Robotics

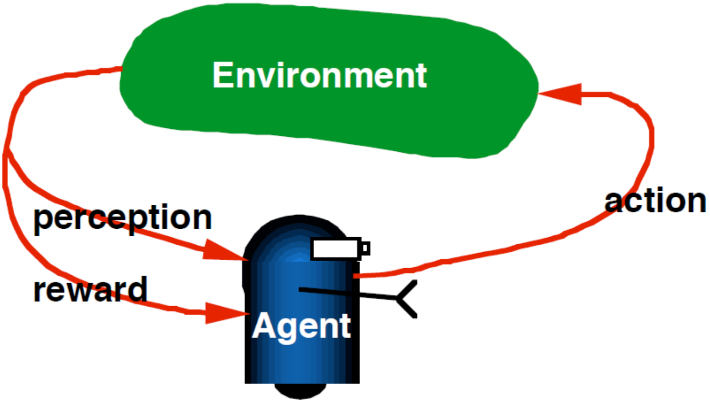
Reinforcement Learning in Robotics –
briefly

NO KEYWORDS

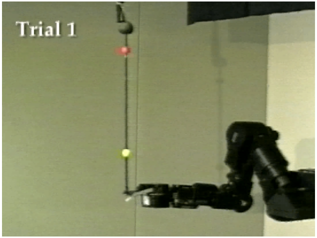
Marc Toussaint
University of Stuttgart
Winter 2016/17

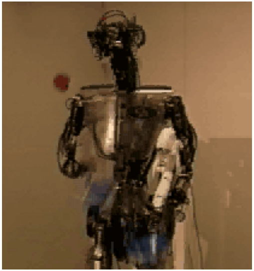
Lecturer: Vien Ngo

RL = Learning to Act



from Satinder Singh's *Introduction to RL*, videolectures.com





(around 2000, by Schaal, Atkeson, Vijayakumar)

(2007, Andrew Ng et al.)

Applications of RL

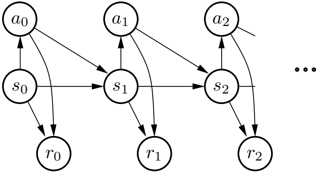
- Robotics
  - Navigation, walking, juggling, helicopters, grasping, etc...
- Games
  - Backgammon, Chess, Othello, Tetris, Atari games(2014), AlphaGO (2016)...
- Control
  - factory processes, resource control in multimedia networks, elevators, ...
- Operations Research
  - Warehousing, transportation, scheduling, online advertisements, ...

Markov Decision Process

Markov world: $P(s_t|a_{t-1}, s_{t-1}, a_{t-2}, s_{t-2}, \ldots, s_0) = P(s_t|a_{t-1}, s_{t-1})$

Markov Decision Process

Markov world: $P(s_t|a_{t-1}, s_{t-1}, a_{t-2}, s_{t-2}, \dots, s_0) = P(s_t|a_{t-1}, s_{t-1})$



$$P(s_{0:T+1}, a_{0:T}, r_{0:T}; \pi) = P(s_0) \ \prod_{t=0}^T P(a_t|s_t; \pi) \ P(r_t|s_t, a_t) \ P(s_{t+1}|s_t, a_t)$$

– world's initial state distribution $P(s_0)$

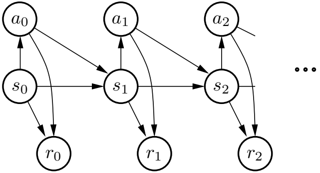
– world's transition probabilities $P(s_{t+1} \mid s_t, a_t)$

– world's reward probabilities $P(r_t \mid s_t, a_t)$

– agent's *policy* $\pi(a_t \mid s_t) = P(a_0|s_0; \pi)$ (or deterministic $a_t = \pi(s_t)$)

Markov Decision Process

Markov world: $P(s_t|a_{t-1}, s_{t-1}, a_{t-2}, s_{t-2}, \dots, s_0) = P(s_t|a_{t-1}, s_{t-1})$



$$P(s_{0:T+1}, a_{0:T}, r_{0:T}; \pi) = P(s_0) \ \prod_{t=0}^T P(a_t|s_t; \pi) \ P(r_t|s_t, a_t) \ P(s_{t+1}|s_t, a_t)$$

– world's initial state distribution $P(s_0)$
– world's transition probabilities $P(s_{t+1} \mid s_t, a_t)$
– world's reward probabilities $P(r_t \mid s_t, a_t)$
– agent's *policy* $\pi(a_t \mid s_t) = P(a_0|s_0; \pi)$ (or deterministic $a_t = \pi(s_t)$)

- **Stationary MDP:**
  – We assume $P(s' \mid s, a)$ and $P(r|s, a)$ independent of time
  – We also define $R(s, a) := \mathsf{E}\{r|s, a\} = \int r \ P(r|s, a) \ dr$

... in the notation of control theory

- We have a (potentially stochastic) controlled system

$$\dot{x} = f(x, u) + \text{noise}(x, u)$$

- We have costs (neg-rewards), e.g. in the finite horizon case:

$$J^\pi = \int_0^T c(x(t), u(t))\; dt + \phi(x(T))$$

- We want a policy ("controller")

$$\pi : (x, t) \mapsto u$$

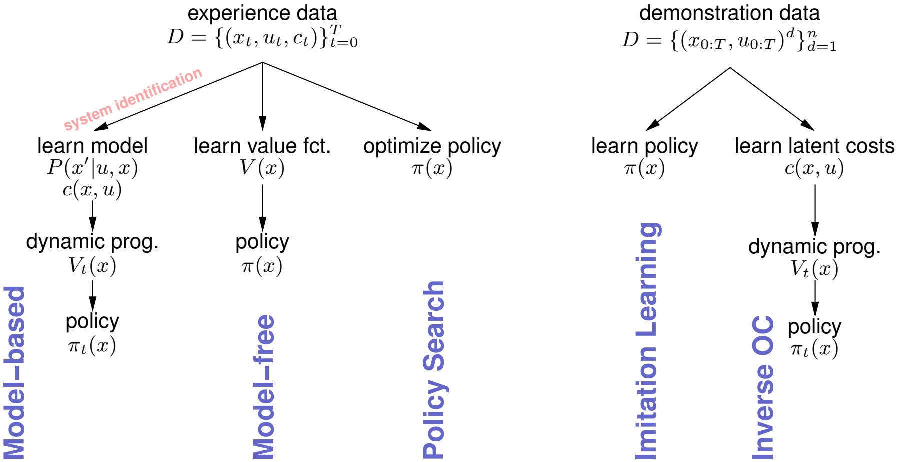
Reinforcement Learning = the dynamics f and costs c are unknown

- All the agent can do is collect data

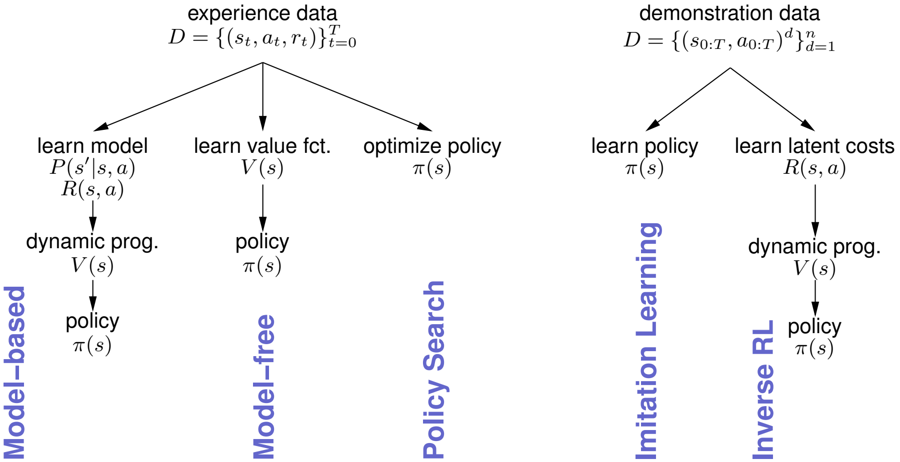
$$D = \{(x_t, u_t, c_t)\}_{t=0}^{T}$$

What can we do with this data?

Five approaches to RL



Five approaches to RL



Imitation Learning

$$\boxed{D = \{(s_{0:T}, a_{0:T})^d\}_{d=1}^n \quad \overset{\text{learn/copy}}{\rightarrow} \quad \pi(s)}$$

- Use ML to imitate demonstrated state trajectories $x_{0:T}$

  Literature:

  Atkeson & Schaal: Robot learning from demonstration (ICML 1997)

  Schaal, Ijspeert & Billard: Computational approaches to motor learning by imitation (Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences 2003)

  Grimes, Chalodhorn & Rao: Dynamic Imitation in a Humanoid Robot through Nonparametric Probabilistic Inference. (RSS 2006)

  Rüdiger Dillmann: Teaching and learning of robot tasks via observation of human performance (Robotics and Autonomous Systems, 2004)

Imitation Learning

- There a many ways to imitate/copy the oberved policy:

  Learn a density model $P(a_t \mid s_t); P(s_t)$ (e.g., with mixture of Gaussians) from the observed data and use it as policy (Billard et al.)

  Or trace observed trajectories by minimizing perturbation costs (Atkeson & Schaal 1997)

Imitation Learning

Atkeson & Schaal

Inverse RL

$$\boxed{D = \{(s_{0:T}, a_{0:T})^d\}_{d=1}^n \quad \overset{\text{learn}}{\rightarrow} \quad R(s,a) \quad \overset{\text{DP}}{\rightarrow} \quad V(s) \quad \rightarrow \quad \pi(s)}$$

- Use ML to “uncover” the latent reward function in observed behavior

Literature:

Pieter Abbeel & Andrew Ng: Apprenticeship learning via inverse reinforcement learning (ICML 2004)

Andrew Ng & Stuart Russell: Algorithms for Inverse Reinforcement Learning (ICML 2000)

Nikolay Jetchev & Marc Toussaint: Task Space Retrieval Using Inverse Feedback Control (ICML 2011).

Inverse RL (Apprenticeship Learning)

- Given: demonstrations $D = \{x^d_{0:T}\}^n_{d=1}$
- Try to find a reward function that **discriminates demonstrations from other policies**

Inverse RL (Apprenticeship Learning)

- Given: demonstrations $D = \{x^d_{0:T}\}^n_{d=1}$
- Try to find a reward function that **discriminates demonstrations from other policies**
  – Assume the reward function is linear in some features $R(x) = w^\top \phi(x)$

Inverse RL (Apprenticeship Learning)

- Given: demonstrations $D = \{x^d_{0:T}\}^n_{d=1}$
- Try to find a reward function that **discriminates demonstrations from other policies**
  - Assume the reward function is linear in some features $R(x) = w^\top \phi(x)$
  - Iterate:
    1. Given a set of candidate policies $\{\pi_0, \pi_1, ..\}$

Inverse RL (Apprenticeship Learning)

- Given: demonstrations $D = \{x^d_{0:T}\}_{d=1}^n$
- Try to find a reward function that **discriminates demonstrations from other policies**
  – Assume the reward function is linear in some features $R(x) = w^\top \phi(x)$
  – Iterate:
    1. Given a set of candidate policies $\{\pi_0, \pi_1, ..\}$
    2. Find weights w that maximize the value margin between teacher and all other candidates

Inverse RL (Apprenticeship Learning)

- Given: demonstrations $D = \{x_{0:T}^d\}_{d=1}^n$
- Try to find a reward function that **discriminates demonstrations from other policies**
  – Assume the reward function is linear in some features $R(x) = w^\top \phi(x)$
  – Iterate:
    1. Given a set of candidate policies $\{\pi_0, \pi_1, ..\}$
    2. Find weights w that maximize the value margin between teacher and all other candidates

$$\max_{w,\xi} \quad \xi$$
$$\text{s.t.} \quad \forall_{\pi_i}: \quad \underbrace{w^\top \langle\phi\rangle_D}_{\text{value of demonstrations}} \geq \underbrace{w^\top \langle\phi\rangle_{\pi_i}}_{\text{value of } \pi_i} + \xi$$
$$\|w\|^2 \leq 1$$

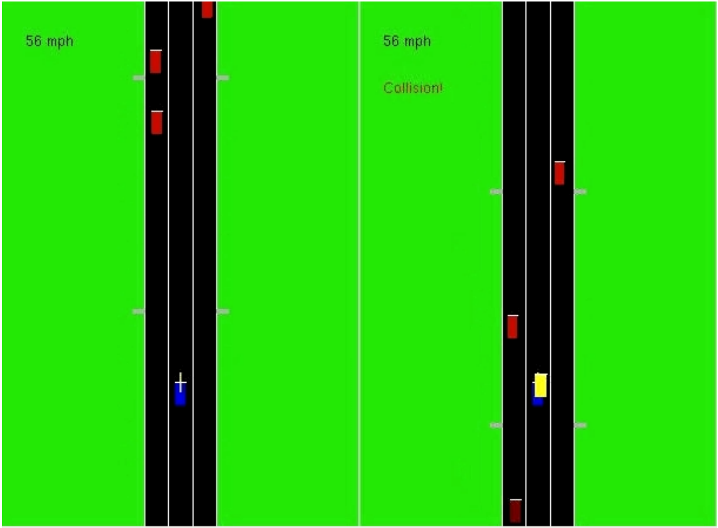
Inverse RL (Apprenticeship Learning)

- Given: demonstrations $D = \{x^d_{0:T}\}_{d=1}^n$
- Try to find a reward function that **discriminates demonstrations from other policies**
  - Assume the reward function is linear in some features $R(x) = w^\top \phi(x)$
  - Iterate:
    1. Given a set of candidate policies $\{\pi_0, \pi_1, ..\}$
    2. Find weights w that maximize the value margin between teacher and all other candidates

$$\begin{aligned}
&\max_{w,\xi} \; \xi \\
&\text{s.t. } \forall_{\pi_i} : \quad \underbrace{w^\top \langle\phi\rangle_D}_{\text{value of demonstrations}} \geq \underbrace{w^\top \langle\phi\rangle_{\pi_i}}_{\text{value of } \pi_i} + \xi \\
&\quad \|w\|^2 \leq 1
\end{aligned}$$

    3. Compute a new candidate policy π_i that optimizes $R(x) = w^\top \phi(x)$ and add to candidate list.

(Abbeel & Ng, ICML 2004)


56 mph
56 mph
Collision!

Policy Search with Policy Gradients

Policy gradients

- In continuous state/action case, represent the policy as linear in arbitrary state features:

$$\pi(s) = \sum_{j=1}^{k} \phi_j(s)\beta_j = \phi(s)^\top \beta \qquad \text{(deterministic)}$$

$$\pi(a \mid s) = \mathcal{N}(a \mid \phi(s)^\top \beta, \phi(s)^\top \Sigma \phi(s)) \qquad \text{(stochastic)}$$

with k features ϕ_j.

- Given an episode $\xi = (s_t, a_t, r_t)_{t=0}^{H}$, we want to estimate

$$\frac{\partial V(\beta)}{\partial \beta}$$

Policy Search

- REINFORCE:

$$\frac{\partial V(\beta)}{\partial \beta} = \frac{\partial}{\partial \beta} \int P(\xi|\beta)\ R(\xi)\ d\xi = \int P(\xi|\beta) \frac{\partial}{\partial \beta} \log P(\xi|\beta) R(\xi) d\xi$$

$$= \mathsf{E}_{\xi|\beta}\{ \frac{\partial}{\partial \beta} \log P(\xi|\beta) R(\xi) \} = \mathsf{E}_{\xi|\beta}\{ \sum_{t=0}^{H} \gamma^t \frac{\partial \log \pi(a_t|s_t)}{\partial \beta} \underbrace{\sum_{t'=t}^{H} \gamma^{t'-t} r_{t'}}_{Q^\pi(s_t,a_t,t)} \}$$

Policy Search

- REINFORCE:

$$\frac{\partial V(\beta)}{\partial \beta} = \frac{\partial}{\partial \beta} \int P(\xi|\beta)\ R(\xi)\ d\xi = \int P(\xi|\beta) \frac{\partial}{\partial \beta} \log P(\xi|\beta) R(\xi) d\xi$$

$$= \mathsf{E}_{\xi|\beta}\{\frac{\partial}{\partial \beta} \log P(\xi|\beta) R(\xi)\} = \mathsf{E}_{\xi|\beta}\{\sum_{t=0}^{H} \gamma^t \frac{\partial \log \pi(a_t|s_t)}{\partial \beta} \underbrace{\sum_{t'=t}^{H} \gamma^{t'-t} r_{t'}}_{Q^\pi(s_t,a_t,t)}\}$$

- Natural Policy Gradient
  – Estimate the Q-function as linear in the basis functions $\frac{\partial}{\partial \beta} \log \pi(a|s)$:

$$Q(s,a) \approx \Big[\frac{\partial \log \pi(a|s)}{\partial \beta}\Big]^\top w$$

  – Then the natural gradient ($\frac{\partial V(\beta)}{\partial \beta}$ multiplied with inv. Fisher metric) updates

$$\beta^{\text{new}} = \beta + \alpha w$$

Policy Search

- REINFORCE:

$$\frac{\partial V(\beta)}{\partial \beta} = \frac{\partial}{\partial \beta} \int P(\xi|\beta)\ R(\xi)\ d\xi = \int P(\xi|\beta) \frac{\partial}{\partial \beta} \log P(\xi|\beta) R(\xi) d\xi$$

$$= \mathsf{E}_{\xi|\beta}\{\frac{\partial}{\partial \beta} \log P(\xi|\beta) R(\xi)\} = \mathsf{E}_{\xi|\beta}\{\sum_{t=0}^{H} \gamma^t \frac{\partial \log \pi(a_t|s_t)}{\partial \beta} \underbrace{\sum_{t'=t}^{H} \gamma^{t'-t} r_{t'}}_{Q^\pi(s_t,a_t,t)}\}$$

- Natural Policy Gradient
  - Estimate the Q-function as linear in the basis functions $\frac{\partial}{\partial \beta} \log \pi(a|s)$:

$$Q(s,a) \approx \Big[\frac{\partial \log \pi(a|s)}{\partial \beta}\Big]^\top w$$

  - Then the natural gradient ($\frac{\partial V(\beta)}{\partial \beta}$ multiplied with inv. Fisher metric) updates

$$\beta^{\textsf{new}} = \beta + \alpha w$$

- PoWER: Use the stochastic policy $\pi(a \,|\, s)$, let $a_t = \phi(s_t)^\top(\beta + \epsilon_t)$ where $\epsilon_t \sim \mathcal{N}(0, \Sigma)$ is the sampled noise, with updates

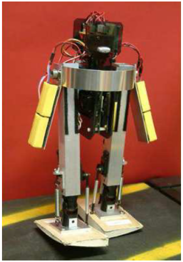
$$\beta \leftarrow \beta + \frac{\mathsf{E}_{\xi|\beta}\{\sum_{t=0}^{H} \epsilon_t Q^\pi(s_t, a_t, t)\}}{\mathsf{E}_{\xi|\beta}\{\sum_{t=0}^{H} Q^\pi(s_t, a_t, t)\}}$$

Kober & Peters: *Policy Search for Motor Primitives in Robotics*, NIPS 2008.

(serious reward shaping!)

Learning to walk in 20 Minutes

- Policy Gradient method (Reinforcement Learning)
  Stationary policy parameterized as linear in features $u = \sum_i w_i \phi_i(q, \dot{q})$
- Problem: find parameters w_i to minimize expected costs
  cost = mimick $(q, \dot{q})$ of the passive down-hill walker at "certain point in cycle"



Learning To Walk

Tedrake, Zhang & Seung: *Stochastic policy gradient reinforcement learning on a simple 3D biped.* IROS, 2849-2854, 2004.
`http://groups.csail.mit.edu/robotics-center/public_papers/Tedrake04a.pdf`

Policy Search – references

Peters & Schaal (2008): *Reinforcement learning of motor skills with policy gradients*, Neural Networks.

Kober & Peters: *Policy Search for Motor Primitives in Robotics*, NIPS 2008.

Vlassis, Toussaint (2009): *Learning Model-free Robot Control by a Monte Carlo EM Algorithm.* Autonomous Robots 27, 123-130.

Rawlik, Toussaint, Vijayakumar(2012): *On Stochastic Optimal Control and Reinforcement Learning by Approximate Inference.* RSS 2012. *(ψ-learning)*

- These methods are sometimes called **white-box optimization**: They optimize the policy parameters β for the total reward $R = \sum \gamma^t r_t$ while tying to exploit knowledge of how the process is actually parameterized

Policy Search with Black-Box Optimization [skipped]

“Black-Box Optimization”

- The term is not really well defined
  – I use it to express that *only* $f(x)$ can be evaluated
  – $\nabla f(x)$ or $\nabla^2 f(x)$ are not (directly) accessible
  More common terms:

- **Global optimization**
  - This usually emphasizes that methods should not get stuck in local optima
  - Very very interesting domain – close analogies to (active) Machine Learning, bandits, POMDPs, optimal decision making/planning, optimal experimental design
  - Usually mathematically well founded methods

- **Stochastic search** or **Evolutionary Algorithms** or **Local Search**
  - Usually these are local methods (extensions trying to be “more” global)
  - Various interesting heuristics
  - Some of them (implicitly or explicitly) locally approximating gradients or 2nd order models

Black-Box Optimization

- Problem: Let $x \in \mathbb{R}^n$, $f:\ \mathbb{R}^n \to \mathbb{R}$, find

$$\min_x \ f(x)$$

where we can only evaluate $f(x)$ for any $x \in \mathbb{R}^n$

- A constrained version: Let $x \in \mathbb{R}^n$, $f:\ \mathbb{R}^n \to \mathbb{R}$, $g:\ \mathbb{R}^n \to \{0,1\}$, find

$$\min_x \ f(x) \quad \text{s.t.} \quad g(x) = 1$$

where we can only evaluate $f(x)$ and $g(x)$ for any $x \in \mathbb{R}^n$

A zoo of approaches

- People with many different backgrounds drawn into this
  Ranging from heuristics and Evolutionary Algorithms to heavy mathematics

  - Evolutionary Algorithms, esp. Evolution Strategies, Covariance Matrix Adaptation, Estimation of Distribution Algorithms
  - Simulated Annealing, Hill Climing, Downhill Simplex
  - local modelling (gradient/Hessian), global modelling
  - Bayesian optimization

Optimizing and Learning

- Black-Box optimization is strongly related to learning:
- When we have local a gradient or Hessian, we can take that local information and run – no need to keep track of the history or learn (exception: BFGS)
- In the black-box case we have no local information directly accessible $\rightarrow$ one needs to account for the history in some way or another to have an idea where to continue search
- “Accounting for the history” very often means learning: Learning a local or global model of f itself, learning which steps have been successful recently (gradient estimation), or which step directions, or other heuristics

Stochastic Search

Stochastic Search

- The general recipe:
  - The algorithm maintains a probability distribution $p_\theta(x)$
  - In each iteration it takes n samples $\{x_i\}_{i=1}^n \sim p_\theta(x)$
  - Each x_i is evaluated $\rightarrow$ data $\{(x_i, f(x_i))\}_{i=1}^n$
  - That data is used to update θ

Stochastic Search

- The general recipe:
  - The algorithm maintains a probability distribution $p_\theta(x)$
  - In each iteration it takes n samples $\{x_i\}_{i=1}^n \sim p_\theta(x)$
  - Each x_i is evaluated $\rightarrow$ data $\{(x_i, f(x_i))\}_{i=1}^n$
  - That data is used to update θ

- Stochastic Search:

Input: initial parameter θ, function $f(x)$, distribution model $p_\theta(x)$, update heuristic $h(\theta, D)$
Output: final θ and best point x

1: **repeat**
2: Sample $\{x_i\}_{i=1}^n \sim p_\theta(x)$
3: Evaluate samples, $D = \{(x_i, f(x_i))\}_{i=1}^n$
4: Update $\theta \leftarrow h(\theta, D)$
5: **until** θ converges

Stochastic Search

- The parameter θ is the only "knowledge/information" that is being propagated between iterations
  θ encodes what has been learned from the history
  θ defines where to search in the future

- Evolutionary Algorithms: θ is a parent population
  Evolution Strategies: θ defines a Gaussian with mean & variance
  Estimation of Distribution Algorithms: θ are parameters of some distribution model, e.g. Bayesian Network
  Simulated Annealing: θ is the "current point" and a temperature

Example: Gaussian search distribution (μ, λ)-ES

From 1960s/70s. Rechenberg/Schwefel

- Perhaps the simplest type of distribution model

$$\theta = (\hat{x}) \;, \quad p_t(x) = \mathcal{N}(x|\hat{x}, \sigma^2)$$

a n-dimenstional isotropic Gaussian with fixed deviation σ

- Update heuristic:
  – Given $D = \{(x_i, f(x_i))\}_{i=1}^{\lambda}$, select μ best: $D' = \text{bestOf}_\mu(D)$
  – Compute the new mean $\hat{x}$ from D'

- This algorithm is called “Evolution Strategy (μ, λ)-ES”
  – The Gaussian is meant to represent a “species”
  – λ offspring are generated
  – the best μ selected

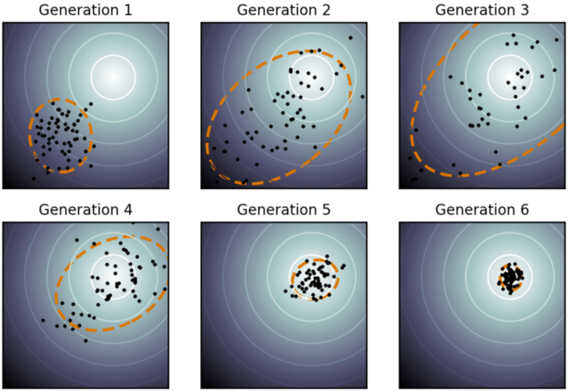
Covariance Matrix Adaptation (CMA-ES)

- An obvious critique of the simple Evolution Strategies:
  - The search distribution $\mathcal{N}(x|\hat{x}, \sigma^2)$ is isotropic
    (no going *forward*, no preferred direction)
  - The variance σ is fixed!

Covariance Matrix Adaptation (CMA-ES)

- An obvious critique of the simple Evolution Strategies:
  - The search distribution $\mathcal{N}(x|\hat{x}, \sigma^2)$ is isotropic (no going *forward*, no preferred direction)
  - The variance σ is fixed!

- Covariance Matrix Adaptation Evolution Strategy (CMA-ES)



Covariance Matrix Adaptation (CMA-ES)

- In Covariance Matrix Adaptation

$$\theta = (\hat{x}, \sigma, C, p_\sigma, p_C) , \quad p_\theta(x) = \mathcal{N}(x|\hat{x}, \sigma^2 C)$$

  where C is the covariance matrix of the search distribution

- The θ maintains two more pieces of information: p_σ and p_C capture the "path" (motion) of the mean $\hat{x}$ in recent iterations
- Rough outline of the θ-update:
  – Let $D' = \text{bestOf}_\mu(D)$ be the set of selected points
  – Compute the new mean $\hat{x}$ from D'
  – Update p_σ and p_C proportional to $\hat{x}_{k+1} - \hat{x}_k$
  – Update σ depending on $|p_\sigma|$
  – Update C depending on $p_c p_c^\top$ (rank-1-update) and $\text{Var}(D')$

CMA references

Hansen, N. (2006), ”The CMA evolution strategy: a comparing review”
Hansen et al.: Evaluating the CMA Evolution Strategy on Multimodal Test Functions, PPSN 2004.

Function	f_{stop}	init	n	CMA-ES	DE	RES	LOS
$f_{\text{Ackley}}(x)$	1e-3	$[-30, 30]^n$	20	**2667**	.	.	6.0e4
			30	**3701**	12481	1.1e5	9.3e4
			100	**11900**	36801	.	.
$f_{\text{Griewank}}(x)$	1e-3	$[-600, 600]^n$	20	**3111**	8691	.	.
			30	**4455**	11410 *	*8.5e-3/2e5*	.
			100	**12796**	31796	.	.
$f_{\text{Rastrigin}}(x)$	0.9	$[-5.12, 5.12]^n$	20	68586	**12971**	.	9.2e4
		DE: $[-600, 600]^n$	30	147416	**20150** *	1.0e5	2.3e5
			100	1010989	**73620**	.	.
$f_{\text{Rastrigin}}(\boldsymbol{A}x)$	0.9	$[-5.12, 5.12]^n$	30	**152000**	*171/1.25e6* *	.	.
			100	**1011556**	*944/1.25e6* *	.	.
$f_{\text{Schwefel}}(x)$	1e-3	$[-500, 500]^n$	5	43810	**2567** *	.	7.4e4
			10	240899	**5522** *	.	5.6e5

- For “large enough” populations local minima are avoided

- An interesting variant:
  Igel et al.: A Computational Efficient Covariance Matrix Update and a $(1+1)$-CMA for Evolution Strategies, GECCO 2006.

CMA conclusions

- It is a good starting point for an off-the-shelf black-box algorithm
- It includes components like estimating the local gradient (p_σ, p_C), the local "Hessian" (Var(D')), smoothing out local minima (large populations)

Stochastic search conclusions

Input: initial parameter θ, function $f(x)$, distribution model $p_\theta(x)$, update heuristic $h(\theta, D)$

Output: final θ and best point x

1: **repeat**
2: Sample $\{x_i\}_{i=1}^n \sim p_\theta(x)$
3: Evaluate samples, $D = \{(x_i, f(x_i))\}_{i=1}^n$
4: Update $\theta \leftarrow h(\theta, D)$
5: **until** θ converges

- The framework is very general
- The crucial difference between algorithms is their choice of $p_\theta(x)$

RL under Partial Observability

- Data:

$$D = \{(u_t, c_t, y_t)_t\}_{t=0}^T$$

  $\to$ state x_t not observable

- Model-based RL is dauting: Learning $P(x'|u,x)$ and $P(y|u,x)$ with latent x is very hard

- Model-free: The policy needs to map the **history** to a new control

$$\pi:\ (y_{t-h,..,t\text{-}1}, u_{t-h,..,t-1}) \mapsto u_t$$

  or any **features of the history**

$$u_t = \phi(y_{t-h,..,t\text{-}1}, u_{t-h,..,t-1})^\top w$$