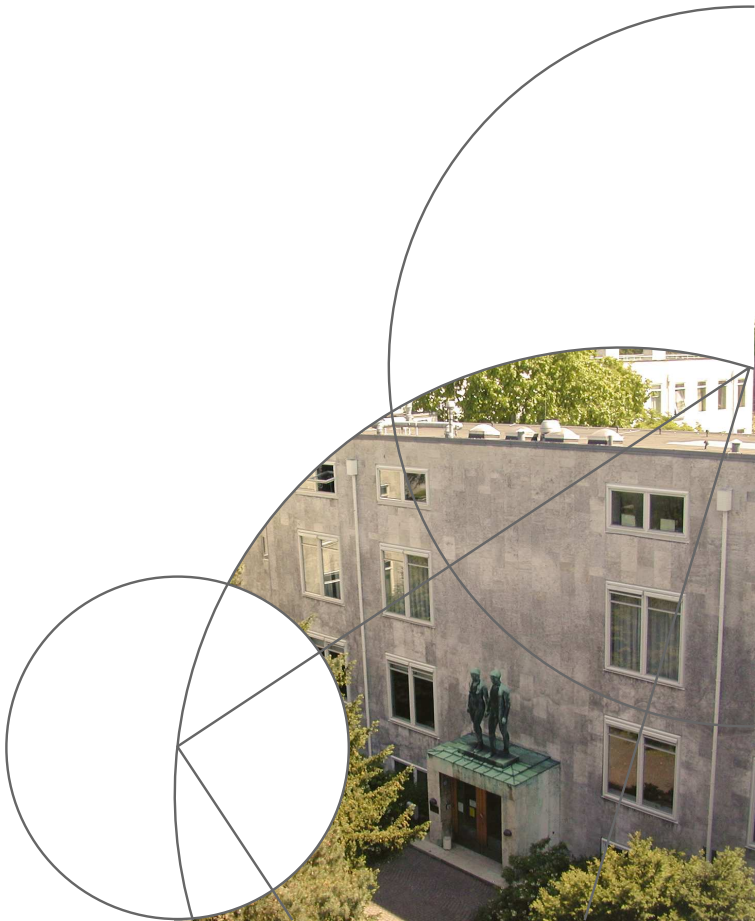
Monte Carlo Methods and Direct Policy Search

Reinforcement Learning

Christian Igel
Department of Computer Science



igel@diku.dk @christian_igel

Outline

Outline

Policy improvement theorem: Idea

- Recall state-value function for a state s given policy π

$$V^\pi(s) = \mathbb{E}_\pi \left\{ R_t \,\middle|\, s_t = s \right\} = \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s \right\}$$

  and suppose we have computed V^π for a deterministic policy π. The value of doing a in s is:

$$\begin{aligned} Q^\pi(s,a) &= \mathbb{E}_\pi \left\{ r_{t+1} + \gamma V^\pi(s_{t+1}) \,\middle|\, s_t = s, a_t = a \right\} \\ &= \sum_{s'} \underbrace{P^a_{ss'}}_{P(s' \mid s,a) = P^a_{ss'}} \Big[\underbrace{R^a_{ss'}}_{R(s,a) = \sum_{s'} P^a_{ss'} R^a_{ss'}} + \gamma V^\pi(s') \Big] \end{aligned}$$

- For a given state s, would it be better to do an action $a \neq \pi(s)$?
- It is better to switch to action a for state s if and only if:

$$Q^\pi(s,a) > V^\pi(s)$$

Policy improvement theorem

Given two policies $\pi, \pi' : S \to A$. If for *all* states $s \in S$

$$Q^{\pi}(s, \pi'(s)) \geq V^{\pi}(s)$$

and for *some* states $s \in S$

$$Q^{\pi}(s, \pi'(s)) > V^{\pi}(s)$$

then for *all* states $s \in S$

$$V^{\pi'}(s) \geq V^{\pi}(s)$$

and for *some* states $s \in S$

$$V^{\pi'}(s) > V^{\pi}(s) \ .$$

Policy improvement theorem proof I

$$
\begin{aligned}
V^\pi(s) &\leq Q^\pi(s, \pi'(s)) \\
&= \mathbb{E}\left\{ r_{t+1} + \gamma V^\pi(s_{t+1}) \,\middle|\, s_t = s, a_t = \pi'(s) \right\} \\
&= \mathbb{E}_{\pi'}\left\{ r_{t+1} + \gamma V^\pi(s_{t+1}) \,\middle|\, s_t = s \right\} \\
&\leq \mathbb{E}_{\pi'}\left\{ r_{t+1} + \gamma Q^\pi(s_{t+1}, \pi'(s_{t+1})) \,\middle|\, s_t = s \right\} \\
&= \mathbb{E}_{\pi'}\left\{ r_{t+1} + \gamma \mathbb{E}\left\{ r_{t+2} + \gamma V^\pi(s_{t+2}) \,\middle|\, s_{t+1}, a_{t+1} = \pi'(s_{t+1}) \right\} \,\middle|\, s_t = s \right\} \\
&= \mathbb{E}_{\pi'}\left\{ r_{t+1} + \gamma r_{t+2} + \gamma^2 V^\pi(s_{t+2})) \,\middle|\, s_t = s \right\} \\
&\leq \mathbb{E}_{\pi'}\left\{ r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 V^\pi(s_{t+3}) \,\middle|\, s_t = s \right\} \\
&\vdots \\
&\leq \mathbb{E}_{\pi'}\left\{ r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \dots \,\middle|\, s_t = s \right\} \\
&= V^{\pi'}(s) \qquad \text{(for all } s \in S\text{)}
\end{aligned}
$$

Policy improvement theorem proof II

Change policy accordingly for all states to get a new policy π' that is greedy with respect to V^π:

$$\begin{aligned}\pi'(s) &= \operatorname*{argmax}_a Q^\pi(s,a) \\ &= \operatorname*{argmax}_a \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V^\pi(s')\right]\end{aligned}$$

Then $\forall s \in S : V^{\pi'}(s) \geq V^\pi(s)$.

$V^{\pi'} = V^\pi$ implies $\forall s \in S : V^\pi(s) = \max_a \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V^\pi(s')\right]$.

That's the Bellman optimality equation and $V^\pi = V^{\pi'} = V^*$. □

Policy iteration: Idea

Alternate:

1. Policy evaluation
2. Policy improvement

$$\pi_0 \underset{\text{evaluate}}{\longrightarrow} V^{\pi_0} \underset{\text{improve}}{\longrightarrow} \pi_1 \underset{\text{evaluate}}{\longrightarrow} V^{\pi_1} \underset{\text{improve}}{\longrightarrow} \dots$$

$$\dots \underset{\text{improve}}{\longrightarrow} \pi^* \underset{\text{evaluate}}{\longrightarrow} V^{\pi^*} \underset{\text{improve}}{\longrightarrow} \pi^*$$

Outline

Monte Carlo methods

Crude Monte Carlo methods approximate an integral

$$\mathbb{E}_{p(z)}[f(z)] = \int f(z)p(z)\mathrm{d}z$$

by

$$\frac{1}{n}\sum_{i=1}^{n} f(z_i) \xrightarrow{n\to\infty} \mathbb{E}_{p(z)}[f(z)] \ ,$$

where

$$z_i \sim p(z)$$

for $i = 1, \ldots, n$.

Monte Carlo methods for RL

- Monte Carlo methods learn from complete sample returns: Integral

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left\{R_t \,\middle|\, s_t = s\right\}$$

  is approximated from trajectories sampled following π and passing through s
- Monte Carlo methods learn directly from experience
  - Simulated: No need for a full model
  - Only defined for episodic tasks

Monte Carlo policy iteration

In Monte Carlo policy iteration we combine:

1. Monte Carlo Policy evaluation
   - Goal: learn $V^\pi(s)$ or $Q^\pi(s,a)$
   - Given: some number of episodes under π which contain s
   - Idea: average returns observed after visits to s
     - Every-Visit MC: average returns for every time s is visited in an episode
     - First-visit MC: average returns only for first time s is visited in an episode
     - Both converge asymptotically
2. Policy improvement

Monte Carlo control

Algorithm 1: Monte Carlo control with exploring starts

Input: arbitrary policy π and state-value function Q

foreach $(s, a) \in S \times A$ **do**
    $R(s, a) \leftarrow 0$ // accumulated returns
    $c(s, a) \leftarrow 0$ // counter
repeat
    generate an episode using exploring starts and π
    foreach *pair* s, a *appearing in the episode* **do**
        $R(s, a) \leftarrow R(s, a) +$ return following first occurrence of s, a
        $c(s, a) \leftarrow c(s, a) + 1$
        $Q(s, a) \leftarrow R(s, a)/c(s, a)$ // policy evaluation
    foreach *state* s *appearing in the episode* **do**
        $\pi(s) \leftarrow \text{argmax}_a Q(s, a)$ // policy improvement
until *some stopping criterion is met*

Output: $\pi \approx \pi^*, Q \approx Q^*$

Sutton & Barto example

Assume you have observed the following 8 state-reward sequences:

A, 0, B, 0	B, 1
B, 1	B, 1
B, 1	B, 1
B, 1	B, 0

What is $V(\mathrm{A})$ and $V(\mathrm{B})$?

Outline

Neo-Darwinean evolution theory 101

- Genome stores information about traits of an individual
- Genome of an individual is based on its parents, but not a perfect copy: errors and new combinations occur
- Individuals better adapted to the environment have higher chances to pass on their genes:

Variation, Drift, and **Selection** lead to **Adaptation**

Variation ... of the genome
Selection ... of better adapted individuals
Adaptation ... to the environment

Well adapted = many offspring who produce fit offspring

- Variation:
  - Mutation: changes in the genetic material of an individual
  - Recombination: genetic material (of several individuals) is split a joined to other genetic material

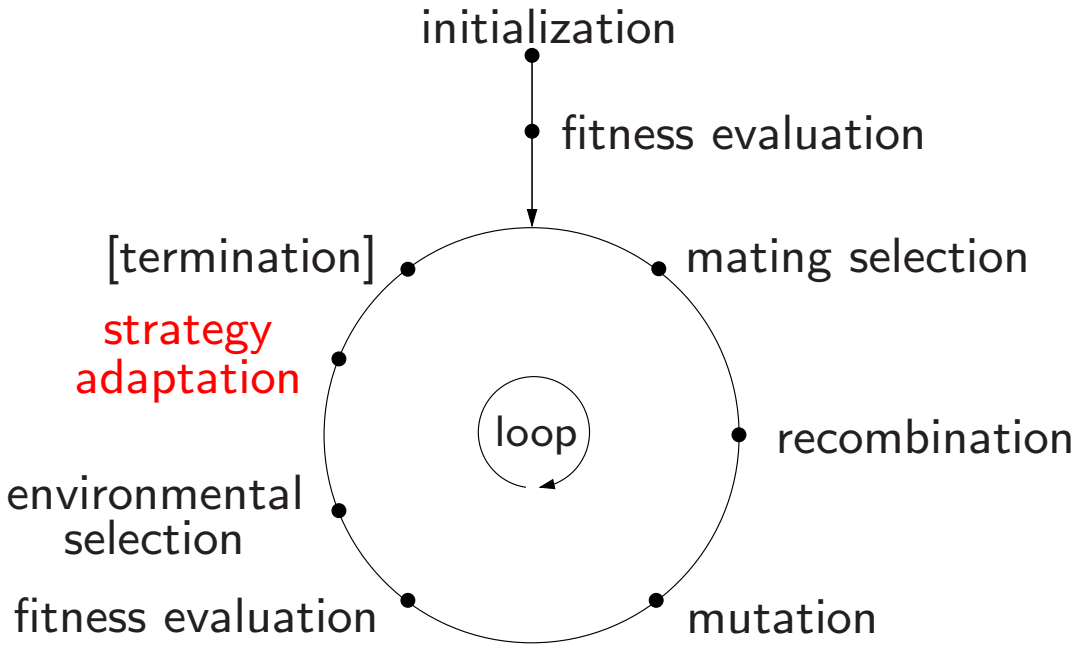
Global random search/optimization

Core of random search/optimization is the search distribution over candidate solutions:

1. Set $t \leftarrow 1$. Choose a joint probability distribution $p^{(t)}_{\mathcal{G}^\lambda}$ on $\mathcal{G}^\lambda$.
2. Obtain λ points $\boldsymbol{g}^{(t)}_1, \ldots, \boldsymbol{g}^{(t)}_\lambda$ by sampling from the distribution $p^{(t)}_{\mathcal{G}^\lambda}$. Evaluate these points using f.
3. According to a fixed (algorithm dependent) rule construct a new probability distribution $p^{(t+1)}_{\mathcal{G}^\lambda}$ on $\mathcal{G}^\lambda$.
4. Check for some appropriate stopping condition; if the algorithm has not terminated, substitute $t \leftarrow t + 1$ and return to step 2.

Canonical evolutionary algorithms (EAs)

- Genomes of individuals encode candidate solutions for given problem.
- Fitness reflects how good problem is solved.



- EAs are global random search/optimization algorithms: Search distribution parameterized by genotypes in parent population and search strategy.

Single-objective optimisation

- Goal: Minimize $f : \mathbb{R}^n \to \mathbb{R}$
- The objective function (equals fitness; convention: lower fitness better) can be
  - non-continuous,
  - multi-modal,
  - noisy,
  - constrained.

Ingredients

- Search space $\mathcal{G}$, fitness function $f : \mathcal{G} \to \mathbb{R}$, variations operators, representation: many possible choices, chosen problem dependent to increase evolvability
- Selection: many possible choices

  Standard examples for environmental selection with μ parents and λ offspring (mating selection random):
  - (μ, λ): best μ of offspring survive $(\mu < \lambda)$
  - $(\mu + \lambda)$: best μ of offspring and parents survive

  Standard example for mating selection (environmental selection: all offspring become the new parents):
  - probability to reproduce is proportional to fitness

Evolution Strategies (ES)

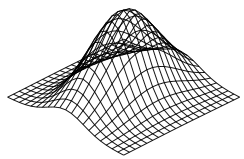
- Offspring $\boldsymbol{x}_k \in \mathbb{R}^n$, $k = 1, \ldots, \lambda$ is generated according to

$$\boldsymbol{x}_k = \boldsymbol{m} + \sigma \boldsymbol{y}_k$$

- Mutation according to multi-variate Gaussian distribution

$$\boldsymbol{y}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{C})$$

$$\sigma \boldsymbol{y}_k \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{C}) =$$



  where $\boldsymbol{C} \in \mathbb{R}^{n \times n}$ and $\sigma \in \mathbb{R}^+$

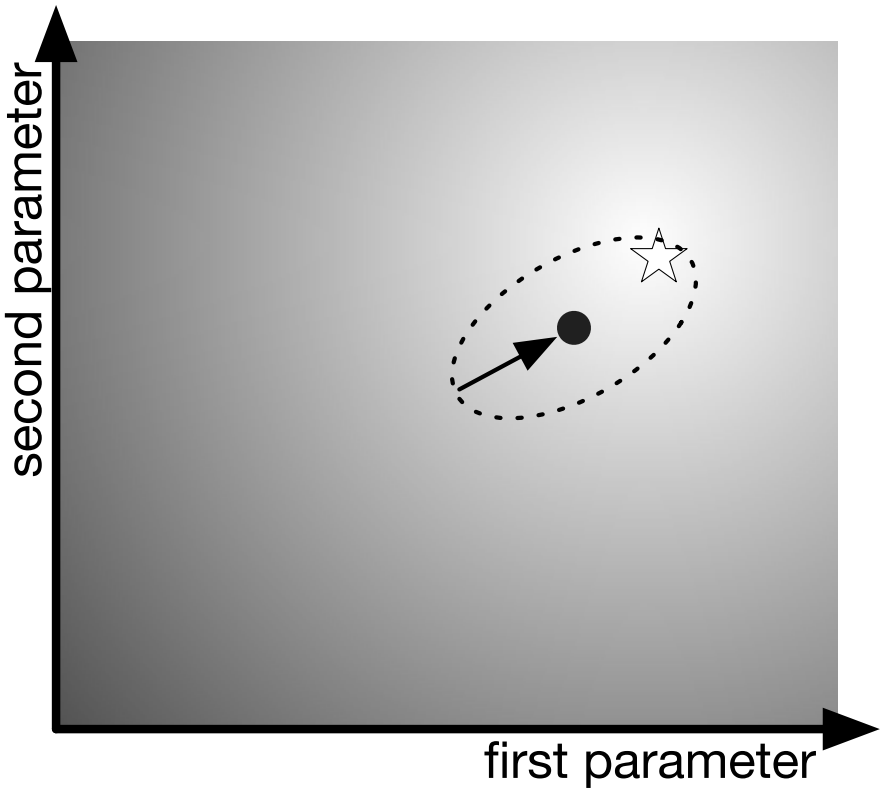
- Weighted global intermediate recombination ($w_1 \geq \cdots \geq w_\mu > 0, \|\boldsymbol{w}\|_1 = 1, \mu < \lambda$):

$$\boldsymbol{m} = \langle \boldsymbol{x}_{\text{parents}} \rangle_{\boldsymbol{w}} = \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i\text{th-best-parent}}$$

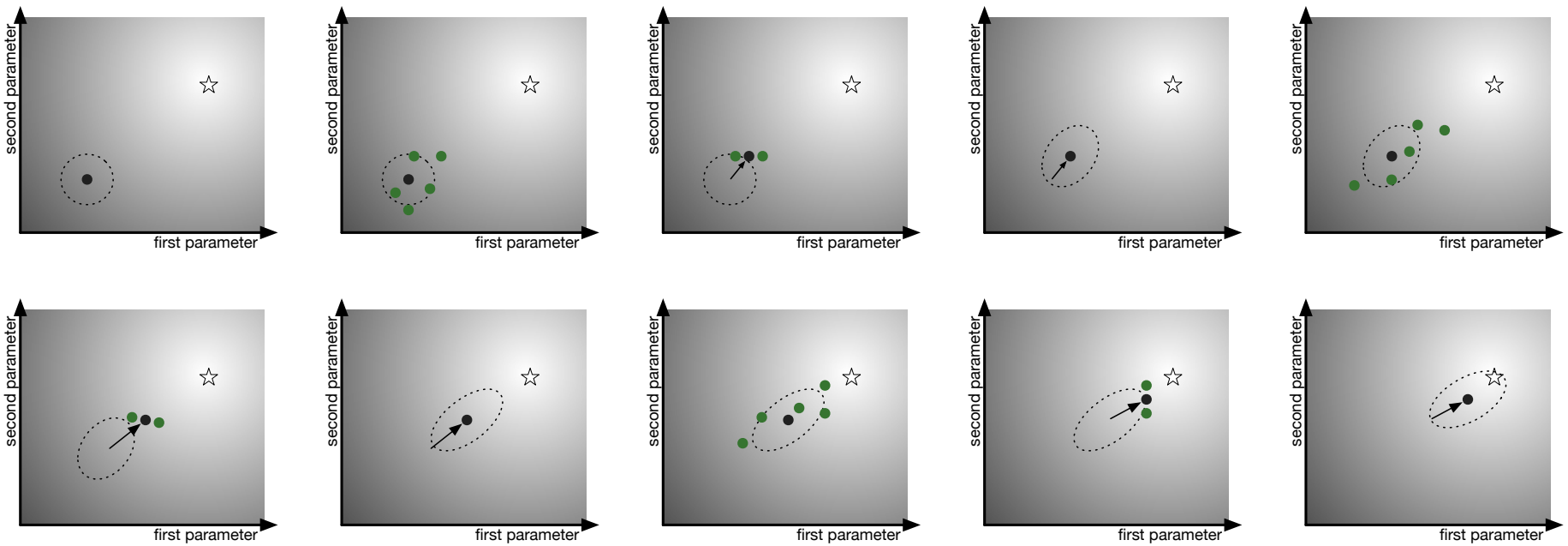
  Implements non-elitist (μ, λ)-selection

Strategy adaptation in ES

- Covariance matrix is updated to make recent beneficial steps more likely ("What has been good in the recent past will be also good in the near future").
- Highly efficient, allows for small population sizes
- Global step size σ and matrix C are updated independently, σ can be adjusted on faster timescale.



Strategy adaptation in ES cartoon



The non-elitist CMA-ES

Mean $\boldsymbol{m} \in \mathbb{R}^n$, scale $\sigma \in \mathbb{R}_+$ and covariance matrix $\boldsymbol{C} \in \mathbb{R}^{n \times n}$ with eigendecomposition

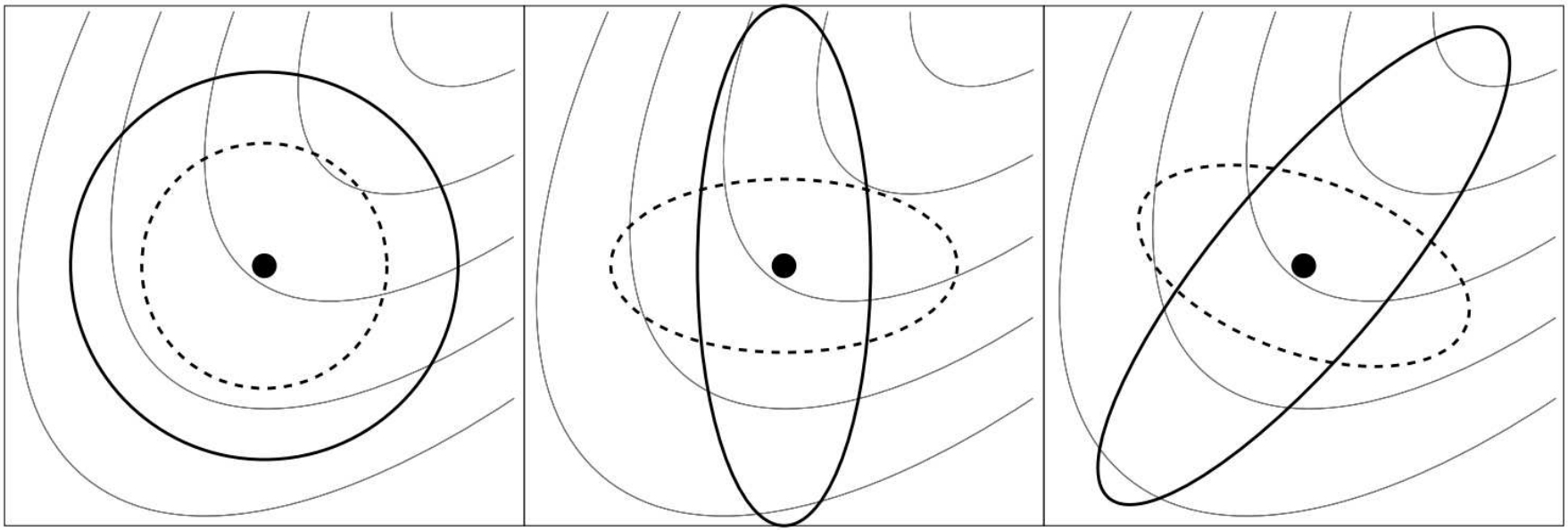
$$\boldsymbol{C} = \boldsymbol{B}\boldsymbol{D}^2\boldsymbol{B}^\mathsf{T}$$

$\boldsymbol{B}$ orthogonal, $\boldsymbol{D}$ diagonal

Sampling k-th offspring:

$$\begin{aligned}
\boldsymbol{z}_k &\sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \quad \text{for } k = 1, ..., \lambda \\
\boldsymbol{y}_k &= \boldsymbol{B}\boldsymbol{D}\boldsymbol{z}_k \sim \mathcal{N}(\mathbf{0}, \boldsymbol{C}) \\
\boldsymbol{x}_k &= \boldsymbol{m} + \sigma \boldsymbol{y}_k \sim \mathcal{N}(\boldsymbol{m}, \sigma^2 \boldsymbol{C})
\end{aligned}$$

Matrix decomposition



$$\sigma \boldsymbol{B}\boldsymbol{D}\mathcal{N}(\mathbf{0}, \boldsymbol{I}) \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{C})$$

Weighted intermediate recombination

- Weights (update considers only μ best offspring):

$$\sum_{i=1}^{\mu} w_i = 1, \quad w_1 \geq w_2 \geq \cdots \geq w_\mu \quad , \quad 0 \leq w_i \leq 1$$

- Update:

$$\boldsymbol{m}^{(g+1)} = \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda}^{(g+1)}$$

$\boldsymbol{x}_{i:\lambda}$: i-th best of the λ offspring

Step by population mean

- Update:

$$\boldsymbol{m}^{(g+1)} = \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda}^{(g+1)}$$

$\sum_{i=1}^{\mu} w_i = 1, \quad w_1 \geq w_2 \geq \cdots \geq w_\mu \quad , \quad 0 \leq w_i \leq 1$

- $\langle \boldsymbol{y} \rangle^{(g+1)}$ is the step of the population mean:

$$\langle \boldsymbol{y} \rangle^{(g+1)} = \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda}^{(g+1)} = \frac{\boldsymbol{m}^{(g+1)} - \boldsymbol{m}^{(g)}}{\sigma^{(g)}}$$

$$\overset{\text{if no selection}}{\sim} \frac{1}{\sqrt{\mu_{\text{eff}}}} \mathcal{N}(\boldsymbol{0}, \boldsymbol{C})$$

- Effective number of parents: $\mu_{\text{eff}} = \left(\sum_{i=1}^{\mu} w_i^2\right)^{-1}, 1 \leq \mu_{\text{eff}} \leq \mu$

Idea: Covariance matrix adaptation

- Rule of thumb: Make recent beneficial steps more likely in the future
- Idea: Low-pass filter evolution path of beneficial directions

$$\boldsymbol{p} \leftarrow \eta_1 \, \boldsymbol{p} + \eta_2 \, \left(\langle \boldsymbol{x}_{\text{new parents}} \rangle_{\boldsymbol{w}} - \langle \boldsymbol{x}_{\text{old parents}} \rangle_{\boldsymbol{w}} \right)$$

and morph $\boldsymbol{C}$ to make direction $\boldsymbol{p}$ more likely:

$$\boldsymbol{C} \leftarrow \eta_3 \, \boldsymbol{C} + \eta_4 \, \boldsymbol{p}\boldsymbol{p}^T$$

with appropriate learning rates and normalisation constants $\eta_1, \ldots, \eta_4$.

- For larger population a rank-μ update increases learning speed.

Rank-1 covariance matrix update

Anisotropic evolution path updated with learning rate c_c:

$$\boldsymbol{p}_c^{(g+1)} = (1 - c_c)\boldsymbol{p}_c^{(g)} + \sqrt{c_c(2 - c_c)\mu_{\text{eff}}}\langle \boldsymbol{y} \rangle^{(g+1)}$$

The factors $(1 - c_c)$ and $\sqrt{c_c(2 - c_c)\mu_{\text{eff}}}$ compensate for changes in variance when adding random variables (see also slide 33).

Rank-1 covariance matrix update with learning rate c_1:

$$\boldsymbol{C}^{(g+1)} = (1 - c_1)\boldsymbol{C}^{(g)} + c_1 \boldsymbol{p}_c^{(g+1)} \boldsymbol{p}_c^{(g+1)^{\mathsf{T}}}$$

Rank-μ covariance matrix update

Rank-μ covariance matrix update with learning rate c_μ:

$$\boldsymbol{C}^{(g+1)} = (1 - c_\mu)\boldsymbol{C}^{(g)} + c_\mu \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda}^{(g+1)} \boldsymbol{y}_{i:\lambda}^{(g+1)\mathsf{T}}$$

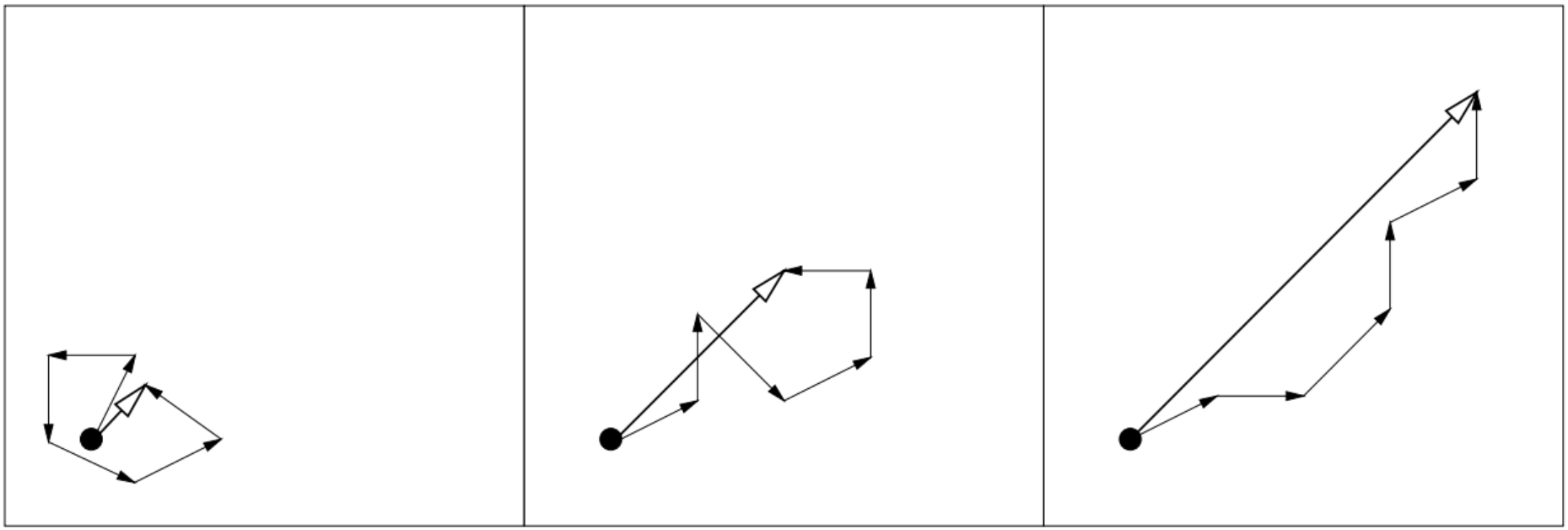
Consider the individual steps that generated the most fit offspring in the last generation to adapt covariance.

(Almost) final covariance matrix update

$$\boldsymbol{p}_c^{(g+1)} = (1 - c_c)\boldsymbol{p}_c^{(g)} + \sqrt{c_c(2 - c_c)\mu_{\mathsf{eff}}}\langle \boldsymbol{y} \rangle^{(g+1)}$$

$$\boldsymbol{C}^{(g+1)} = (1 - c_1 - c_\mu)\boldsymbol{C}^{(g)} + c_1 \left(\boldsymbol{p}_c^{(g+1)} \boldsymbol{p}_c^{(g+1)^\mathsf{T}} \right) + c_\mu \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda} \boldsymbol{y}_{i:\lambda}^\mathsf{T}$$

Step sizes



Isotropic evolution path

Evolution paths allow to take previous steps in the search space into account.
Learned using exponential smoothing and learning rate c_σ:

$$\boldsymbol{p}_\sigma^{(g+1)} = (1-c_\sigma)\boldsymbol{p}_\sigma^{(g)} + \sqrt{c_\sigma(2-c_\sigma)\mu_{\text{eff}}}\boldsymbol{C}^{(g)^{-\frac{1}{2}}}\langle \boldsymbol{y} \rangle^{(g+1)}$$

The factors $(1-c_\sigma)$ and $\sqrt{c_\sigma(2-c_\sigma)\mu_{\text{eff}}}\boldsymbol{C}^{(g)^{-\frac{1}{2}}}$ ensure $\boldsymbol{p}_\sigma^{(g+1)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ when $\boldsymbol{p}_\sigma^{(g)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$:

$$\begin{aligned}\boldsymbol{p}_\sigma^{(g+1)} &\sim (1-c_\sigma)\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) + \sqrt{c_\sigma(2-c_\sigma)\mu_{\text{eff}}}\boldsymbol{C}^{(g)^{-\frac{1}{2}}}\frac{1}{\sqrt{\mu_{\text{eff}}}}\mathcal{N}(\boldsymbol{0}, \boldsymbol{C}^{(g)}) \\ &\sim \mathcal{N}\left(\boldsymbol{0}, \left((1-c_\sigma)^2 + \sqrt{c_\sigma(2-c_\sigma)}^2\right)\boldsymbol{I}\right) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})\end{aligned}$$

Cumulative step size adaptation

Compare length of evolution path with expected length under random selection:

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma^{(g+1)}\|}{\mathsf{E}\|\mathcal{N}(\mathbf{0},\boldsymbol{I})\|} - 1\right)\right)$$

If $\|\boldsymbol{p}_\sigma^{(g+1)}\| > \mathsf{E}\|\mathcal{N}(\mathbf{0},\boldsymbol{I})\|$, step size becomes larger.
If $\|\boldsymbol{p}_\sigma^{(g+1)}\| < \mathsf{E}\|\mathcal{N}(\mathbf{0},\boldsymbol{I})\|$, step size becomes smaller.
If $\|\boldsymbol{p}_\sigma^{(g+1)}\| = \mathsf{E}\|\mathcal{N}(\mathbf{0},\boldsymbol{I})\|$, step size is unchanged.

$0 < \frac{c_\sigma}{d_\sigma} < 1$ is a damping factor for the scale changes.

Implements the rules:

- Steps large and/or correlated $\rightarrow$ increase
- Steps small and/or anti-correlated $\rightarrow$ decrease

Default parameters

Choose $\sigma^{(0)}$ and $\boldsymbol{m}^{(0)}$ dependent on problem.
Initialize with $\boldsymbol{p}_c^{(0)} = 0$, $\boldsymbol{p}_\sigma^{(0)} = 0$, $\boldsymbol{C}^{(0)} = \boldsymbol{I}$.

$$\lambda = 4 + \lfloor 3 \ln n \rfloor, \quad \mu = \lfloor \mu' \rfloor, \quad \mu' = \frac{\lambda}{2}$$

$$w_i = \frac{w_i'}{\sum_{j=1}^{\mu} w_j'}, \quad w_i' = \ln(\mu' + 0.5) - \ln i \quad \text{for } i = 1, \ldots, \mu$$

$$c_\sigma = \frac{\mu_{\text{eff}} + 2}{n + \mu_{\text{eff}} + 5}, \quad d_\sigma = 1 + 2 \max\left(0, \sqrt{\frac{\mu_{\text{eff}} - 1}{n+1}} - 1\right) + c_\sigma$$

$$c_c = \frac{4 + \frac{\mu_{\text{eff}}}{n}}{n + 4 + 2\frac{\mu_{\text{eff}}}{n}}, \quad c_1 = \frac{2}{(n+1.3)^2 + \mu_{\text{eff}}}$$

$$c_\mu = \min\left(1 - c_1, \alpha_\mu \frac{\mu_{\text{eff}} - 2 + \frac{1}{\mu_{\text{eff}}}}{(n+2)^2 + \alpha_\mu \frac{\mu_{\text{eff}}}{2}}\right), \quad \alpha_\mu = 2$$

CMA-ES

The CMA-ES

- represents "the state-of-the-art in evolutionary optimisation in real-valued $\mathbb{R}^n$ search spaces",

  Beyer: Evolution Strategies, Scholarpedia 2(8)

  is "widely regarded as the state of the art in numerical optimization",

  Eiben, Smith: From evolutionary computation to the evolution of things, Nature 521, 2015

- is ("almost") invariant under translation, rotation, flipping of search space and under order preserving transformations of objective function, and
- can be turned into a MO-CMA-ES for vector optimisation.

Hansen, Ostermeier: Completely Derandomized Self-Adaptation in Evolution Strategies. Evolut Comput 9(2), 2001.

Hansen et al.: Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES), Evolut Comput, 11(1), 2003.

Auger, Hansen: A Restart CMA Evolution Strategy With Increasing Population Size, CEC 2005, IEEE Press.

Igel, Hansen, Roth: Covariance Matrix Adaptation for Multi-objective Optimization, Evolut Comput 15(1), 2007.

Outline

CMA-ES for reinforcement learning

- Idea: Apply CMA-ES to reinforcement learning (RL, i.e., solving stochastic decision processes)!

  Igel: Neuroevolution for Reinforcement Learning Using Evolution Strategies, Proc. IEEE CEC, 2003

- CMA-ES is used for direct policy search and searches a fixed parameterized class of functions.
- When ESs optimize neural networks we talk of Neuroevolution Strategies (NeuroESs).

  Heidrich-Meisner & Igel: Neuroevolution Strategies for Episodic Reinforcement Learning. Journal of Algorithms 64, 2009

- CMA-ES is related to variable metric policy gradient methods (PGMs), in particular to the episodic natural actor-critic (NAC) algorithm.

  Peters, Schaal: Natural actor-critic, Neurocomputing 71(7-9), 2008

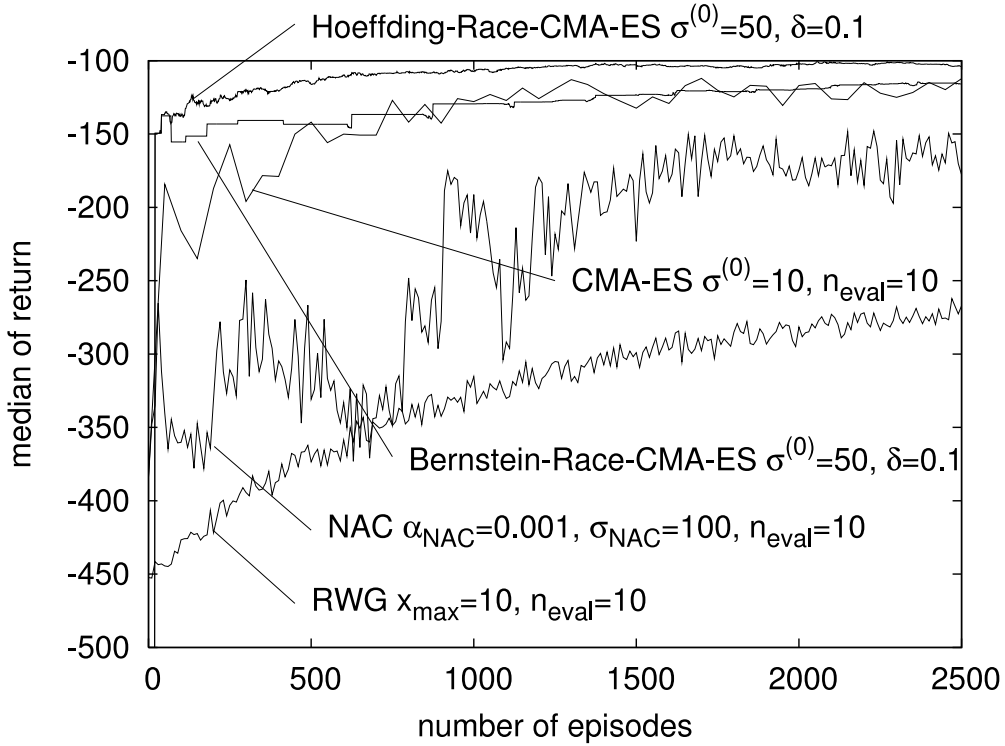
Empirical results

There are several (independent) evaluations of the CMA-ES for RL, all give excellent results.

non-Markovian double pole balancing, recurrent neural network policies

method	reward function	
	standard	damping
RWG	415,209	1,232,296
SANE	262,700	451,612
ESP	7,374	26,342
NEAT	–	6,929
RPG	(5,649)	–
CoSyNE	1,249	3,416
CMA-ES	860	1,141

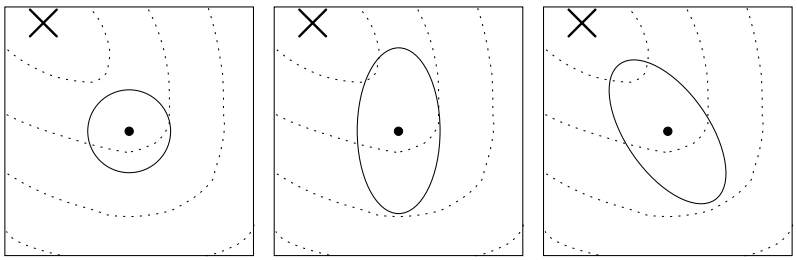
noisy Mountain-car

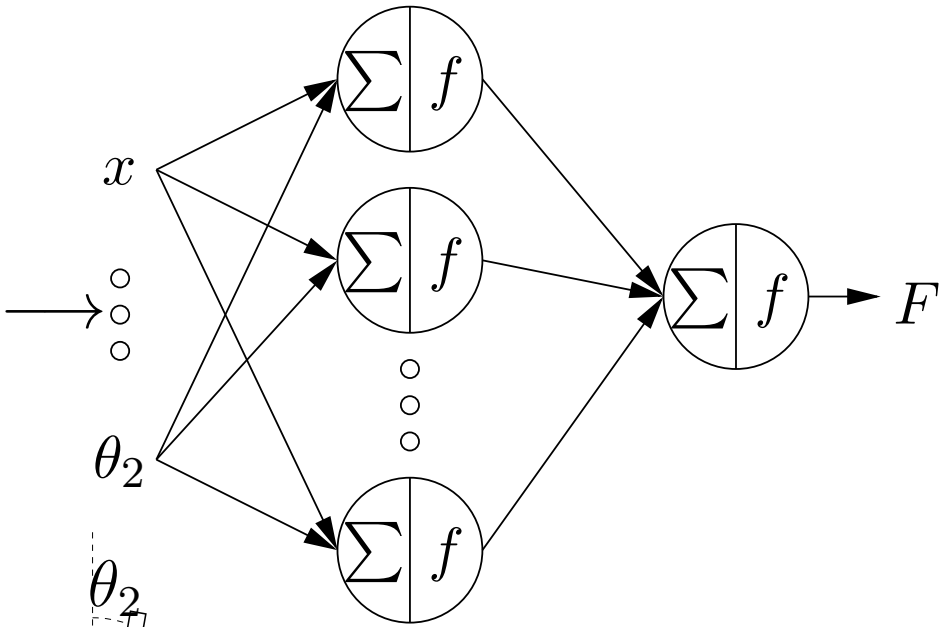


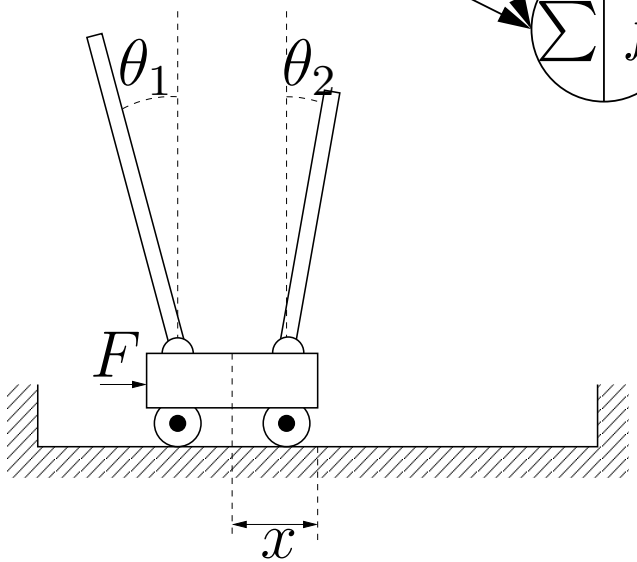
Heidrich-Meisner, Igel: Neuroevolution Strategies for Episodic Reinforcement Learning. *Journal of Algorithms* 64, 2009

Heidrich-Meisner, Igel: Hoeffding and Bernstein Races for Selecting Policies in Evolutionary Direct Policy Search, *ICML*, 2009

Experimental setup







Results: Incomplete state information I

Single Pole	
Method	Evaluations
SARSA-CMAC	13,562
Q-MLP	11,331
RWG	8,557
RPG	(1,893)
NEAT	1,523
SANE	1,212
CNE	724
ESP	589
CMA-NeuroES	192
CoSyNE	127

Results: Incomplete state information II

Double Pole		
Method	Evaluations standard	Evaluations damping
RWG	415,209	1,232,296
CE	–	(840,000)
SANE	262,700	451,612
CNE	76,906	87,623
ESP	7,374	26,342
NEAT	–	6,929
RPG	(5,649)	–
CoSyNE	1,249	3,416
CMA-NeuroES	860	1,140

Limitations

- CMA-NeuroES relies on fixed network structures.

  Siebel, Sommer: Evolutionary reinforcement learning of artificial neural networks, International Journal of Hybrid Intelligent Systems 4(3), 2007

  Togelius, Schaul, Schmidhuber, Gomez: Countering Poisonous Inputs with Memetic Neuroevolution, LNCS 5199, 2008

- CMA-ES does not exploit intermediate rewards, just final Monte Carlo returns.
- CMA-ES has no concept of states, which would support credit assignment.
- CMA-ES in its canonical form is just defined for episodic tasks.
- If the search space dimension gets too large, you cannot adapt the full covariance matrix anymore. However, you can restrict the CMA-ES to diagonal matrices.

The CMA-NeuroES...

1. is a variable metric algorithm learning correlations between parameters,
2. extracts a search direction from the scalar reward signals,
3. is robust w.r.t. tuning of meta-parameters,
4. is based on ranking policies, which is robust w.r.t noise/uncertainty, and allows efficient uncertainty handling and distribution of roll-outs

   Heidrich-Meisner, Igel: Hoeffding and Bernstein Races for Selecting Policies in Evolutionary Direct Policy Search, ICML, 2009
5. is applicable if the function approximators are non-differentiable,
6. performs great for episodic tasks with short episodes or in which the intermediate rewards do not carry (much) more information than their sum and the number of policy parameters is not too large.

Further reading

R. S. Sutton, A. G. Barto. Reinforcement Learning: An Introduction. 2nd edition, MIT Press, 2018

http://incompleteideas.net/book/RLbook2020.pdf

N. Hansen: The CMA Evolution Strategy: A Tutorial

https://arxiv.org/abs/1604.00772