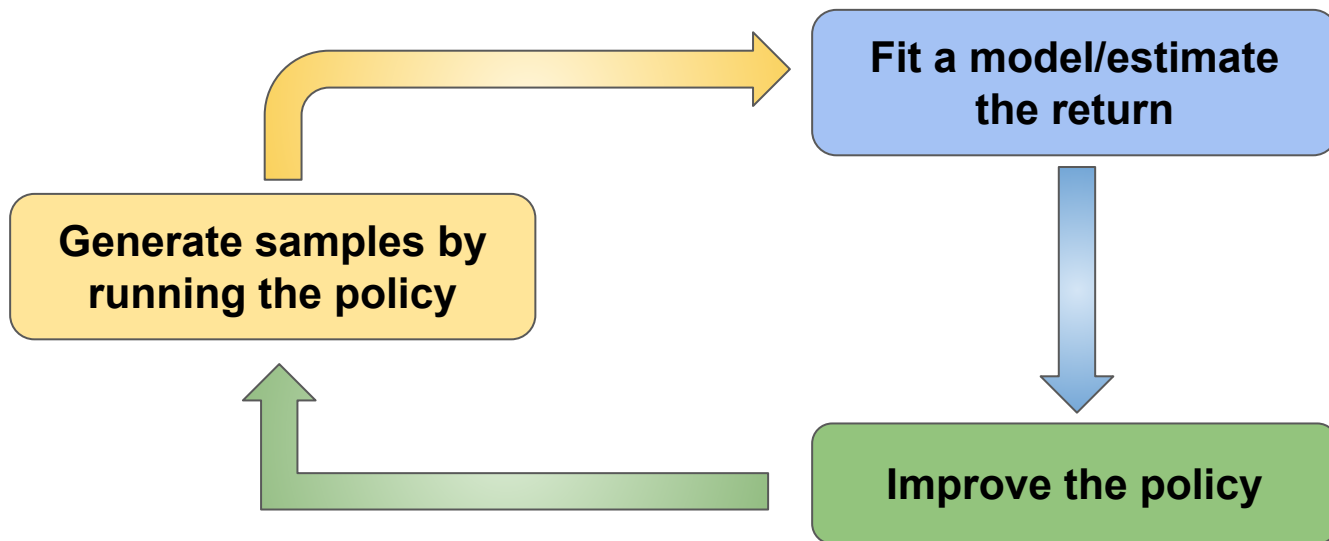
YARDI SCHOOL OF ARTIFICIAL INTELLIGENCE
INDIAN INSTITUTE OF TECHNOLOGY DELHI

Recap

- Deep Q Networks
- Target Nets and Exp Replay
- Double Q Learning

Unified View

$$Q(s, a) \leftarrow r(s, a) + \gamma \cdot \mathbb{E}_{p(s'|s, a)} \left[V^\pi(s') \right]$$



$$V(s) \leftarrow \max_a Q(s, a)$$

Other Problems



[Source](#)



[Source](#)



[Source](#)

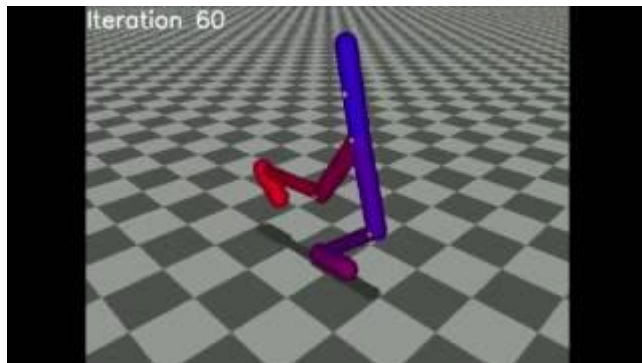
Examples



[Source](#)



[Source](#)

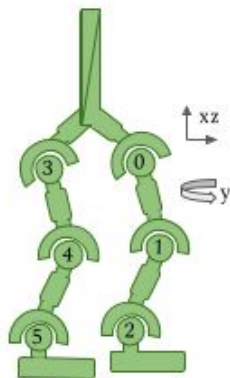
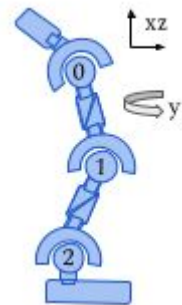


[Source](#)



[Source](#)

Hopper & Walker



Legend



axis

Hinge joint



Slide joint (global)



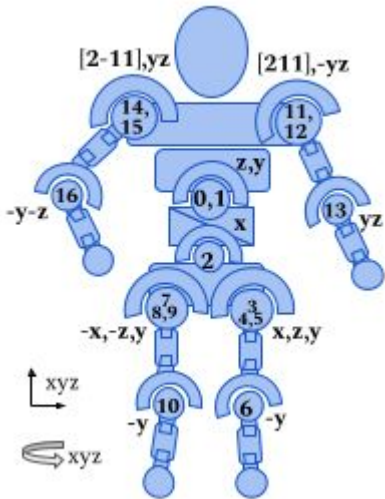
Hinge joint (fixed)



Hinge joint (global)

Num	Action	Control Min	Control Max	Name (in corresponding XML file)	Joint	Type (Unit)
0	Torque applied on the thigh rotor	-1	1	thigh_joint	hinge	torque (N m)
1	Torque applied on the leg rotor	-1	1	leg_joint	hinge	torque (N m)
2	Torque applied on the foot rotor	-1	1	foot_joint		

Num	Action	Control Min	Control Max	Name (in corresponding XML file)
0	Torque applied on the thigh rotor	-1	1	thigh_joint
1	Torque applied on the leg rotor	-1	1	leg_joint
2	Torque applied on the foot rotor	-1	1	foot_joint
3	Torque applied on the left thigh rotor	-1	1	thigh_left_joint
4	Torque applied on the left leg rotor	-1	1	leg_left_joint
5	Torque applied on the left foot rotor	-1	1	foot_left_joint



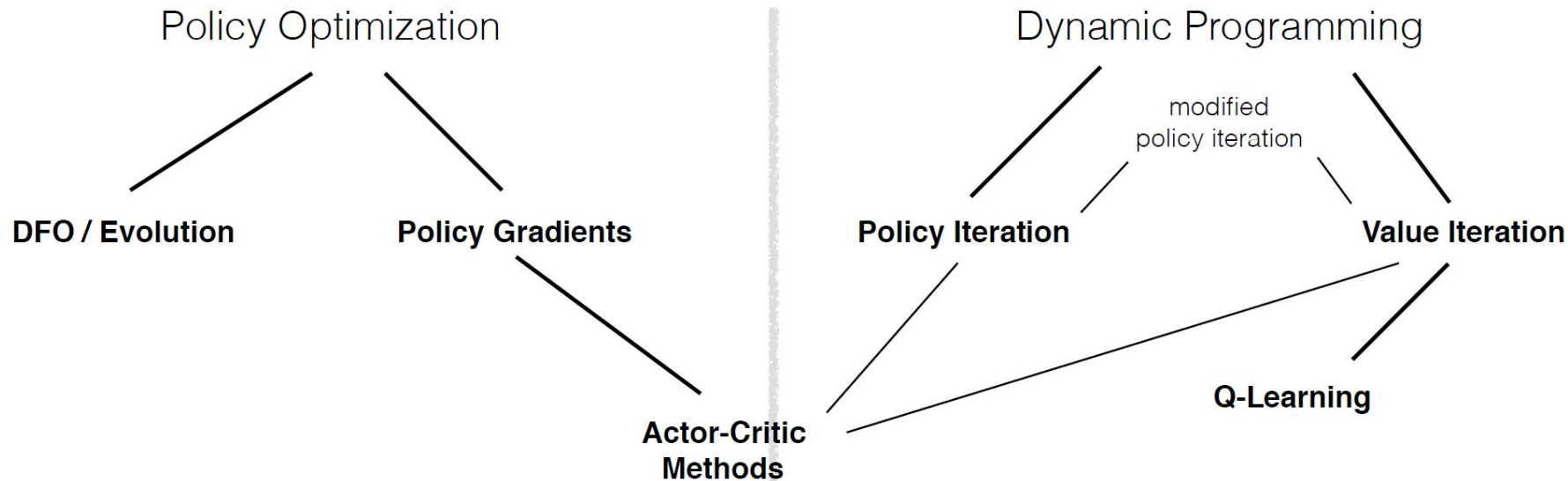
Plan

- Why policy optimisation
- Derivative-Free methods
- Gradient-Based methods
- Likelihood ratio PG

Policy Optimisation in RL Landscape

Direct search in policy space

If we know the policy's corresponding Q-function,
we can improve the policy



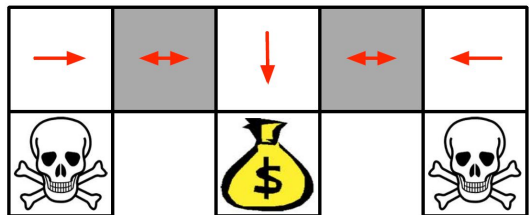
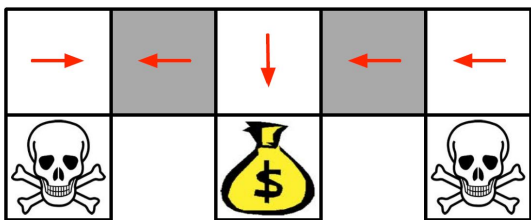
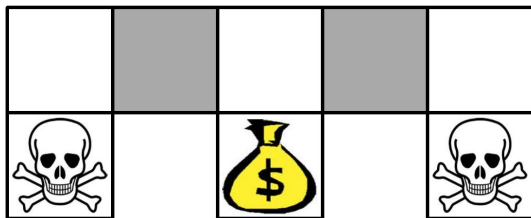
Why Policy Optimisation instead of Value Function

- Might be simpler than V or Q
 - Robotic grasping
- V does not prescribe what action to pick
 - Compute 1 Bellman backup
 - Needs dynamics model
- Q also does not directly prescribe the action
 - Need to compute argmax efficiently



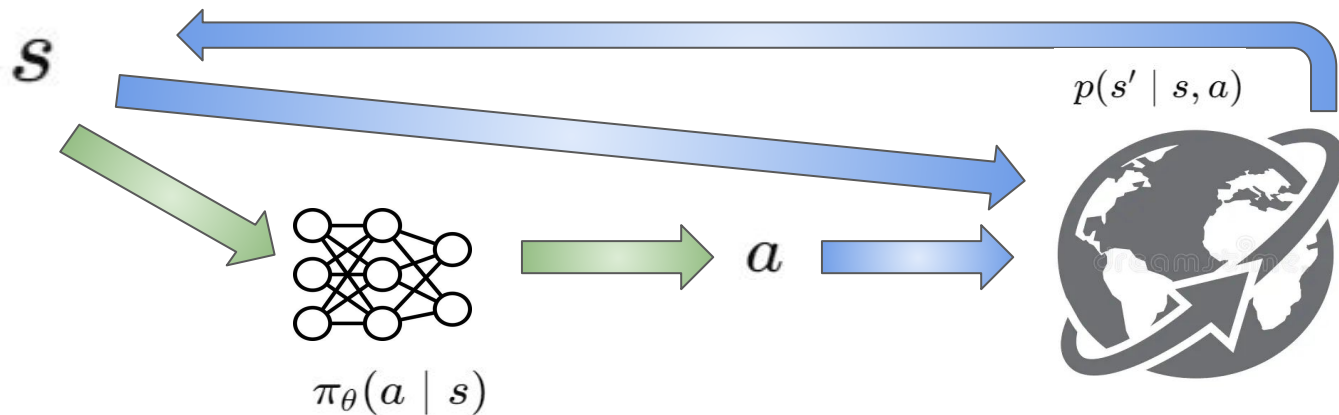
[Source: Deepmind, DQN](#)

PG and stochastic policies



- Representation may not ensure that Markov property is met. No guarantee of an optimal deterministic policy
- Smooths the optimisation landscape
- Ensures exploration

RL Objective



$$\theta^* = \arg \max_{\theta} \mathbb{E}_{\tau \sim p_{\theta}(\tau)} [\sum_t r(s_t, a_t)]$$

Derivative-Free Policy Optimization

Recipe

$$\max_{\theta} U(\theta) = \max_{\theta} \mathbb{E}\left[\sum_{t=0}^H R(s_t) | \pi_{\theta}\right]$$

General Algorithm:

- Make some random change to the parameters
- If the result improves, keep the change
- Repeat

Regulator Problem

State (s): Single-valued real number ($s \in \mathbb{R}$).

Action (a): Single-valued real number ($a \in \mathbb{R}$).

State Transition: $s' \sim \mathcal{N}(s + a, 0.1^2)$

(Gaussian noise with variance 0.01).

Regulator Problem

Reward Function: $R(s, a) = -s^2$

The reward depends only on the current state and penalizes large values of s , encouraging the system to drive s toward zero.

Initial State Distribution: $s_0 \sim \mathcal{N}(0.3, 0.1^2)$

How does the state evolve?

$$a^*(s) = -Ks$$

$$a \sim \mathcal{N}(\theta_1 s, (|\theta_2| + 10^{-5})^2)$$

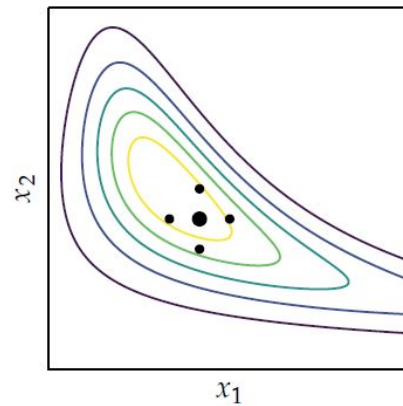
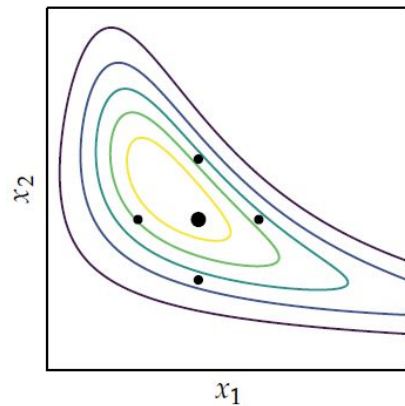
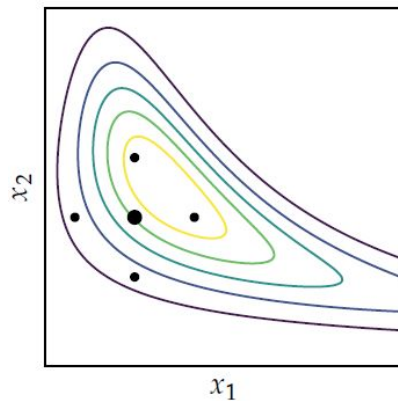
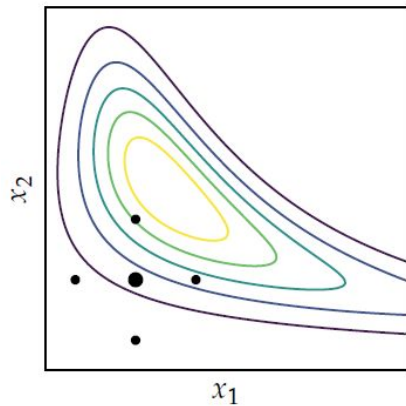
Local Search: Hooke-Jeeves

- Start with initial guess x_0
- Explore at several points using coordinate directions
- Move to the location that gives best improvement
- If improvement not found, reduce step size
- Repeat
- Terminate if step size reduces below threshold

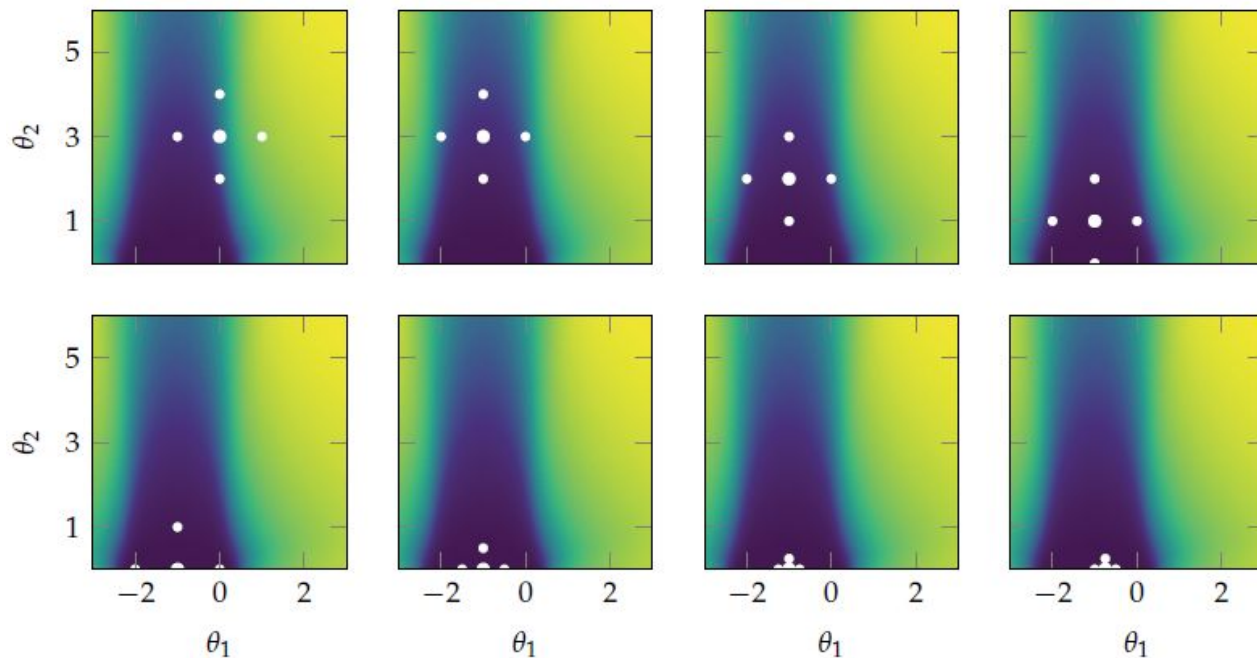
$\pm \alpha$

How many points?

Hooke-Jeeves Method



Hooke Jeeves on the Regulator Problem



Cross Entropy Method

$$H(p, q) = - \sum_x p(x) \log q(x).$$

- *Sample.* We draw n samples from $P(\lambda; \theta)$ and evaluate their performance using Algorithm 4.11. Sort the samples in decreasing order of performance so that $i < j$ implies that $V(\lambda_i) \geq V(\lambda_j)$.
- *Update.* Use the top m performing samples (often called the *elite* samples) to update θ using cross entropy minimization, which boils down to:

$$\theta \leftarrow \arg \max_{\theta} \sum_{j=1}^m \log P(\lambda_j | \theta). \quad (4.40)$$

Cross Entropy Method

CEM:

Initialize $\mu \in \mathbb{R}^d, \sigma \in \mathbb{R}_{>0}^d$

for iteration = 1, 2, ...

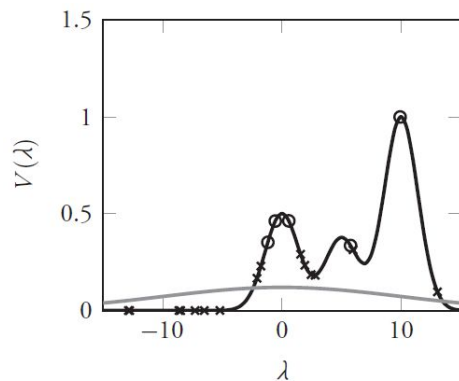
 Sample n parameters $\theta_i \sim N(\mu, \text{diag}(\sigma^2))$

 For each θ_i , perform one rollout to get return $R(\tau_i)$

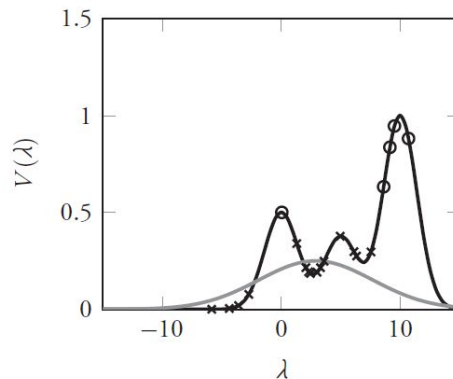
 Select the top k% of θ , and fit a new diagonal Gaussian to those samples. Update μ, σ

endfor

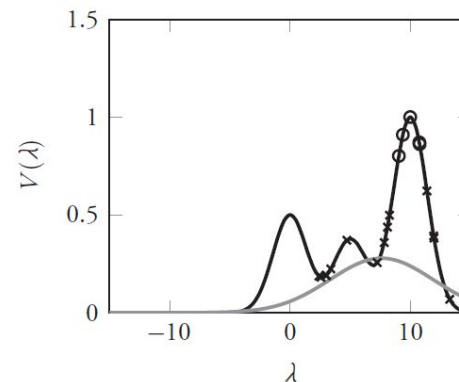
Cross Entropy Method 1D



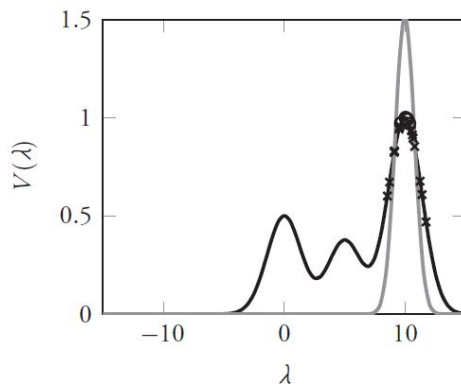
(a) Iteration 1.



(b) Iteration 2.



(c) Iteration 3.



(d) Iteration 4.

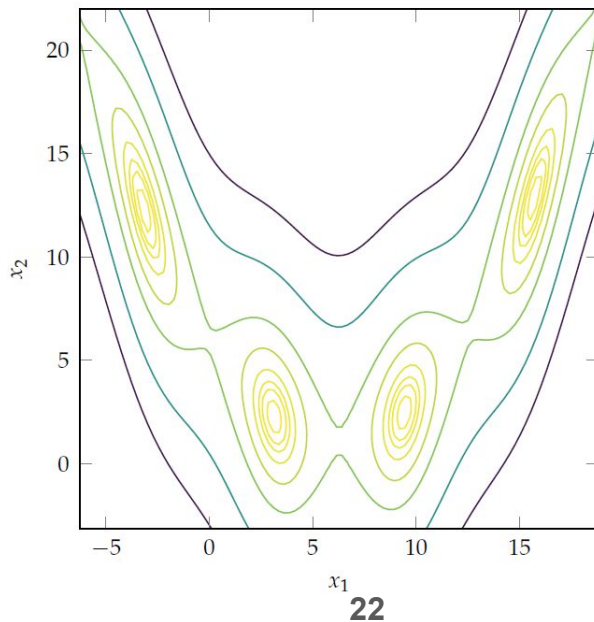
— $V(\lambda)$ — $P(\lambda; \theta)$ ○ Elite samples × Non-elite samples

Cross Entropy Method 2D

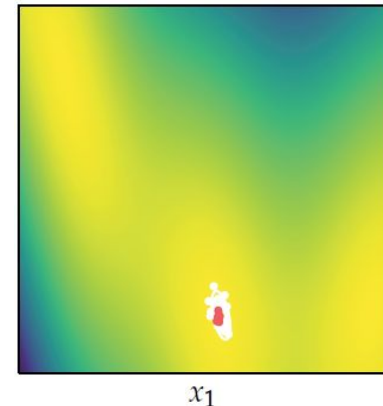
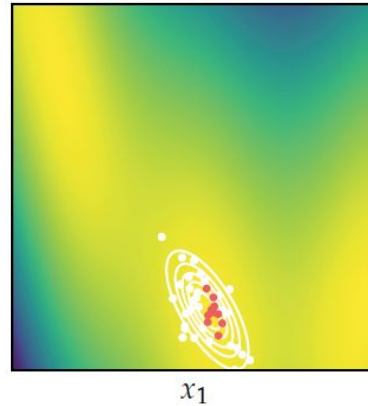
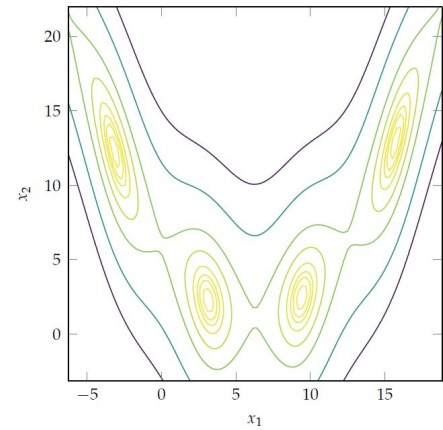
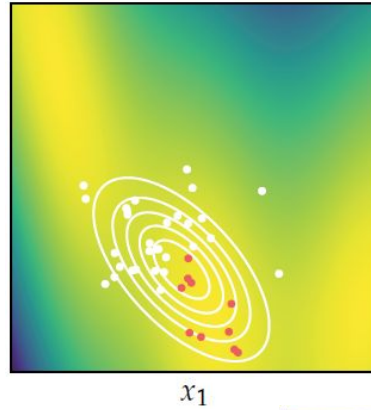
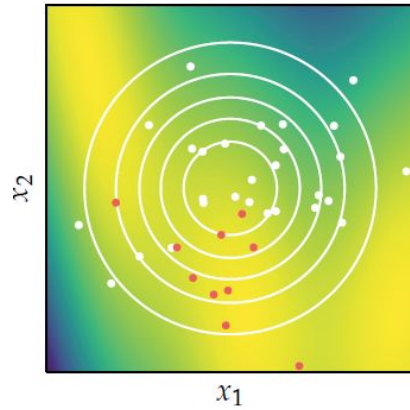
The *Branin function* (figure B.3) is a two-dimensional function,

$$f(\mathbf{x}) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1 - t) \cos(x_1) + s \quad (\text{B.3})$$

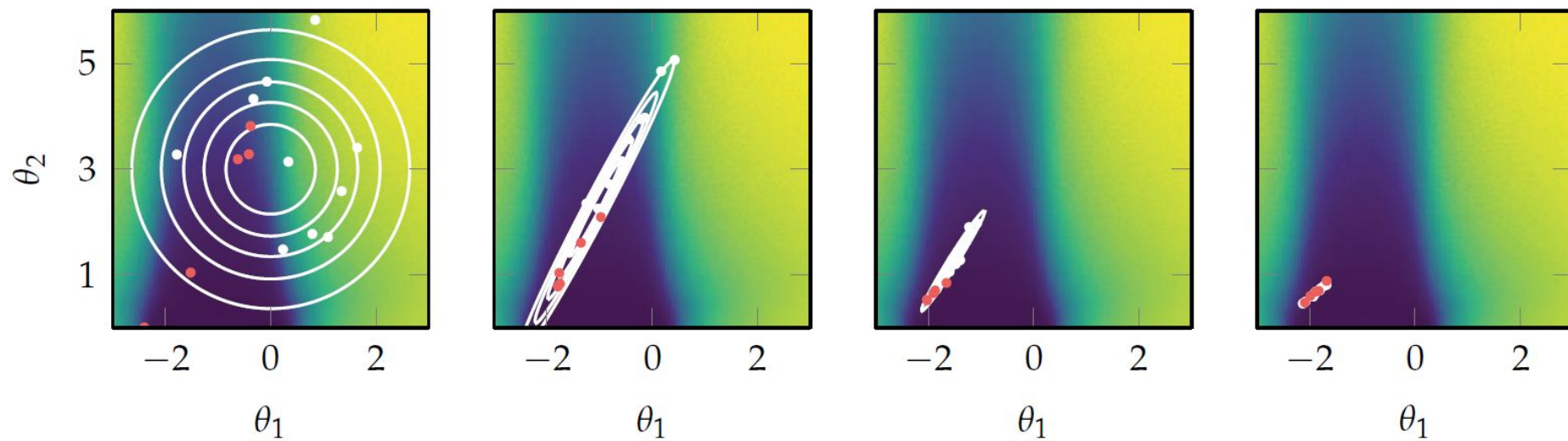
with recommended values $a = 1$, $b = 5.1/(4\pi^2)$, $c = 5/\pi$, $r = 6$, $s = 10$, and $t = 1/(8\pi)$.



Cross Entropy Method



CEM on the Regulator Problem



Tetris

Approximate Dynamic Programming Finally Performs Well in the Game of Tetris

Victor Gabillon
INRIA Lille - Nord Europe,
Team SequeL, FRANCE
victor.gabillon@inria.fr

Mohammad Ghavamzadeh*
INRIA Lille - Team SequeL
& Adobe Research
mohammad.ghavamzadeh@inria.fr

Bruno Scherrer
INRIA Nancy - Grand Est,
Team Maia, FRANCE
bruno.scherrer@inria.fr

- Tsitsiklis and Van Roy (1996):
 - Approximate value iteration 30-40 lines
 - Two features: board height, number of holes in board
- Bertsekas and Ioffe (1996):
 - Lambda policy iteration 4000 lines
 - Elaborate set of 22 features
- Farias and Van Roy (2006):
 - Linear programming
- Kakade (2001):
 - Natural policy gradient 4700, 6800
- Furmston and Barber (2012):
 - Newton method in policy space 14000 lines

CEM-based Approaches for Tetris

- Szita and Lorintz (2006):
 - Bertsekas features 3,50,000 lines
 - Optimise weights using CEM
- Dellacherie (2006):
 - Heuristic evaluation function 50,00,000 lines
 - Eval func made by 6 features and hand designed weights
- Dellacherie (2009):
 - 9 features including Dellacherie 3,50,00,000 lines
 - CEM to optimise the weights

Gradient-Based Policy Optimization

Finite Differences

$$\frac{\partial U}{\partial \theta_j}(\theta) = \frac{U(\theta + \epsilon e_j) - U(\theta - \epsilon e_j)}{2\epsilon}$$

$$e_j = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow j'\text{th entry}$$

Finite Differences

$$\frac{\partial U}{\partial \theta_j}(\theta) = \frac{U(\theta + \epsilon e_j) - U(\theta - \epsilon e_j)}{2\epsilon}$$

$$e_j = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow j\text{'th entry}$$



[Source](#)



[Source](#)

Numerical Differentiation

$$f(x+h) = f(x) + \frac{f'(x)}{1!}h + \frac{f''(x)}{2!}h^2 + \frac{f'''(x)}{3!}h^3 + \dots$$

$$f'(x)h = f(x+h) - f(x) - \frac{f''(x)}{2!}h^2 - \frac{f'''(x)}{3!}h^3 - \dots$$

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{f''(x)}{2!}h - \frac{f'''(x)}{3!}h^2 - \dots$$

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

error term is $O(h)$

Central Difference

$$f(x + h/2) = f(x) + f'(x)\frac{h}{2} + \frac{f''(x)}{2!}\left(\frac{h}{2}\right)^2 + \frac{f'''(x)}{3!}\left(\frac{h}{2}\right)^3 + \dots$$

$$f(x - h/2) = f(x) - f'(x)\frac{h}{2} + \frac{f''(x)}{2!}\left(\frac{h}{2}\right)^2 - \frac{f'''(x)}{3!}\left(\frac{h}{2}\right)^3 + \dots$$

$$f(x + h/2) - f(x - h/2) \approx 2f'(x)\frac{h}{2} + \frac{2}{3!}f'''(x)\left(\frac{h}{2}\right)^3$$

$$f'(x) \approx \frac{f(x + h/2) - f(x - h/2)}{h} - \frac{f'''(x)h^2}{24}$$

error term of $O(h^2)$

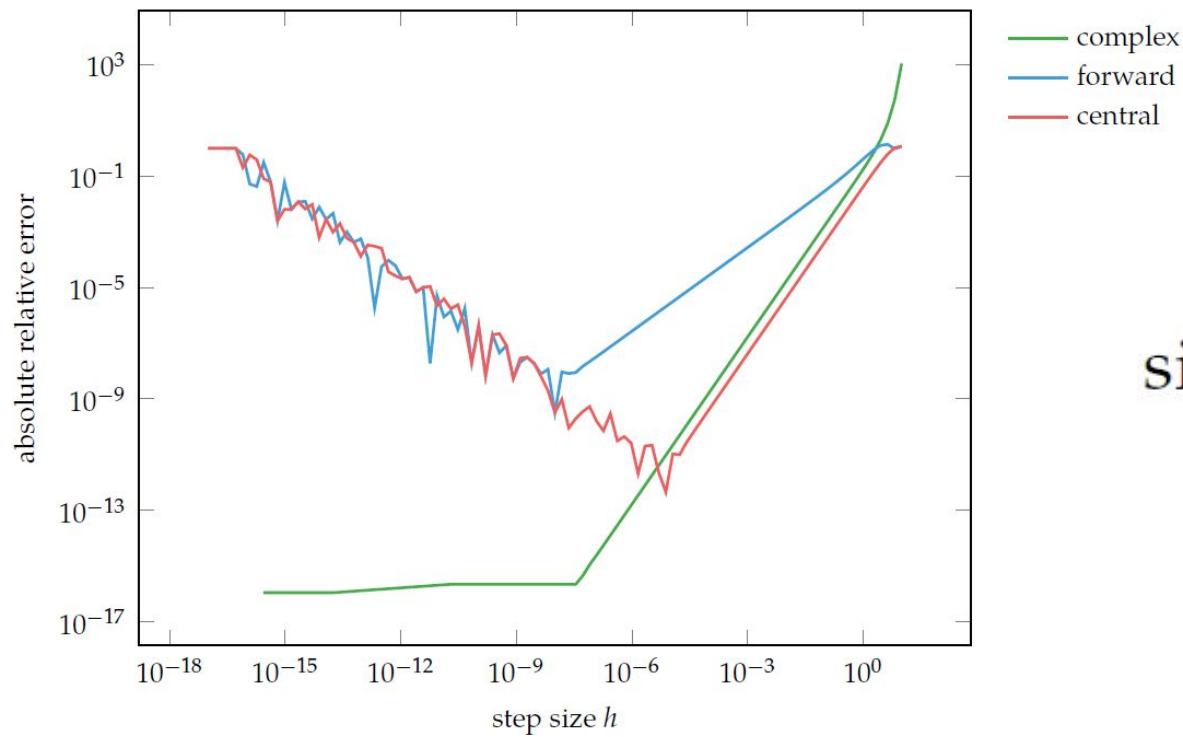
Complex Step Method

$$f(x + ih) = f(x) + ihf'(x) - h^2 \frac{f''(x)}{2!} - ih^3 \frac{f'''(x)}{3!} + \dots$$

$$\operatorname{Im}(f(x + ih)) = hf'(x) - h^3 \frac{f'''(x)}{3!} + \dots$$

$$\begin{aligned}\Rightarrow f'(x) &= \frac{\operatorname{Im}(f(x + ih))}{h} + h^2 \frac{f'''(x)}{3!} - \dots \\ &= \frac{\operatorname{Im}(f(x + ih))}{h} + O(h^2) \text{ as } h \rightarrow 0\end{aligned}$$

Derivative Evaluation Comparison



$\sin(x)$ at $x = 1/2$

Example: Parameterization Issue

Consider a single-state, single-step MDP with a one-dimensional continuous action space and a reward function $R(s, a) = a$. In this case, larger actions produce higher rewards.

Suppose we have a stochastic policy π_{θ} that samples its action according to a uniform distribution between θ_1 and θ_2 for $\theta_2 > \theta_1$. The expected value is

$$U(\theta) = \mathbb{E}[a] = \int_{\theta_1}^{\theta_2} a \frac{1}{\theta_2 - \theta_1} da = \frac{\theta_1 + \theta_2}{2}$$

The policy gradient is

$$\nabla U(\theta) = [1/2, 1/2]$$

The policy could be reparameterized to draw actions from a uniform distribution between θ'_1 and $100\theta'_2$, for $100\theta'_2 > \theta'_1$. Now the expected reward is $(\theta'_1 + 100\theta'_2)/2$ and the policy gradient is $[1/2, 50]$.

The two parameterizations can represent the same policies, but they have very different gradients. Finding a suitable perturbation scalar for the second policy is much more difficult because the parameters vary widely in scale.

Finite Difference Gradient Computation

$$\begin{aligned}\nabla U(\boldsymbol{\theta}) &= \left[\frac{\partial U}{\partial \theta_1}(\boldsymbol{\theta}), \dots, \frac{\partial U}{\partial \theta_n}(\boldsymbol{\theta}) \right] \\ &\approx \left[\frac{U(\boldsymbol{\theta} + \delta \mathbf{e}^{(1)}) - U(\boldsymbol{\theta})}{\delta}, \dots, \frac{U(\boldsymbol{\theta} + \delta \mathbf{e}^{(n)}) - U(\boldsymbol{\theta})}{\delta} \right]\end{aligned}$$

Need to simulate policy rollouts to estimate objective value. High variance of rollout rewards

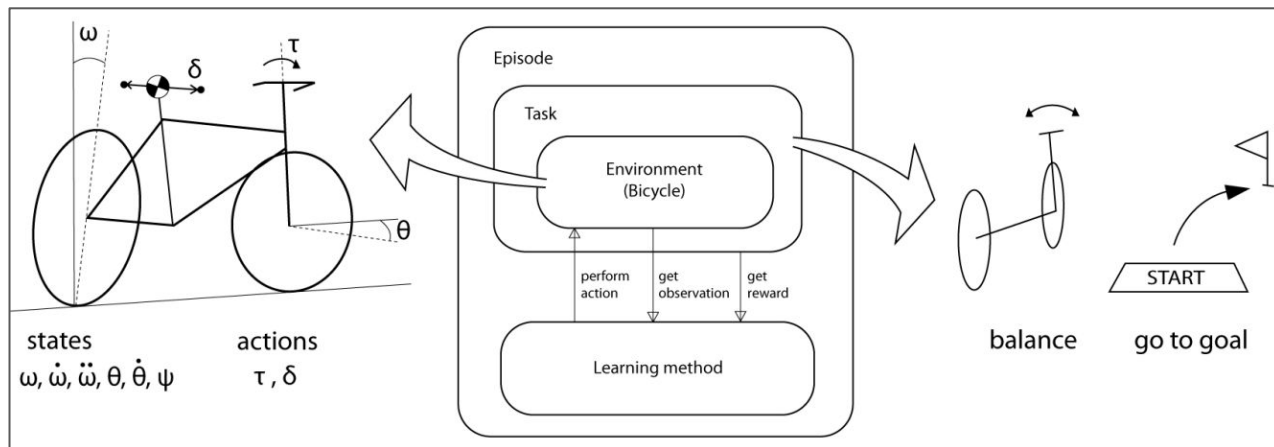
High Variance Issue

PEGASUS: A policy search method for large MDPs and POMDPs

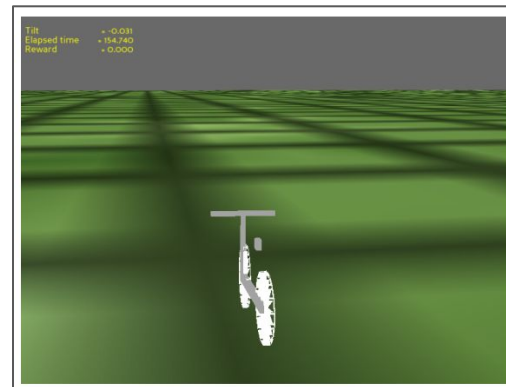
Andrew Y. Ng
Computer Science Division
UC Berkeley
Berkeley, CA 94720

Michael Jordan
Computer Science Division
& Department of Statistics
UC Berkeley
Berkeley, CA 94720

UAI 2000



Bicycle task from Randlov & Alstrom, ICML 1998



Cam et. al., CS 229, Stanford 2013

Likelihood Ratio Policy Gradient

Finding the Objective Value

$$\theta^* = \arg \max_{\theta} \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[\sum_{t=1}^T r(s_t, a_t) \right]$$

$$\text{Let } r(\tau) = \sum_{t=1}^T r(s_t, a_t)$$

$$\text{Thus, } J(\theta) = \mathbb{E}_{\tau \sim p_{\theta}(\tau)} [r(\tau)]$$

$$\approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T r(s_{i,t}, a_{i,t})$$

Take samples and estimate the expectation

Rollouts generated by running the policy

Optimising the Objective Value

$$J(\theta) = \mathbb{E}_{\tau \sim p_{\theta}(\tau)}[r(\tau)]$$

$$\text{i.e., } J(\theta) = \int p_{\theta}(\tau) r(\tau) d\tau$$

$$\nabla_{\theta} J(\theta) = \int \nabla_{\theta} p_{\theta}(\tau) r(\tau) d\tau$$

$$p_{\theta}(\tau) = p_{\theta}(s_1, a_1, \dots, s_T, a_T)$$

$$p_{\theta}(s_1, a_1, \dots, s_T, a_T) = p(s_1) \prod_{t=1}^T \pi_{\theta}(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t)$$

To compute the gradient, we need to know the initial state distribution and transition probability distribution

Gradient Expression

$$\nabla_{\theta} J(\theta) = \int \nabla_{\theta} p_{\theta}(\tau) r(\tau) d\tau$$
$$r(\tau) = \sum_{t=1}^T r(s_t, a_t)$$

Convert the integral to an expected value, and estimate it using samples

Goal: Isolate $p_{\theta}(\tau)$

$$\nabla_{\theta} p_{\theta}(\tau) = p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau)$$

$$\begin{aligned} \nabla_{\theta} J(\theta) &= \int p_{\theta}(\tau) \nabla_{\theta} \log p_{\theta}(\tau) r(\tau) d\tau \\ &= \mathbb{E}_{\tau \sim p_{\theta}(\tau)} [\nabla_{\theta} \log p_{\theta}(\tau) r(\tau)] \end{aligned}$$

What's the chain of thought?

$$\begin{aligned} J(\theta) &= \mathbb{E}_{\tau \sim p_{\theta}(\tau)} [r(\tau)] \\ &= \int p_{\theta}(\tau) r(\tau) d\tau \\ \nabla_{\theta} J(\theta) &= \int \nabla_{\theta} p_{\theta}(\tau) r(\tau) d\tau \\ &= \mathbb{E}_{\tau \sim p_{\theta}(\tau)} [\nabla_{\theta} \log p_{\theta}(\tau) r(\tau)] \end{aligned}$$

Gradient Estimator

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim p_{\theta}(\tau)} [\nabla_{\theta} \log p_{\theta}(\tau) r(\tau)]$$

$$p_{\theta}(\tau) = p_{\theta}(s_1, a_1, \dots, s_T, a_T)$$

Recall

$$p_{\theta}(s_1, a_1, \dots, s_T, a_T) = p(s_1) \prod_{t=1}^T \pi_{\theta}(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t)$$

$$\log p_{\theta}(\tau) = \log p(s_1) + \sum_{t=1}^T \log \pi_{\theta}(a_t \mid s_t) + \log p(s_{t+1} \mid s_t, a_t)$$

$$\nabla_{\theta} \log p_{\theta}(\tau) = \nabla_{\theta} \sum_{t=1}^T \log \pi_{\theta}(a_t \mid s_t)$$

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[\left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t \mid s_t) \right) \left(\sum_{t=1}^T r(s_t, a_t) \right) \right]$$

Gradient Computation

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[\left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t \mid s_t) \right) \left(\sum_{t=1}^T r(s_t, a_t) \right) \right]$$

$$J(\theta) = \mathbb{E}_{\tau \sim p_{\theta}(\tau)} \left[r(\tau) \right]$$

Estimating expectation

$$\approx \frac{1}{N} \sum_{i=1}^N \sum_{t=1}^T r(s_{i,t}, a_{i,t})$$

$$\nabla_{\theta} J(\theta) \approx \frac{1}{N} \sum_{i=1}^N \left(\sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_{i,t} \mid s_{i,t}) \right) \left(\sum_{t=1}^T r(s_{i,t}, a_{i,t}) \right)$$

No need to know the initial state distribution or transition dynamics

$$p_{\theta}(s_1, a_1, \dots, s_T, a_T) = p(s_1) \prod_{t=1}^T \pi_{\theta}(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t)$$