CMA-ES and Advanced Adaptation Mechanisms

Youhei Akimoto[1] & Anne Auger[2] & Nikolaus Hansen[2]
1. Shinshu University, Nagano, Japan
2. Inria, Research Centre Saclay, France

y_akimoto@shinshu-u.ac.jp
anne.auger@lri.fr
nikolaus.hansen@lri.fr

We are happy to answer questions at any time.

Problem Statement

Continuous Domain Search/Optimization

- Task: minimize an objective function (*fitness* function, *loss* function) in continuous domain

$$f : \mathcal{X} \subseteq \mathbb{R}^n \to \mathbb{R}, \qquad \boldsymbol{x} \mapsto f(\boldsymbol{x})$$

- Black Box scenario (direct search scenario)

  - gradients are not available or not useful
  - problem domain specific knowledge is used only within the black box, e.g. within an appropriate encoding
- Search costs: number of function evaluations

Problem Statement

Continuous Domain Search/Optimization

- Goal
  - fast convergence to the global optimum
  - solution x with small function value $f(x)$ with least search cost
    ... or to a robust solution x
    there are two conflicting objectives
- Typical Examples
  - shape optimization (e.g. using CFD) — curve fitting, airfoils
  - model calibration — biological, physical
  - parameter calibration — controller, plants, images
- Problems
  - exhaustive search is infeasible
  - naive random search takes too long
  - deterministic search is not successful / takes too long

Approach: stochastic search, Evolutionary Algorithms

Problem Statement | Black Box Optimization and Its Difficulties

Problem Statement
Continuous Domain Search/Optimization
Goal
fast convergence to the global optimum
solution x with small function value $f(x)$ with least search cost
... or to a robust solution x
there are two conflicting objectives
Typical Examples
shape optimization (e.g. using CFD) — curve fitting, airfoils
model calibration — biological, physical
parameter calibration — controller, plants, images
Problems
exhaustive search is infeasible
naive random search takes too long
deterministic search is not successful / takes too long
Approach: stochastic search, Evolutionary Algorithms

Problem Statement | Black Box Optimization and Its Difficulties

Problem Statement
Continuous Domain Search/Optimization
Goal
fast convergence to the global optimum
solution x with small function value $f(x)$ with least search cost
... or to a robust solution x
there are two conflicting objectives
Typical Examples
shape optimization (e.g. using CFD) — curve fitting, airfoils
model calibration — biological, physical
parameter calibration — controller, plants, images
Problems
exhaustive search is infeasible
naive random search takes too long
deterministic search is not successful / takes too long
Approach: stochastic search, Evolutionary Algorithms

Objective Function Properties

We assume $f : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$ to be *non-linear, non-separable* and to have at least moderate dimensionality, say $n \not\ll 10$.

Additionally, f can be

- non-convex
- multimodal
  there are possibly many local optima
- non-smooth
  derivatives do not exist
- discontinuous, plateaus
- ill-conditioned
- noisy
- ...

Goal : cope with any of these function properties
they are related to real-world problems

Objective Function Properties

We assume $f : \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}$ to be *non-linear, non-separable* and to have at least moderate dimensionality, say $n \not\ll 10$.
Additionally, f can be

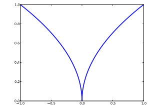
- non-convex
- multimodal
  there are possibly many local optima
- non-smooth
  derivatives do not exist
- discontinuous, plateaus
- ill-conditioned
- noisy
- ...

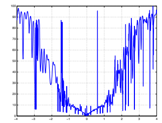
Goal: cope with any of these function properties
they are related to real-world problems

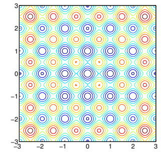
What Makes a Function Difficult to Solve?

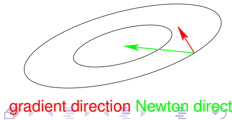
Why stochastic search?

- non-linear, non-quadratic, non-convex
  on linear and quadratic functions much better search policies are available
- ruggedness
  non-smooth, discontinuous, multimodal, and/or noisy function
- dimensionality (size of search space)
  (considerably) larger than three
- non-separability
  dependencies between the objective variables
- ill-conditioning









What Makes a Function Difficult to Solve?

... and what can be done

The Problem	Possible Approaches
Dimensionality	exploiting the problem structure separability, locality/neighborhood, encoding
Ill-conditioning	second order approach changes the neighborhood metric
Ruggedness	non-local policy, large sampling width (step-size) as large as possible while preserving a reasonable convergence speed
	population-based method, stochastic, non-elitistic
	recombination operator serves as repair mechanism
	restarts

...metaphors

What Makes a Function Difficult to Solve?

... and what can be done

The Problem	Possible Approaches
Dimensionality	exploiting the problem structure separability, locality/neighborhood, encoding
Ill-conditioning	second order approach changes the neighborhood metric
Ruggedness	non-local policy, large sampling width (step-size) as large as possible while preserving a reasonable convergence speed
	population-based method, stochastic, non-elitistic
	recombination operator serves as repair mechanism
	restarts

...metaphors

What Makes a Function Difficult to Solve?

. . . and what can be done

The Problem	Possible Approaches
Dimensionality	exploiting the problem structure separability, locality/neighborhood, encoding
Ill-conditioning	second order approach changes the neighborhood metric
Ruggedness	non-local policy, large sampling width (step-size) as large as possible while preserving a reasonable convergence speed
	population-based method, stochastic, non-elitistic
	recombination operator serves as repair mechanism
	restarts

. . . metaphors

Questions?

Stochastic Search

A black box search template to minimize $f : \mathbb{R}^n \to \mathbb{R}$

Initialize distribution parameters $\boldsymbol{\theta}$, set population size $\lambda \in \mathbb{N}$
While not terminate

1. Sample distribution $P(\boldsymbol{x}|\boldsymbol{\theta}) \to \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda \in \mathbb{R}^n$
2. Evaluate $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda$ on f
3. Update parameters $\boldsymbol{\theta} \leftarrow F_\theta(\boldsymbol{\theta}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda, f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_\lambda))$

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

In many Evolutionary Algorithms the distribution P is implicitly defined via operators on a population, in particular, selection, recombination and mutation

Natural template for (incremental) *Estimation of Distribution Algorithms*

Stochastic Search

A black box search template to minimize $f : \mathbb{R}^n \to \mathbb{R}$

Initialize distribution parameters θ, set population size $\lambda \in \mathbb{N}$
While not terminate

1. Sample distribution $P(\mathbf{x}|\theta) \to \mathbf{x}_1, \dots, \mathbf{x}_\lambda \in \mathbb{R}^n$
2. Evaluate $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ on f
3. Update parameters $\theta \leftarrow F_\theta(\theta, \mathbf{x}_1, \dots, \mathbf{x}_\lambda, f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda))$

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

In many Evolutionary Algorithms the distribution P is implicitly defined via operators on a population, in particular, selection, recombination and mutation

Natural template for (incremental) *Estimation of Distribution Algorithms*

Stochastic Search

A black box search template to minimize $f : \mathbb{R}^n \to \mathbb{R}$

Initialize distribution parameters θ, set population size $\lambda \in \mathbb{N}$
While not terminate

1. Sample distribution $P(\mathbf{x}|\theta) \to \mathbf{x}_1, \dots, \mathbf{x}_\lambda \in \mathbb{R}^n$
2. Evaluate $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ on f
3. Update parameters $\theta \leftarrow F_\theta(\theta, \mathbf{x}_1, \dots, \mathbf{x}_\lambda, f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda))$

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

In many Evolutionary Algorithms the distribution P is implicitly defined via operators on a population, in particular, selection, recombination and mutation

Natural template for (incremental) *Estimation of Distribution Algorithms*

Stochastic Search

A black box search template to minimize $f : \mathbb{R}^n \to \mathbb{R}$

Initialize distribution parameters θ, set population size $\lambda \in \mathbb{N}$
While not terminate

1. Sample distribution $P(\mathbf{x}|\theta) \to \mathbf{x}_1, \dots, \mathbf{x}_\lambda \in \mathbb{R}^n$
2. Evaluate $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ on f
3. Update parameters $\theta \leftarrow F_\theta(\theta, \mathbf{x}_1, \dots, \mathbf{x}_\lambda, f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda))$

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

In many Evolutionary Algorithms the distribution P is implicitly defined via operators on a population, in particular, selection, recombination and mutation

Natural template for (incremental) *Estimation of Distribution Algorithms*

Stochastic Search

A black box search template to minimize $f : \mathbb{R}^n \to \mathbb{R}$

Initialize distribution parameters θ, set population size $\lambda \in \mathbb{N}$
While not terminate

1. Sample distribution $P(\mathbf{x}|\theta) \to \mathbf{x}_1, \dots, \mathbf{x}_\lambda \in \mathbb{R}^n$
2. Evaluate $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ on f
3. Update parameters $\theta \leftarrow F_\theta(\theta, \mathbf{x}_1, \dots, \mathbf{x}_\lambda, f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda))$

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

In many Evolutionary Algorithms the distribution P is implicitly defined via operators on a population, in particular, selection, recombination and mutation

Natural template for (incremental) *Estimation of Distribution Algorithms*

Stochastic Search

A black box search template to minimize $f : \mathbb{R}^n \to \mathbb{R}$

Initialize distribution parameters θ, set population size $\lambda \in \mathbb{N}$
While not terminate

1. Sample distribution $P(\boldsymbol{x}|\theta) \to \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda \in \mathbb{R}^n$
2. Evaluate $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda$ on f
3. Update parameters $\theta \leftarrow F_\theta(\theta, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda, f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_\lambda))$

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

In many Evolutionary Algorithms the distribution P is implicitly defined via operators on a population, in particular, selection, recombination and mutation

Natural template for (incremental) *Estimation of Distribution Algorithms*

Stochastic Search

A black box search template to minimize $f : \mathbb{R}^n \to \mathbb{R}$

Initialize distribution parameters θ, set population size $\lambda \in \mathbb{N}$
While not terminate

1. Sample distribution $P(\boldsymbol{x}|\theta) \to \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda \in \mathbb{R}^n$
2. Evaluate $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda$ on f
3. Update parameters $\theta \leftarrow F_\theta(\theta, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda, f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_\lambda))$

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

In many Evolutionary Algorithms the distribution P is implicitly defined via operators on a population, in particular, selection, recombination and mutation

Natural template for (incremental) *Estimation of Distribution Algorithms*

Stochastic Search

A black box search template to minimize $f : \mathbb{R}^n \to \mathbb{R}$

Initialize distribution parameters θ, set population size $\lambda \in \mathbb{N}$
While not terminate

1. Sample distribution $P(\boldsymbol{x}|\theta) \to \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda \in \mathbb{R}^n$
2. Evaluate $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda$ on f
3. Update parameters $\theta \leftarrow F_\theta(\theta, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda, f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_\lambda))$

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

In many Evolutionary Algorithms the distribution P is implicitly defined via operators on a population, in particular, selection, recombination and mutation

Natural template for (incremental) *Estimation of Distribution Algorithms*

The CMA-ES

Input: $\boldsymbol{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, λ
Initialize: $\mathbf{C} = \mathbf{I}$, and $\boldsymbol{p}_c = \mathbf{0}$, $\boldsymbol{p}_\sigma = \mathbf{0}$,
Set: $c_c \approx 4/n$, $c_\sigma \approx 4/n$, $c_1 \approx 2/n^2$, $c_\mu \approx \mu_w/n^2$, $c_1 + c_\mu \le 1$, $d_\sigma \approx 1 + \sqrt{\frac{\mu_w}{n}}$,
and $w_{i=1\ldots\lambda}$ such that $\mu_w = \frac{1}{\sum_{i=1}^{\mu} w_i^2} \approx 0.3\,\lambda$

While not terminate

$$\boldsymbol{x}_i = \boldsymbol{m} + \sigma \boldsymbol{y}_i, \quad \boldsymbol{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C}), \quad \text{for } i = 1, \ldots, \lambda \qquad \text{sampling}$$

$$\boldsymbol{m} \leftarrow \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda} = \boldsymbol{m} + \sigma \boldsymbol{y}_w \quad \text{where } \boldsymbol{y}_w = \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda} \qquad \text{update mean}$$

$$\boldsymbol{p}_c \leftarrow (1 - c_c)\,\boldsymbol{p}_c + \mathbb{1}_{\{\|\boldsymbol{p}_\sigma\| < 1.5\sqrt{n}\}} \sqrt{1 - (1 - c_c)^2} \sqrt{\mu_w}\, \boldsymbol{y}_w \qquad \text{cumulation for } \mathbf{C}$$

$$\boldsymbol{p}_\sigma \leftarrow (1 - c_\sigma)\,\boldsymbol{p}_\sigma + \sqrt{1 - (1 - c_\sigma)^2} \sqrt{\mu_w}\, \mathbf{C}^{-\frac{1}{2}} \boldsymbol{y}_w \qquad \text{cumulation for } \sigma$$

$$\mathbf{C} \leftarrow (1 - c_1 - c_\mu)\,\mathbf{C} + c_1 \boldsymbol{p}_c \boldsymbol{p}_c^{\mathrm{T}} + c_\mu \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda} \boldsymbol{y}_{i:\lambda}^{\mathrm{T}} \qquad \text{update } \mathbf{C}$$

$$\sigma \leftarrow \sigma \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma\|}{\mathsf{E}\|\mathcal{N}(\mathbf{0},\mathbf{I})\|} - 1\right)\right) \qquad \text{update of } \sigma$$

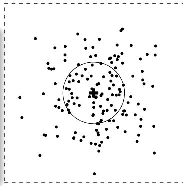
Not covered on this slide: termination, restarts, useful output, boundaries and encoding

Evolution Strategies

New search points are sampled normally distributed

$$\boldsymbol{x}_i \sim \boldsymbol{m} + \sigma\,\mathcal{N}_i(\mathbf{0}, \mathbf{C}) \qquad \text{for } i = 1, \ldots, \lambda$$

as perturbations of $\boldsymbol{m}$, where $\boldsymbol{x}_i, \boldsymbol{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, $\mathbf{C} \in \mathbb{R}^{n \times n}$



where

- the mean vector $\boldsymbol{m} \in \mathbb{R}^n$ represents the favorite solution
- the so-called step-size $\sigma \in \mathbb{R}_+$ controls the *step length*
- the covariance matrix $\mathbf{C} \in \mathbb{R}^{n \times n}$ determines the shape of the distribution ellipsoid

here, all new points are sampled with the same parameters

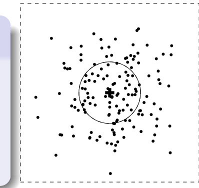
The question remains how to update $\boldsymbol{m}$, $\mathbf{C}$, and σ.

Evolution Strategies

New search points are sampled normally distributed

$$\boldsymbol{x}_i \sim \boldsymbol{m} + \sigma\,\mathcal{N}_i(\mathbf{0}, \mathbf{C}) \qquad \text{for } i = 1, \ldots, \lambda$$

as perturbations of $\boldsymbol{m}$, where $\boldsymbol{x}_i, \boldsymbol{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, $\mathbf{C} \in \mathbb{R}^{n \times n}$



where

- the mean vector $\boldsymbol{m} \in \mathbb{R}^n$ represents the favorite solution
- the so-called step-size $\sigma \in \mathbb{R}_+$ controls the *step length*
- the covariance matrix $\mathbf{C} \in \mathbb{R}^{n \times n}$ determines the shape of the distribution ellipsoid

here, all new points are sampled with the same parameters

The question remains how to update $\boldsymbol{m}$, $\mathbf{C}$, and σ.

The $(\mu/\mu, \lambda)$-ES

Non-elitist selection and intermediate (weighted) recombination

Given the i-th solution point $\boldsymbol{x}_i = \boldsymbol{m} + \sigma\,\underbrace{\mathcal{N}_i(\mathbf{0}, \mathbf{C})}_{=:\boldsymbol{y}_i} = \boldsymbol{m} + \sigma\,\boldsymbol{y}_i$

Let $\boldsymbol{x}_{i:\lambda}$ the i-th ranked solution point, such that $f(\boldsymbol{x}_{1:\lambda}) \leq \cdots \leq f(\boldsymbol{x}_{\lambda:\lambda})$.
The new mean reads

$$\boldsymbol{m} \leftarrow \sum_{i=1}^{\mu} w_i\,\boldsymbol{x}_{i:\lambda}$$

where

$$w_1 \geq \cdots \geq w_\mu > 0, \quad \textstyle\sum_{i=1}^{\mu} w_i = 1, \quad \frac{1}{\sum_{i=1}^{\mu} {w_i}^2} =: \mu_w \approx \frac{\lambda}{4}$$

The best μ points are selected from the new solutions (non-elitistic) and weighted intermediate recombination is applied.

The $(\mu/\mu, \lambda)$-ES

Non-elitist selection and intermediate (weighted) recombination

Given the i-th solution point $\boldsymbol{x}_i = \boldsymbol{m} + \sigma\,\underbrace{\mathcal{N}_i(\mathbf{0}, \mathbf{C})}_{=:\boldsymbol{y}_i} = \boldsymbol{m} + \sigma\,\boldsymbol{y}_i$

Let $\boldsymbol{x}_{i:\lambda}$ the i-th ranked solution point, such that $f(\boldsymbol{x}_{1:\lambda}) \leq \cdots \leq f(\boldsymbol{x}_{\lambda:\lambda})$.
The new mean reads

$$\boldsymbol{m} \leftarrow \sum_{i=1}^{\mu} w_i\,\boldsymbol{x}_{i:\lambda}$$

where

$$w_1 \geq \cdots \geq w_\mu > 0, \quad \textstyle\sum_{i=1}^{\mu} w_i = 1, \quad \frac{1}{\sum_{i=1}^{\mu} {w_i}^2} =: \mu_w \approx \frac{\lambda}{4}$$

The best μ points are selected from the new solutions (non-elitistic) and weighted intermediate recombination is applied.

The $(\mu/\mu, \lambda)$-ES

Non-elitist selection and intermediate (weighted) recombination

Given the i-th solution point $\boldsymbol{x}_i = \boldsymbol{m} + \sigma \underbrace{\mathcal{N}_i(\mathbf{0}, \mathbf{C})}_{=: \boldsymbol{y}_i} = \boldsymbol{m} + \sigma \boldsymbol{y}_i$

Let $\boldsymbol{x}_{i:\lambda}$ the i-th ranked solution point, such that $f(\boldsymbol{x}_{1:\lambda}) \leq \cdots \leq f(\boldsymbol{x}_{\lambda:\lambda})$.
The new mean reads

$$\boldsymbol{m} \leftarrow \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda} = \boldsymbol{m} + \sigma \underbrace{\sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda}}_{=: \boldsymbol{y}_w}$$

where

$$w_1 \geq \cdots \geq w_\mu > 0, \quad \sum_{i=1}^{\mu} w_i = 1, \quad \frac{1}{\sum_{i=1}^{\mu} {w_i}^2} =: \mu_w \approx \frac{\lambda}{4}$$

The best μ points are selected from the new solutions (non-elitistic) and weighted intermediate recombination is applied.

Problem Statement

Evolution Strategy (ES)

Step-Size Adaptation

- Why Step-Size Control
- Path Length Control (CSA)
- Limitations of CSA and its Alternatives

Covariance Matrix Adaptation (CMA)

- Rank-One Update and Cumulation
- Rank-μ Update
- Active Covariance Update

Design Principle

- Theoretical Foundations
- Variants for Large Scale Problems

Summary and Final Remarks

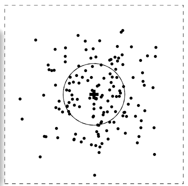
Evolution Strategies

Recalling

New search points are sampled normally distributed

$$\boldsymbol{x}_i \sim \boldsymbol{m} + \sigma \mathcal{N}_i(\mathbf{0}, \mathbf{C}) \qquad \text{for } i = 1, \ldots, \lambda$$

as perturbations of $\boldsymbol{m}$, where $\boldsymbol{x}_i, \boldsymbol{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, $\mathbf{C} \in \mathbb{R}^{n \times n}$

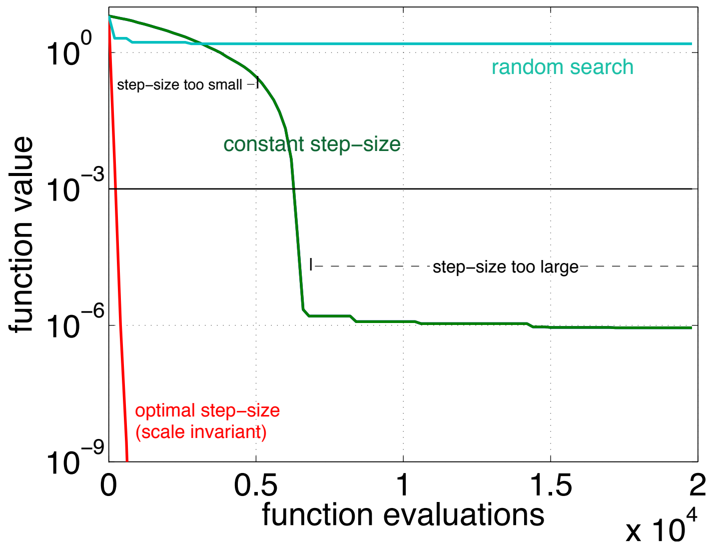


where

- the mean vector $\boldsymbol{m} \in \mathbb{R}^n$ represents the favorite solution and $\boldsymbol{m} \leftarrow \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda}$
- the so-called step-size $\sigma \in \mathbb{R}_+$ controls the *step length*
- the covariance matrix $\mathbf{C} \in \mathbb{R}^{n \times n}$ determines the shape of the distribution ellipsoid

The remaining question is how to update σ and $\mathbf{C}$.

Why Step-Size Control?



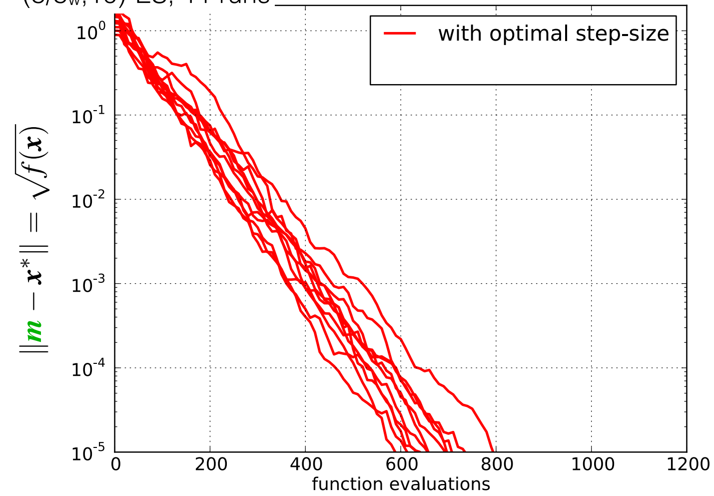
(1+1)-ES
(red & green)

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$$

in $[-2.2, 0.8]^n$
for $n = 10$

Why Step-Size Control?

(5/5$_w$,10)-ES, 11 runs



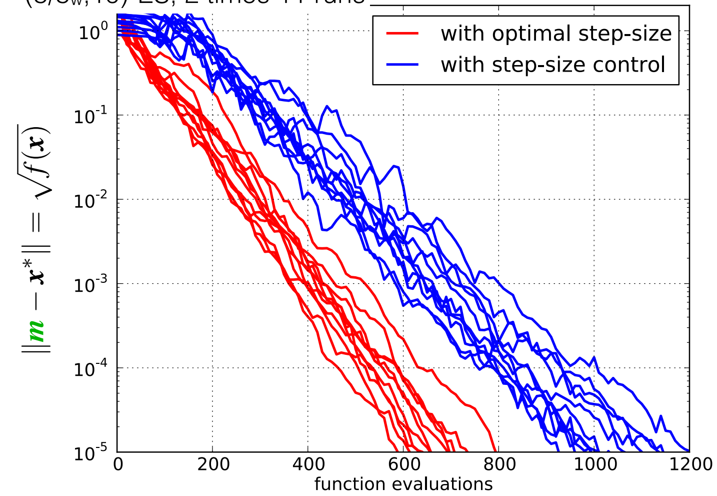
$$f(x) = \sum_{i=1}^{n} x_i^2$$

for $n = 10$ and
$x^0 \in [-0.2, 0.8]^n$

with optimal step-size σ

Why Step-Size Control?

(5/5$_w$,10)-ES, 2 times 11 runs



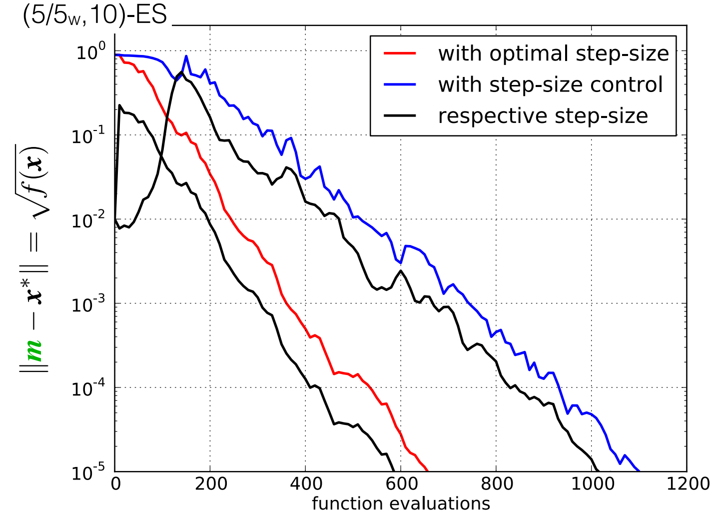
$$f(x) = \sum_{i=1}^{n} x_i^2$$

for $n = 10$ and
$x^0 \in [-0.2, 0.8]^n$

with optimal versus adaptive step-size σ with too small initial σ

Why Step-Size Control?

(5/5$_w$,10)-ES



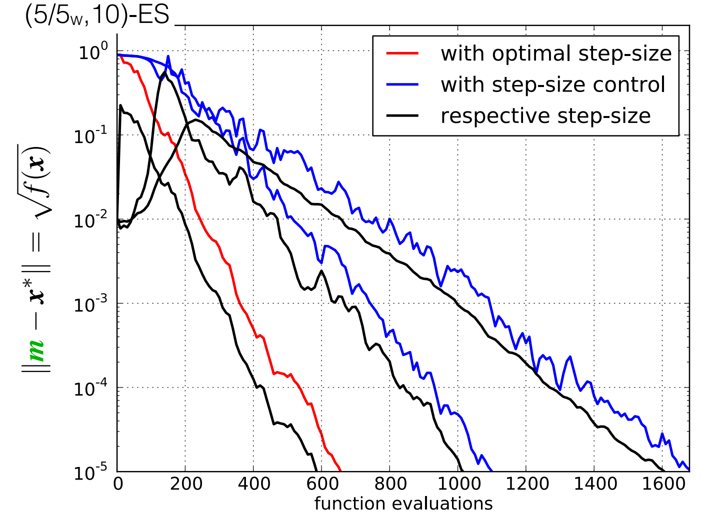
$$f(x) = \sum_{i=1}^{n} x_i^2$$

for $n = 10$ and
$x^0 \in [-0.2, 0.8]^n$

comparing number of f-evals to reach $\|m\| = 10^{-5}$: $\frac{1100-100}{650} \approx 1.5$

Why Step-Size Control?

(5/5$_w$,10)-ES



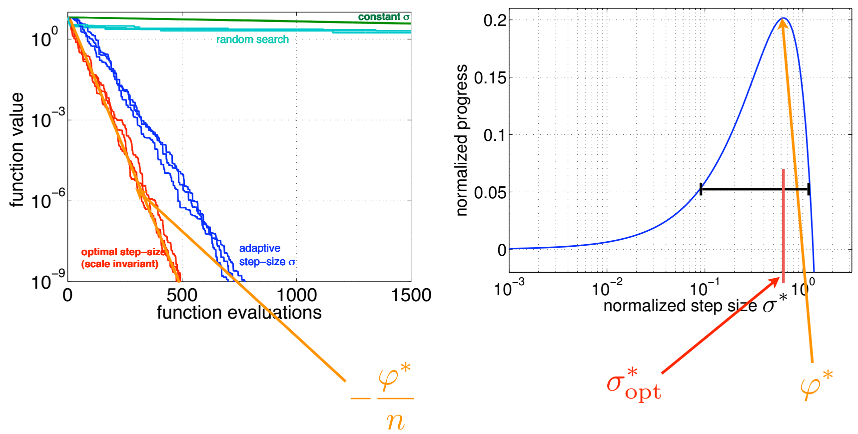
$$f(x) = \sum_{i=1}^{n} x_i^2$$

in $[-0.2, 0.8]^n$
for $n = 10$

comparing optimal versus default damping parameter d_σ: $\frac{1700}{1100} \approx 1.5$

Why Step-Size Control?

$$\sigma_{\text{opt}} = \sigma^*_{\text{opt}} \frac{\|\boldsymbol{m}\|}{n} \approx \mu_w \frac{\|\boldsymbol{m}\|}{n}$$



evolution window refers to the step-size interval (|——|) where reasonable performance is observed

Methods for Step-Size Control

- $1/5$-th success rule[ab], often applied with "+"-selection

  increase step-size if more than 20% of the new solutions are successful, decrease otherwise

- σ-self-adaptation[c], applied with ","-selection

  mutation is applied to the step-size and the better, according to the objective function value, is selected

  simplified "global" self-adaptation

- path length control[d] (Cumulative Step-size Adaptation, CSA)[e]

  self-adaptation derandomized and non-localized

[a] Rechenberg 1973, *Evolutionsstrategie, Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Frommann-Holzboog

[b] Schumer and Steiglitz 1968. Adaptive step size random search. *IEEE TAC*

[c] Schwefel 1981, *Numerical Optimization of Computer Models*, Wiley

[d] Hansen & Ostermeier 2001, Completely Derandomized Self-Adaptation in Evolution Strategies, *Evol. Comput. 9(2)*

[e] Ostermeier *et al* 1994, Step-size adaptation based on non-local use of selection information, *PPSN IV*

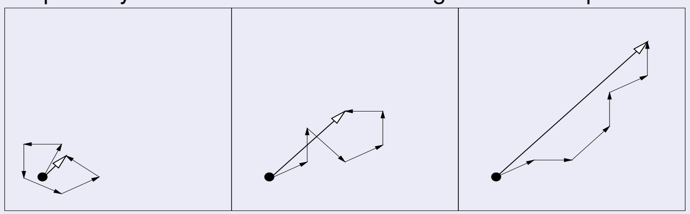
Path Length Control (CSA)

The Concept of Cumulative Step-Size Adaptation

$$\boldsymbol{x}_i = \boldsymbol{m} + \sigma \boldsymbol{y}_i$$
$$\boldsymbol{m} \leftarrow \boldsymbol{m} + \sigma \boldsymbol{y}_w$$

Measure the length of the *evolution path*

the pathway of the mean vector $\boldsymbol{m}$ in the generation sequence



⇓ decrease σ ⇓ increase σ

loosely speaking steps are

- perpendicular under random selection (in expectation)
- perpendicular in the desired situation (to be most efficient)

Path Length Control (CSA)

The Equations

Initialize $\boldsymbol{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, evolution path $\boldsymbol{p}_\sigma = \mathbf{0}$,
set $c_\sigma \approx 4/n$, $d_\sigma \approx 1$.

$\boldsymbol{m} \leftarrow \boldsymbol{m} + \sigma \boldsymbol{y}_w$ where $\boldsymbol{y}_w = \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda}$ update mean

$\boldsymbol{p}_\sigma \leftarrow (1 - c_\sigma)\boldsymbol{p}_\sigma + \underbrace{\sqrt{1-(1-c_\sigma)^2}}_{\text{accounts for } 1-c_\sigma} \underbrace{\sqrt{\mu_w}}_{\text{accounts for } w_i} \boldsymbol{y}_w$

$\sigma \leftarrow \sigma \times \underbrace{\exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma\|}{\mathsf{E}\|\mathcal{N}(\mathbf{0},\mathbf{I})\|} - 1\right)\right)}_{>1 \iff \|\boldsymbol{p}_\sigma\| \text{ is greater than its expectation}}$ update step-size

Path Length Control (CSA)

The Equations

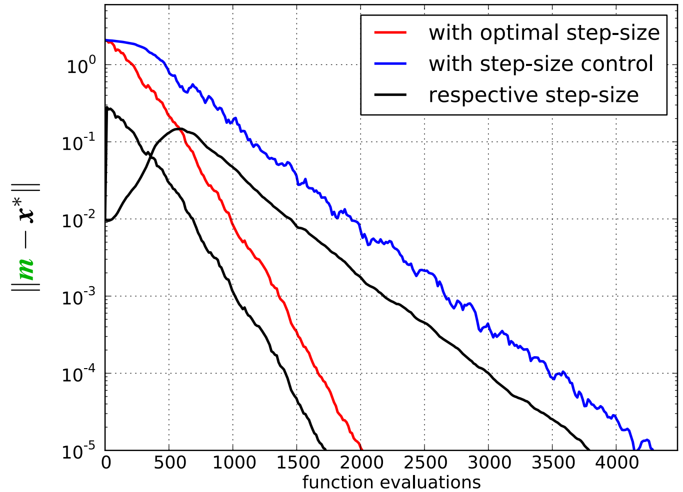
Initialize $\boldsymbol{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, evolution path $\boldsymbol{p}_\sigma = \mathbf{0}$,
set $c_\sigma \approx 4/n$, $d_\sigma \approx 1$.

$$\boldsymbol{m} \leftarrow \boldsymbol{m} + \sigma \boldsymbol{y}_w \quad \text{where } \boldsymbol{y}_w = \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda} \qquad \text{update mean}$$

$$\boldsymbol{p}_\sigma \leftarrow (1 - c_\sigma)\boldsymbol{p}_\sigma + \underbrace{\sqrt{1-(1-c_\sigma)^2}}_{\text{accounts for } 1-c_\sigma} \; \underbrace{\sqrt{\mu_w}}_{\text{accounts for } w_i} \; \boldsymbol{y}_w$$

$$\sigma \leftarrow \sigma \times \underbrace{\exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma\|}{\mathsf{E}\|\mathcal{N}(\mathbf{0},\mathbf{I})\|} - 1\right)\right)}_{>1 \iff \|\boldsymbol{p}_\sigma\| \text{ is greater than its expectation}} \qquad \text{update step-size}$$

$(5/5, 10)$-CSA-ES, default parameters



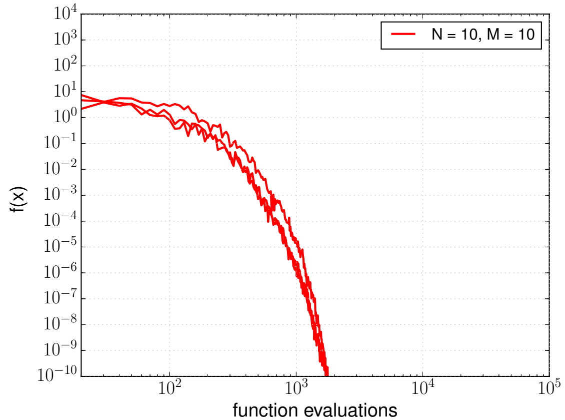
$$f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$$

in $[-0.2, 0.8]^n$
for $n = 30$

Known Issues of CSA I: Ineffective Axes

On a function with ineffective axes

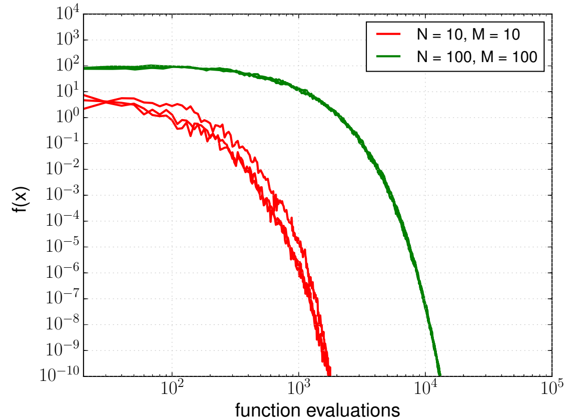
- $f(\boldsymbol{x}) = \sum_{i=1}^{M} [\boldsymbol{x}]_i^2, \quad \boldsymbol{x} \in \mathbb{R}^N, \quad M \leq N.$
- $N - M$ variables do not affect the function value



Known Issues of CSA I: Ineffective Axes

On a function with ineffective axes

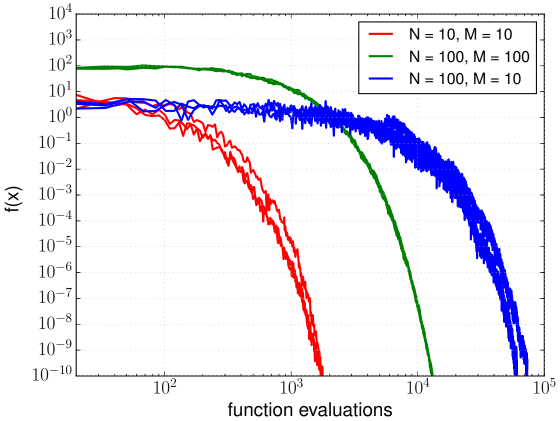
- $f(\boldsymbol{x}) = \sum_{i=1}^{M} [\boldsymbol{x}]_i^2, \quad \boldsymbol{x} \in \mathbb{R}^N, \quad M \leq N.$
- $N - M$ variables do not affect the function value



Known Issues of CSA I: Ineffective Axes

On a function with ineffective axes

- $f(\boldsymbol{x}) = \sum_{i=1}^{M} [\boldsymbol{x}]_i^2, \quad \boldsymbol{x} \in \mathbb{R}^N, \quad M \leq N.$
- $N - M$ variables do not affect the function value



Known Issues of CSA I: Ineffective Axes

On a function with ineffective axes

- $f(\boldsymbol{x}) = \sum_{i=1}^{M} [\boldsymbol{x}]_i^2, \quad \boldsymbol{x} \in \mathbb{R}^N, \quad M \leq N.$
- $N - M$ variables do not affect the function value

If $M \ll N$,

- The $N - M$ components of $\boldsymbol{p}_\sigma$ are normally distributed
- The signal to change the step size is in the M components

$$\boldsymbol{p}_\sigma \leftarrow (1 - c_\sigma)\boldsymbol{p}_\sigma + \sqrt{c_\sigma(2 - c_\sigma)}\sqrt{\mu_w}\sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda}$$

$$\sigma \leftarrow \sigma \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma\|}{\mathsf{E}\|\mathcal{N}(\boldsymbol{0},\boldsymbol{I})\|} - 1\right)\right)$$

Known Issues of CSA II: With a Large Population

With large λ and μ

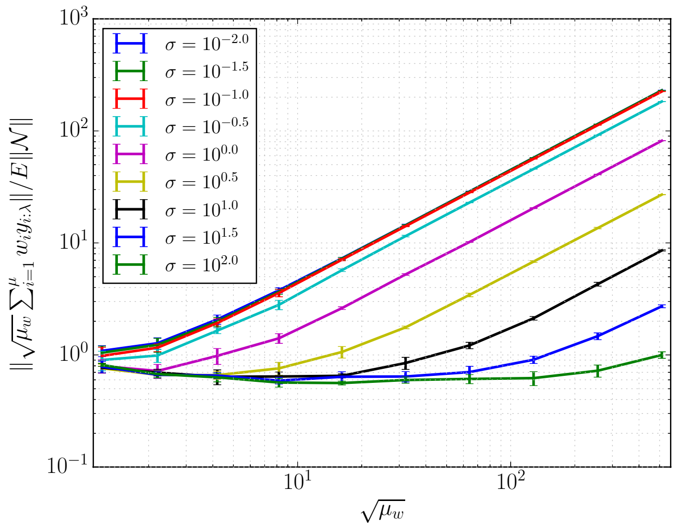
$$\boldsymbol{p}_\sigma \leftarrow (1 - c_\sigma)\boldsymbol{p}_\sigma + \sqrt{c_\sigma(2 - c_\sigma)}\underbrace{\sqrt{\mu_w}}_{\propto\mu\to\infty \text{ as } \lambda,\mu\to\infty}\overbrace{\sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda}}^{\text{converges to a nonzero constant as } \lambda,\mu\to\infty}$$

$$\sigma \leftarrow \sigma \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma\|}{\mathsf{E}\|\mathcal{N}(\boldsymbol{0},\boldsymbol{I})\|} - 1\right)\right)$$

- $\|\boldsymbol{p}_\sigma\| \propto \sqrt{\mu_w} \implies \frac{\|\boldsymbol{p}_\sigma\|}{\mathsf{E}\|\mathcal{N}(\boldsymbol{0},\boldsymbol{I})\|} - 1 \propto \sqrt{\frac{\mu_w}{N}} - 1$ unless $\|\boldsymbol{m} - \boldsymbol{x}^*\| \ll \sigma$

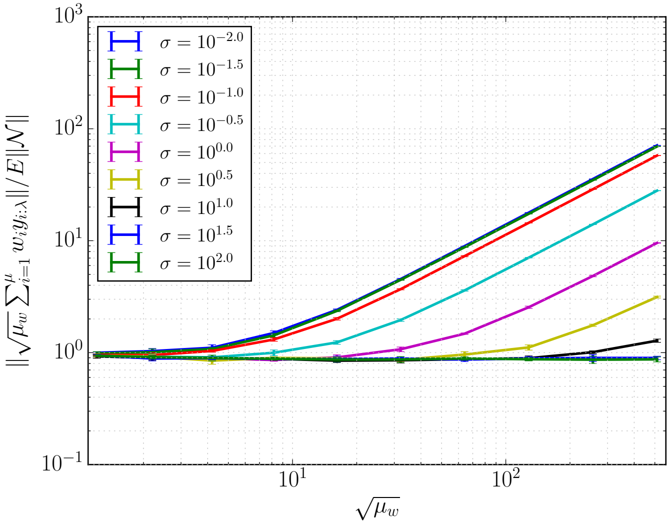
Known Issues of CSA II: With a Large Population

Sphere function $(n = 10,\ \boldsymbol{m} = (1, 0, \dots, 0))$



Known Issues of CSA II: With a Large Population

Sphere function ($n = 100$, $\boldsymbol{m} = (1, 0, \dots, 0)$)



Known Issues of CSA II: With a Large Population

With large λ and μ

$$\boldsymbol{p}_\sigma \leftarrow (1 - c_\sigma)\boldsymbol{p}_\sigma + \underbrace{\sqrt{c_\sigma(2 - c_\sigma)}\sqrt{\mu_w}}_{\propto \mu \to \infty \text{ as } \lambda,\mu\to\infty} \overbrace{\sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda}}^{\text{converges to a nonzero constant as } \lambda,\mu\to\infty}$$

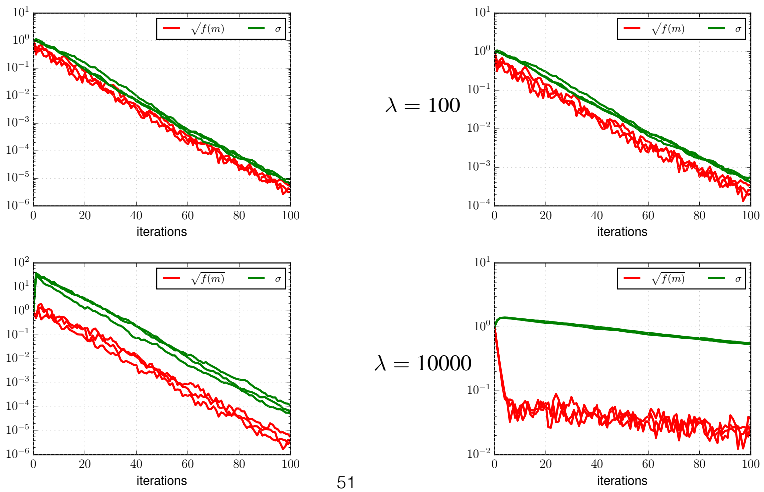
$$\sigma \leftarrow \sigma \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma\|}{\mathsf{E}\|\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})\|} - 1\right)\right)$$

- $\|\boldsymbol{p}_\sigma\| \propto \sqrt{\mu_w} \implies \frac{\|\boldsymbol{p}_\sigma\|}{\mathsf{E}\|\mathcal{N}(\boldsymbol{0},\boldsymbol{I})\|} - 1 \propto \sqrt{\frac{\mu_w}{N}} - 1$ unless $\|\boldsymbol{m} - \boldsymbol{x}^*\| \ll \sigma$
- $d_\sigma \propto \sqrt{\frac{N}{\mu_w}}$ to prevent a increase of σ in one step, however, the convergence will be very slow.

Known Issues of CSA II: With a Large Population

Sphere function ($n = 10$, $\mathbf{m}^{(0)} = (1, 0, \dots, 0)$, $\sigma^{(0)} = 1$)

small damping $d_\sigma = 1 + c_\sigma$ | large damping $d_\sigma \approx \sqrt{\frac{\mu_w}{N}}$



Two-Point Step-Size Adaptation (TPA)

- Sample a pair of symmetric points along the previous mean shift

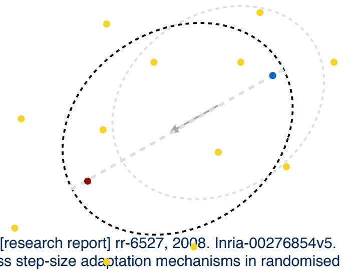
$$\boldsymbol{x}_{1/2} = \boldsymbol{m}^{(g)} \pm \sigma^{(g)} \frac{\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|}{\|\boldsymbol{m}^{(g)} - \boldsymbol{m}^{(g-1)}\|_{\mathbf{C}^{(g)}}} (\boldsymbol{m}^{(g)} - \boldsymbol{m}^{(g-1)}) \qquad \|\boldsymbol{x}\|_{\mathbf{C}} := \boldsymbol{x}^{\mathrm{T}}\mathbf{C}^{-1}\boldsymbol{x}$$

- Compare the ranking of x_1 and x_2 among λ current populations

$$s^{(g+1)} = (1 - c_s)s^{(g)} + c_s \underbrace{\frac{\text{rank}(x_2) - \text{rank}(x_1)}{\lambda - 1}}_{>0 \text{ if the previous step still produces a promising solution}}$$

- Update the step-size

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{s^{(g+1)}}{d_\sigma}\right)$$

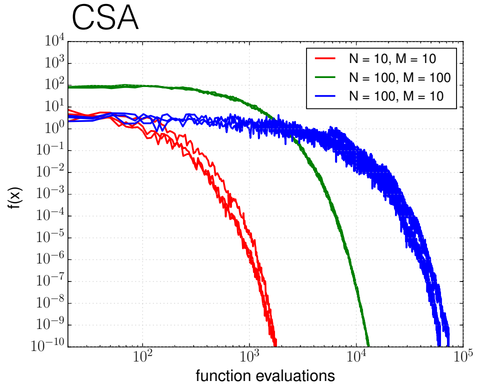


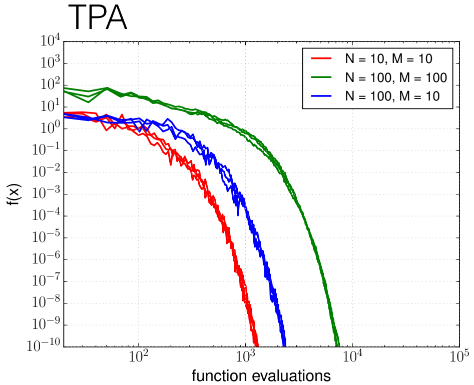
[Hansen, 2008] Hansen, N. (2008). CMA-ES with two-point step-size adaptation. [research report] rr-6527, 2008. Inria-00276854v5.
[Hansen et al., 2014] Hansen, N., Atamna, A., and Auger, A. (2014). How to assess step-size adaptation mechanisms in randomised search. In Parallel Problem Solving from Nature–PPSN XIII, pages 60–69. Springer.

On Sphere with Ineffective Axes

On a function with ineffective axes

- $f(\boldsymbol{x}) = \sum_{i=1}^{M} [\boldsymbol{x}]_i^2, \quad \boldsymbol{x} \in \mathbb{R}^N, \quad M \leq N.$
- $N - M$ variables do not affect the function value





Alternatives: Success-Based Step-Size Control

comparing the fitness distributions of current and previous iterations

Generalizations of $1/5$th-success-rule for non-elitist and multi-recombinant ES

- Median Success Rule [Ait Elhara et al., 2013]
- Population Success Rule [Loshchilov, 2014]

controls a *success probability*

An advantage over CSA and TPA: Cheap Computation

- It depends only on λ.
- cf. CSA and TPA require a computation of $C^{-1/2}\boldsymbol{x}$ and $C^{-1}\boldsymbol{x}$, respectively.

[Ait Elhara et al., 2013] Ait Elhara, O., Auger, A., and Hansen, N. (2013). A median success rule for non- elitist evolution strategies: Study of feasibility. In Proc. of the GECCO, pages 415–422.
[Loshchilov, 2014] Loshchilov, I. (2014). A computationally efficient limited memory cma-es for large scale optimization. In Proc. of the GECCO, pages 397–404.

Alternatives: Success-Based Step-Size Control

comparing the fitness distributions of current and previous iterations

Generalizations of $1/5$th-success-rule for non-elitist and multi-recombinant ES

- Median Success Rule [Ait Elhara et al., 2013]
- Population Success Rule [Loshchilov, 2014]

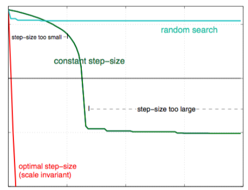
controls a *success probability*

An advantage over CSA and TPA: Cheap Computation

- It depends only on λ.
- cf. CSA and TPA require a computation of $C^{-1/2}\boldsymbol{x}$ and $C^{-1}\boldsymbol{x}$, respectively.

[Ait Elhara et al., 2013] Ait Elhara, O., Auger, A., and Hansen, N. (2013). A median success rule for non- elitist evolution strategies: Study of feasibility. In Proc. of the GECCO, pages 415–422.
[Loshchilov, 2014] Loshchilov, I. (2014). A computationally efficient limited memory cma-es for large scale optimization. In Proc. of the GECCO, pages 397–404.

Step-Size Control: Summary



Why Step-Size Control?

- to achieve linear convergence

Cumulative Step-Size Adaptation

- efficient and robust for $\lambda \leq N$
- inefficient (1) $\lambda \gg N$, (2) function with ineffective axes

Alternative Step-Size Adaptation Mechanisms

- Two-Point Step-Size Adaptation
- Median Success Rule, Population Success Rule

the effective adaptation of the overall population diversity seems yet to pose open questions, in particular with recombination or without entire control over the realized distribution.[a]

[a] Hansen et al. How to Assess Step-Size Adaptation Mechanisms in Randomised Search. PPSN 2014

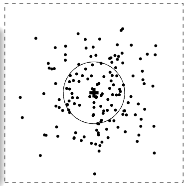
Evolution Strategies

Recalling

New search points are sampled normally distributed

$$x_i \sim m + \sigma \mathcal{N}_i(\mathbf{0}, \mathbf{C}) \qquad \text{for } i = 1, \ldots, \lambda$$

as perturbations of m, where $x_i, m \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, $\mathbf{C} \in \mathbb{R}^{n \times n}$



where

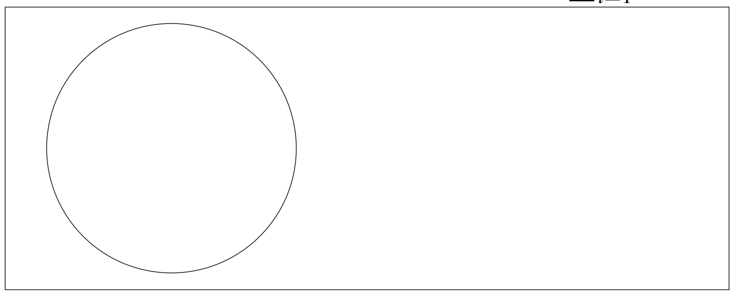
- the mean vector $m \in \mathbb{R}^n$ represents the favorite solution
- the so-called step-size $\sigma \in \mathbb{R}_+$ controls the *step length*
- the covariance matrix $\mathbf{C} \in \mathbb{R}^{n \times n}$ determines the shape of the distribution ellipsoid

The remaining question is how to update $\mathbf{C}$.

Covariance Matrix Adaptation

Rank-One Update

$$m \leftarrow m + \sigma y_w, \quad y_w = \sum_{i=1}^{\mu} w_i y_{i:\lambda}, \quad y_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$$



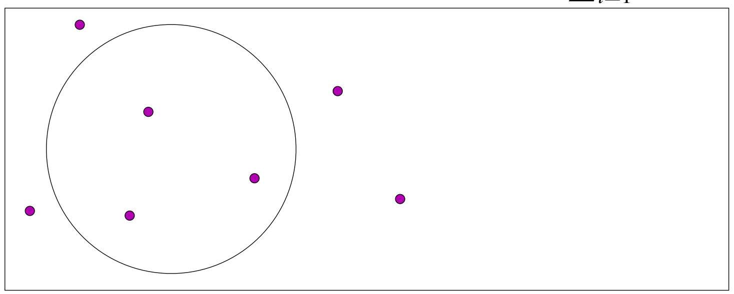
initial distribution, $\mathbf{C} = \mathbf{I}$

…equations

Covariance Matrix Adaptation

Rank-One Update

$$m \leftarrow m + \sigma y_w, \quad y_w = \sum_{i=1}^{\mu} w_i y_{i:\lambda}, \quad y_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$$



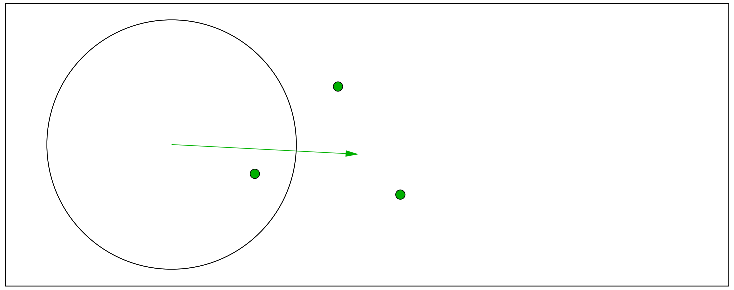
initial distribution, $\mathbf{C} = \mathbf{I}$

…equations

Covariance Matrix Adaptation

Rank-One Update

$$\boldsymbol{m} \leftarrow \boldsymbol{m} + \sigma \boldsymbol{y}_w, \quad \boldsymbol{y}_w = \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda}, \quad \boldsymbol{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$$



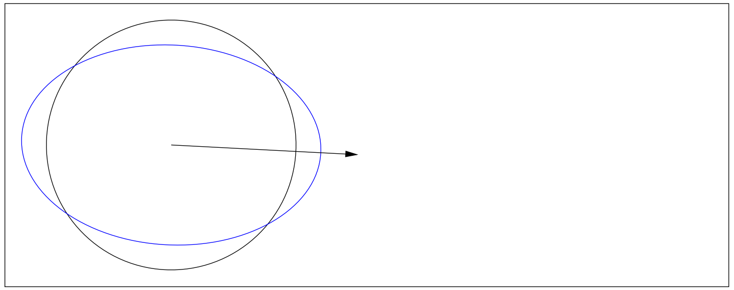
$\boldsymbol{y}_w$, movement of the population mean $\boldsymbol{m}$ (disregarding σ)

. . . equations

Covariance Matrix Adaptation

Rank-One Update

$$\boldsymbol{m} \leftarrow \boldsymbol{m} + \sigma \boldsymbol{y}_w, \quad \boldsymbol{y}_w = \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda}, \quad \boldsymbol{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$$



mixture of distribution $\mathbf{C}$ and step $\boldsymbol{y}_w$,

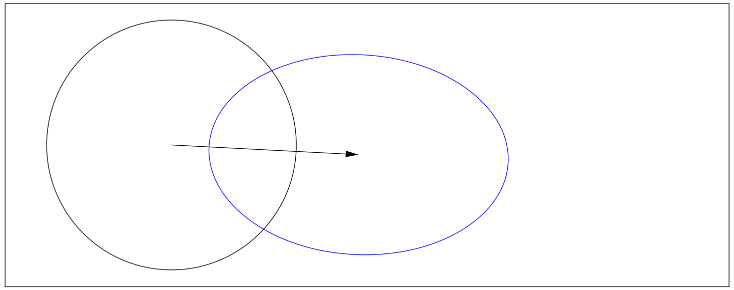
$\mathbf{C} \leftarrow 0.8 \times \mathbf{C} + 0.2 \times \boldsymbol{y}_w \boldsymbol{y}_w^{\mathrm{T}}$

. . . equations

Covariance Matrix Adaptation

Rank-One Update

$$\boldsymbol{m} \leftarrow \boldsymbol{m} + \sigma \boldsymbol{y}_w, \quad \boldsymbol{y}_w = \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda}, \quad \boldsymbol{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$$



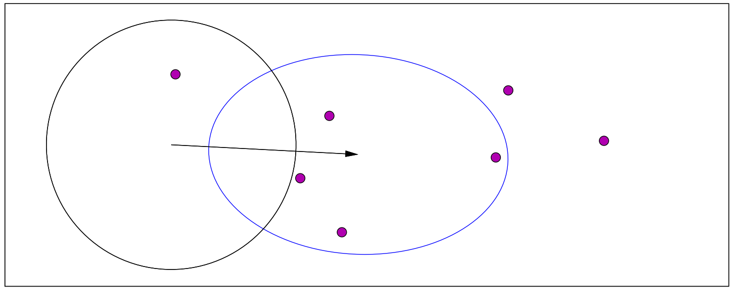
new distribution (disregarding σ)

. . . equations

Covariance Matrix Adaptation

Rank-One Update

$$\boldsymbol{m} \leftarrow \boldsymbol{m} + \sigma \boldsymbol{y}_w, \quad \boldsymbol{y}_w = \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda}, \quad \boldsymbol{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$$



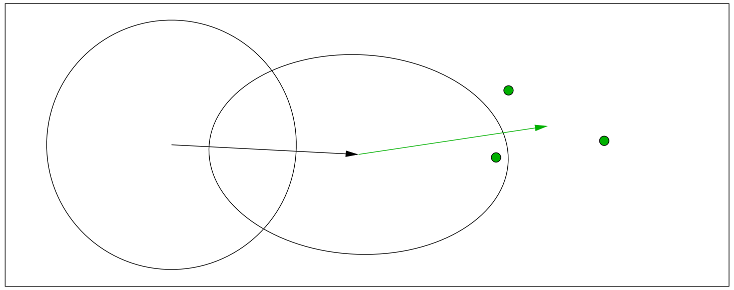
new distribution (disregarding σ)

. . . equations

Covariance Matrix Adaptation

Rank-One Update

$$\boldsymbol{m} \leftarrow \boldsymbol{m} + \sigma \boldsymbol{y}_w, \quad \boldsymbol{y}_w = \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda}, \quad \boldsymbol{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$$



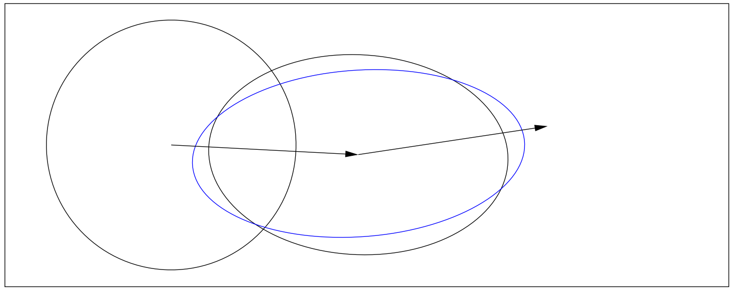
movement of the population mean $\boldsymbol{m}$

. . . equations

Covariance Matrix Adaptation

Rank-One Update

$$\boldsymbol{m} \leftarrow \boldsymbol{m} + \sigma \boldsymbol{y}_w, \quad \boldsymbol{y}_w = \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda}, \quad \boldsymbol{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$$



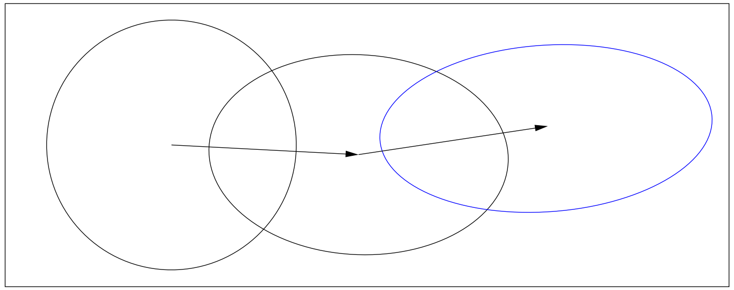
mixture of distribution $\mathbf{C}$ and step $\boldsymbol{y}_w$,
$\mathbf{C} \leftarrow 0.8 \times \mathbf{C} + 0.2 \times \boldsymbol{y}_w \boldsymbol{y}_w^{\mathrm{T}}$

. . . equations

Covariance Matrix Adaptation

Rank-One Update

$$\boldsymbol{m} \leftarrow \boldsymbol{m} + \sigma \boldsymbol{y}_w, \quad \boldsymbol{y}_w = \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda}, \quad \boldsymbol{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C})$$



new distribution,
$\mathbf{C} \leftarrow 0.8 \times \mathbf{C} + 0.2 \times \boldsymbol{y}_w \boldsymbol{y}_w^{\mathrm{T}}$
the ruling principle: the adaptation increases the likelihood of successful steps, $\boldsymbol{y}_w$, to appear again
another viewpoint: the adaptation follows a natural gradient approximation of the expected fitness

. . . equations

Covariance Matrix Adaptation

Rank-One Update

Initialize $\boldsymbol{m} \in \mathbb{R}^n$, and $\mathbf{C} = \mathbf{I}$, set $\sigma = 1$, learning rate $c_{\mathrm{cov}} \approx 2/n^2$
While not terminate

$$\boldsymbol{x}_i = \boldsymbol{m} + \sigma \boldsymbol{y}_i, \qquad \boldsymbol{y}_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C}),$$

$$\boldsymbol{m} \leftarrow \boldsymbol{m} + \sigma \boldsymbol{y}_w \qquad \text{where } \boldsymbol{y}_w = \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda}$$

$$\mathbf{C} \leftarrow (1 - c_{\mathrm{cov}})\mathbf{C} + c_{\mathrm{cov}} \mu_w \underbrace{\boldsymbol{y}_w \boldsymbol{y}_w^{\mathrm{T}}}_{\text{rank-one}} \qquad \text{where } \mu_w = \frac{1}{\sum_{i=1}^{\mu} {w_i}^2} \geq 1$$

The rank-one update has been found independently in several domains[6] [7] [8] [9]

[6] Kjellström&Taxén 1981. Stochastic Optimization in System Design, IEEE TCS

[7] Hansen&Ostermeier 1996. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation, ICEC

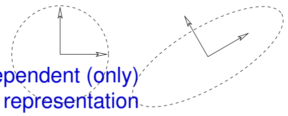
[8] Ljung 1999. System Identification: Theory for the User

[9] Haario et al 2001. An adaptive Metropolis algorithm, JSTOR

$$\mathbf{C} \leftarrow (1 - c_{\text{cov}})\mathbf{C} + c_{\text{cov}}\mu_w \boldsymbol{y}_w \boldsymbol{y}_w^{\mathrm{T}}$$

covariance matrix adaptation

- learns all pairwise dependencies between variables
  off-diagonal entries in the covariance matrix reflect the dependencies
- conducts a principle component analysis (PCA) of steps $\boldsymbol{y}_w$, sequentially in time and space
  eigenvectors of the covariance matrix $\mathbf{C}$ are the principle components / the principle axes of the mutation ellipsoid
- learns a new rotated problem representation
  components are independent (only) in the new representation



- learns a new (Mahalanobis) metric
  variable metric method
- approximates the inverse Hessian on quadratic functions
  transformation into the sphere function
- for $\mu = 1$: conducts a natural gradient ascent on the distribution $\mathcal{N}$
  entirely independent of the given coordinate system

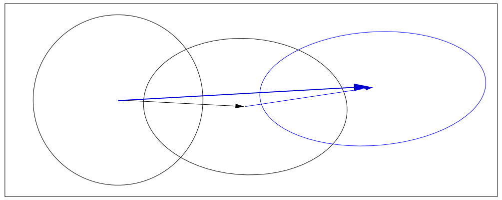
. . . cumulation-rank-μ

Cumulation

The Evolution Path

Evolution Path

Conceptually, the evolution path is the search path the strategy takes over a number of generation steps. It can be expressed as a sum of consecutive *steps* of the mean $\boldsymbol{m}$.



An exponentially weighted sum of steps $\boldsymbol{y}_w$ is used

$$\boldsymbol{p}_c \propto \sum_{i=0}^{g} \underbrace{(1 - c_c)^{g-i}}_{\text{exponentially fading weights}} \boldsymbol{y}_w^{(i)}$$

The recursive construction of the evolution path (cumulation):

$$\boldsymbol{p}_c \leftarrow \underbrace{(1 - c_c)}_{\text{decay factor}} \boldsymbol{p}_c + \underbrace{\sqrt{1 - (1 - c_c)^2}\sqrt{\mu_w}}_{\text{normalization factor}} \underbrace{\boldsymbol{y}_w}_{\text{input} = \frac{\boldsymbol{m} - \boldsymbol{m}_{\text{old}}}{\sigma}}$$

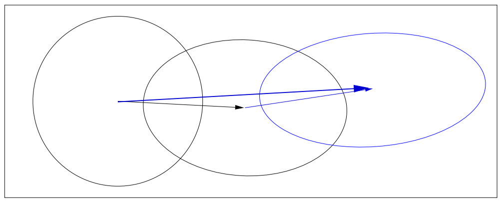
where $\mu_w = \frac{1}{\sum w_i^2}$, $c_c \ll 1$. History information is accumulated in the evolution path.

Cumulation

The Evolution Path

Evolution Path

Conceptually, the evolution path is the search path the strategy takes over a number of generation steps. It can be expressed as a sum of consecutive *steps* of the mean $\boldsymbol{m}$.



An exponentially weighted sum of steps $\boldsymbol{y}_w$ is used

$$\boldsymbol{p}_c \propto \sum_{i=0}^{g} \underbrace{(1 - c_c)^{g-i}}_{\text{exponentially fading weights}} \boldsymbol{y}_w^{(i)}$$

The recursive construction of the evolution path (cumulation):

$$\boldsymbol{p}_c \leftarrow \underbrace{(1 - c_c)}_{\text{decay factor}} \boldsymbol{p}_c + \underbrace{\sqrt{1 - (1 - c_c)^2}\sqrt{\mu_w}}_{\text{normalization factor}} \underbrace{\boldsymbol{y}_w}_{\text{input} = \frac{\boldsymbol{m} - \boldsymbol{m}_{\text{old}}}{\sigma}}$$

where $\mu_w = \frac{1}{\sum w_i^2}$, $c_c \ll 1$. History information is accumulated in the evolution path.

“Cumulation” is a widely used technique and also know as

- *exponential smoothing* in time series, forecasting
- exponentially weighted *mooving average*
- *iterate averaging* in stochastic approximation
- *momentum* in the back-propagation algorithm for ANNs
- . . .

“Cumulation” conducts a *low-pass* filtering, but there is more to it. . .

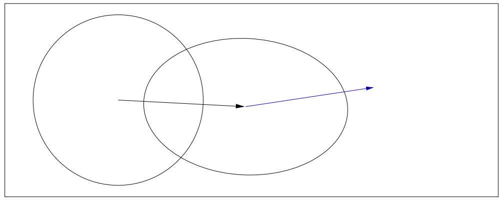
. . . why?

Cumulation

$\mathbf{C} \leftarrow (1 - c_{\mathrm{cov}})\mathbf{C} + c_{\mathrm{cov}}\mu_w \mathbf{y}_w \mathbf{y}_w^{\mathrm{T}}$

Utilizing the Evolution Path

We used $\mathbf{y}_w \mathbf{y}_w^{\mathrm{T}}$ for updating $\mathbf{C}$. Because $\mathbf{y}_w \mathbf{y}_w^{\mathrm{T}} = -\mathbf{y}_w(-\mathbf{y}_w)^{\mathrm{T}}$ the sign of $\mathbf{y}_w$ is lost.



The sign information (signifying correlation *between* steps) is (re-)introduced by using the *evolution path*.

$$\mathbf{p}_c \leftarrow \underbrace{(1-c_c)}_{\text{decay factor}}\mathbf{p}_c + \underbrace{\sqrt{1-(1-c_c)^2}\sqrt{\mu_w}}_{\text{normalization factor}}\mathbf{y}_w$$

$$\mathbf{C} \leftarrow (1-c_{\mathrm{cov}})\mathbf{C} + c_{\mathrm{cov}}\underbrace{\mathbf{p}_c\mathbf{p}_c^{\mathrm{T}}}_{\text{rank-one}}$$

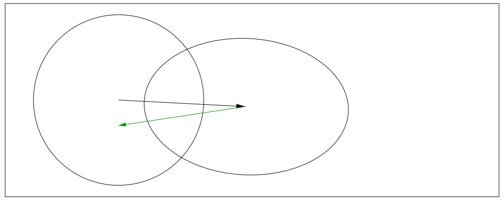
where $\mu_w = \frac{1}{\sum w_i^2}$, $c_{\mathrm{cov}} \ll c_c \ll 1$ such that $1/c_c$ is the "backward time horizon".

Cumulation

$\mathbf{C} \leftarrow (1 - c_{\mathrm{cov}})\mathbf{C} + c_{\mathrm{cov}}\mu_w \mathbf{y}_w \mathbf{y}_w^{\mathrm{T}}$

Utilizing the Evolution Path

We used $\mathbf{y}_w \mathbf{y}_w^{\mathrm{T}}$ for updating $\mathbf{C}$. Because $\mathbf{y}_w \mathbf{y}_w^{\mathrm{T}} = -\mathbf{y}_w(-\mathbf{y}_w)^{\mathrm{T}}$ the sign of $\mathbf{y}_w$ is lost.



The sign information (signifying correlation *between* steps) is (re-)introduced by using the *evolution path*.

$$\mathbf{p}_c \leftarrow \underbrace{(1-c_c)}_{\text{decay factor}}\mathbf{p}_c + \underbrace{\sqrt{1-(1-c_c)^2}\sqrt{\mu_w}}_{\text{normalization factor}}\mathbf{y}_w$$

$$\mathbf{C} \leftarrow (1-c_{\mathrm{cov}})\mathbf{C} + c_{\mathrm{cov}}\underbrace{\mathbf{p}_c\mathbf{p}_c^{\mathrm{T}}}_{\text{rank-one}}$$

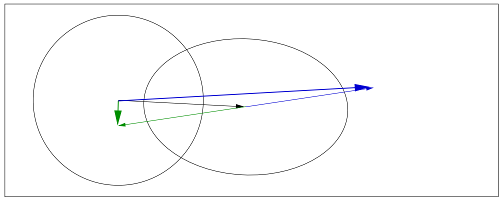
where $\mu_w = \frac{1}{\sum w_i^2}$, $c_{\mathrm{cov}} \ll c_c \ll 1$ such that $1/c_c$ is the "backward time horizon".

Cumulation

$\mathbf{C} \leftarrow (1 - c_{\mathrm{cov}})\mathbf{C} + c_{\mathrm{cov}}\mu_w \mathbf{y}_w \mathbf{y}_w^{\mathrm{T}}$

Utilizing the Evolution Path

We used $\mathbf{y}_w \mathbf{y}_w^{\mathrm{T}}$ for updating $\mathbf{C}$. Because $\mathbf{y}_w \mathbf{y}_w^{\mathrm{T}} = -\mathbf{y}_w(-\mathbf{y}_w)^{\mathrm{T}}$ the sign of $\mathbf{y}_w$ is lost.



The sign information (signifying correlation *between* steps) is (re-)introduced by using the *evolution path*.

$$\mathbf{p}_c \leftarrow \underbrace{(1-c_c)}_{\text{decay factor}}\mathbf{p}_c + \underbrace{\sqrt{1-(1-c_c)^2}\sqrt{\mu_w}}_{\text{normalization factor}}\mathbf{y}_w$$

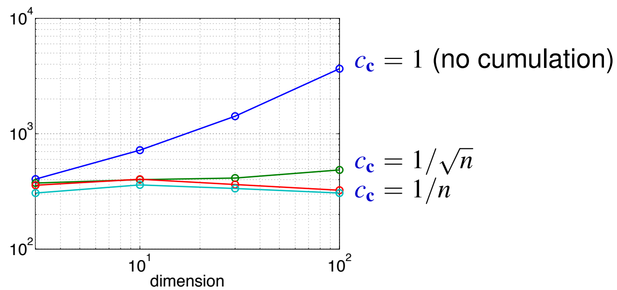
$$\mathbf{C} \leftarrow (1-c_{\mathrm{cov}})\mathbf{C} + c_{\mathrm{cov}}\underbrace{\mathbf{p}_c\mathbf{p}_c^{\mathrm{T}}}_{\text{rank-one}}$$

where $\mu_w = \frac{1}{\sum w_i^2}$, $c_{\mathrm{cov}} \ll c_c \ll 1$ such that $1/c_c$ is the "backward time horizon".

Using an evolution path for the rank-one update of the covariance matrix reduces the number of function evaluations to adapt to a straight ridge from about $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$.[a]

[a] Hansen & Auger 2013. Principled design of continuous stochastic search: From theory to practice.

Number of f-evaluations divided by dimension on the cigar function $f(\mathbf{x}) = x_1^2 + 10^6 \sum_{i=2}^{n} x_i^2$



The overall model complexity is n^2 but important parts of the model can be learned in time of order n

Rank-μ Update

$$\begin{aligned} \boldsymbol{x}_i &= \boldsymbol{m} + \sigma \boldsymbol{y}_i, & \boldsymbol{y}_i &\sim \mathcal{N}_i(\mathbf{0}, \mathbf{C}), \\ \boldsymbol{m} &\leftarrow \boldsymbol{m} + \sigma \boldsymbol{y}_w & \boldsymbol{y}_w &= \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda} \end{aligned}$$

The rank-μ update extends the update rule for large population sizes λ using $\mu > 1$ vectors to update $\mathbf{C}$ at each generation step.
The weighted empirical covariance matrix

$$\mathbf{C}_\mu = \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda} \boldsymbol{y}_{i:\lambda}^{\mathrm{T}}$$

computes a weighted mean of the outer products of the best μ steps and has rank $\min(\mu, n)$ with probability one.

with $\mu = \lambda$ weights can be negative [10]

The rank-μ update then reads

$$\mathbf{C} \leftarrow (1 - c_{\mathrm{cov}})\,\mathbf{C} + c_{\mathrm{cov}}\,\mathbf{C}_\mu$$

where $c_{\mathrm{cov}} \approx \mu_w / n^2$ and $c_{\mathrm{cov}} \leq 1$.

[10] Jastrebski and Arnold (2006). Improving evolution strategies through active covariance matrix adaptation. CEC.

Rank-μ Update

$$\begin{aligned} \boldsymbol{x}_i &= \boldsymbol{m} + \sigma \boldsymbol{y}_i, & \boldsymbol{y}_i &\sim \mathcal{N}_i(\mathbf{0}, \mathbf{C}), \\ \boldsymbol{m} &\leftarrow \boldsymbol{m} + \sigma \boldsymbol{y}_w & \boldsymbol{y}_w &= \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda} \end{aligned}$$

The rank-μ update extends the update rule for large population sizes λ using $\mu > 1$ vectors to update $\mathbf{C}$ at each generation step.
The weighted empirical covariance matrix

$$\mathbf{C}_\mu = \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda} \boldsymbol{y}_{i:\lambda}^{\mathrm{T}}$$

computes a weighted mean of the outer products of the best μ steps and has rank $\min(\mu, n)$ with probability one.

with $\mu = \lambda$ weights can be negative [10]

The rank-μ update then reads

$$\mathbf{C} \leftarrow (1 - c_{\mathrm{cov}})\,\mathbf{C} + c_{\mathrm{cov}}\,\mathbf{C}_\mu$$

where $c_{\mathrm{cov}} \approx \mu_w / n^2$ and $c_{\mathrm{cov}} \leq 1$.

[10] Jastrebski and Arnold (2006). Improving evolution strategies through active covariance matrix adaptation. CEC.

Rank-μ Update

$$\begin{aligned} \boldsymbol{x}_i &= \boldsymbol{m} + \sigma \boldsymbol{y}_i, & \boldsymbol{y}_i &\sim \mathcal{N}_i(\mathbf{0}, \mathbf{C}), \\ \boldsymbol{m} &\leftarrow \boldsymbol{m} + \sigma \boldsymbol{y}_w & \boldsymbol{y}_w &= \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda} \end{aligned}$$

The rank-μ update extends the update rule for large population sizes λ using $\mu > 1$ vectors to update $\mathbf{C}$ at each generation step.
The weighted empirical covariance matrix

$$\mathbf{C}_\mu = \sum_{i=1}^{\mu} w_i \boldsymbol{y}_{i:\lambda} \boldsymbol{y}_{i:\lambda}^{\mathrm{T}}$$

computes a weighted mean of the outer products of the best μ steps and has rank $\min(\mu, n)$ with probability one.

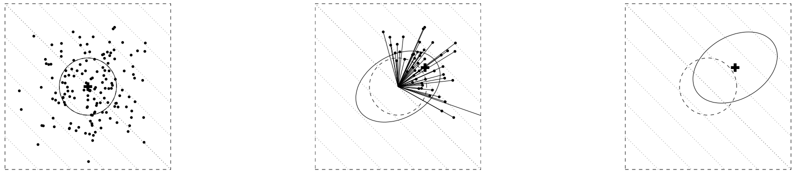
with $\mu = \lambda$ weights can be negative [10]

The rank-μ update then reads

$$\mathbf{C} \leftarrow (1 - c_{\mathrm{cov}})\,\mathbf{C} + c_{\mathrm{cov}}\,\mathbf{C}_\mu$$

where $c_{\mathrm{cov}} \approx \mu_w / n^2$ and $c_{\mathrm{cov}} \leq 1$.

[10] Jastrebski and Arnold (2006). Improving evolution strategies through active covariance matrix adaptation. CEC.



$x_i = m + \sigma y_i, \quad y_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$

$\mathbf{C}_\mu = \frac{1}{\mu}\sum y_{i:\lambda} y_{i:\lambda}^{\mathrm{T}}$

$\mathbf{C} \leftarrow (1-1) \times \mathbf{C} + 1 \times \mathbf{C}_\mu$

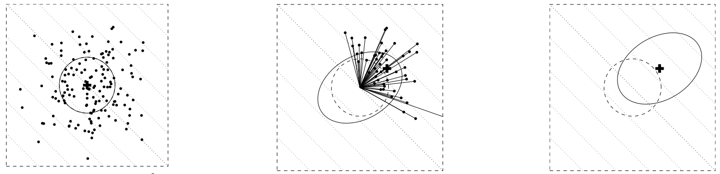
$m_{\text{new}} \leftarrow m + \frac{1}{\mu}\sum y_{i:\lambda}$

sampling of $\lambda = 150$ solutions where $\mathbf{C} = \mathbf{I}$ and $\sigma = 1$

calculating $\mathbf{C}$ where $\mu = 50$, $w_1 = \cdots = w_\mu = \frac{1}{\mu}$, and $c_{\text{cov}} = 1$

new distribution

Rank-μ CMA versus Estimation of Multivariate Normal Algorithm $\text{EMNA}_{\text{global}}$ [11]

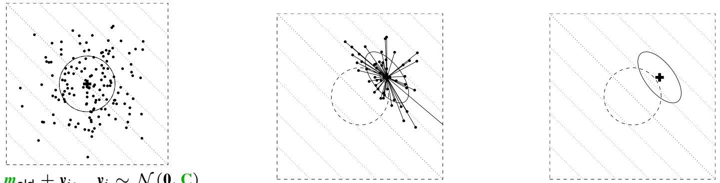


$x_i = m_{\text{old}} + y_i, \quad y_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$

$\mathbf{C} \leftarrow \frac{1}{\mu}\sum (x_{i:\lambda} - m_{\text{old}})(x_{i:\lambda} - m_{\text{old}})^{\mathrm{T}}$

$m_{\text{new}} = m_{\text{old}} + \frac{1}{\mu}\sum y_{i:\lambda}$

rank-μ CMA conducts a PCA of steps



$x_i = m_{\text{old}} + y_i, \quad y_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$

$\mathbf{C} \leftarrow \frac{1}{\mu}\sum (x_{i:\lambda} - m_{\text{new}})(x_{i:\lambda} - m_{\text{new}})^{\mathrm{T}}$

$m_{\text{new}} = m_{\text{old}} + \frac{1}{\mu}\sum y_{i:\lambda}$

$\text{EMNA}_{\text{global}}$ conducts a PCA of points

sampling of $\lambda = 150$ solutions (dots)

calculating $\mathbf{C}$ from $\mu = 50$ solutions

new distribution

m_{new} is the minimizer for the variances when calculating $\mathbf{C}$

[11] Hansen, N. (2006). The CMA Evolution Strategy: A Comparing Review. In J.A. Lozano, P. Larranga, I. Inza and E. Bengoetxea (Eds.). Towards a new evolutionary computation. Advances in estimation of distribution algorithms. pp. 75-102

The rank-μ update

- increases the possible learning rate in large populations
  roughly from $2/n^2$ to μ_w/n^2
- can reduce the number of necessary generations roughly from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ (12)
  given $\mu_w \propto \lambda \propto n$

Therefore the rank-μ update is the primary mechanism whenever a large population size is used
say $\lambda \geq 3n + 10$

The rank-one update

- uses the evolution path and reduces the number of necessary function evaluations to learn straight ridges from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$.

Rank-one update and rank-μ update can be combined

...all equations

[12] Hansen, Müller, and Koumoutsakos 2003. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation, 11(1)*, pp. 1-18

The rank-μ update

- increases the possible learning rate in large populations
  roughly from $2/n^2$ to μ_w/n^2
- can reduce the number of necessary generations roughly from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ (12)
  given $\mu_w \propto \lambda \propto n$

Therefore the rank-μ update is the primary mechanism whenever a large population size is used
say $\lambda \geq 3n + 10$

The rank-one update

- uses the evolution path and reduces the number of necessary function evaluations to learn straight ridges from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$.

Rank-one update and rank-μ update can be combined

...all equations

[12] Hansen, Müller, and Koumoutsakos 2003. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation, 11(1)*, pp. 1-18

The rank-μ update

- increases the possible learning rate in large populations
  roughly from $2/n^2$ to μ_w/n^2
- can reduce the number of necessary generations roughly from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ (12)
  given $\mu_w \propto \lambda \propto n$

Therefore the rank-μ update is the primary mechanism whenever a large population size is used
say $\lambda \geq 3n + 10$

The rank-one update

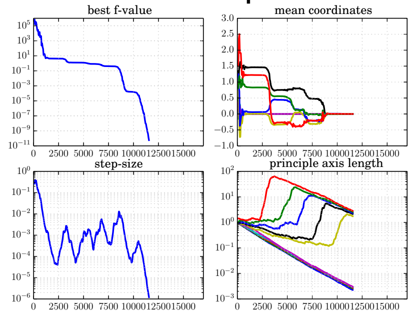
- uses the evolution path and reduces the number of necessary function evaluations to learn straight ridges from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$.

Rank-one update and rank-μ update can be combined

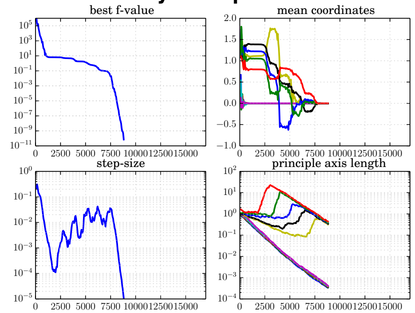
...all equations

[12] Hansen, Müller, and Koumoutsakos 2003. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation, 11(1)*, pp. 1-18

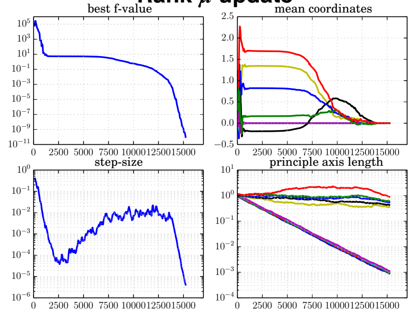
Rank-one update



Hybrid update



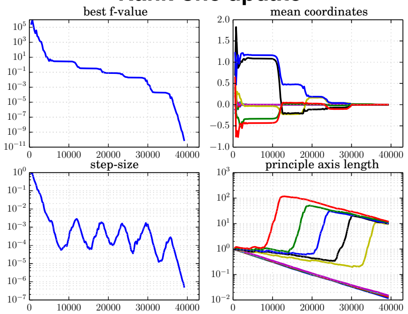
Rank-μ update



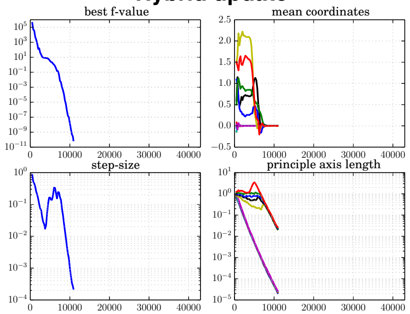
$$f_{\text{TwoAxes}}(x) = \sum_{i=1}^{5} x_i^2 + 10^6 \sum_{i=6}^{10} x_i^2$$

$\lambda = 10$ (default for $N = 10$)

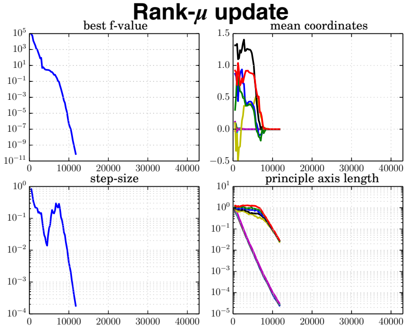
Rank-one update



Hybrid update



Rank-μ update



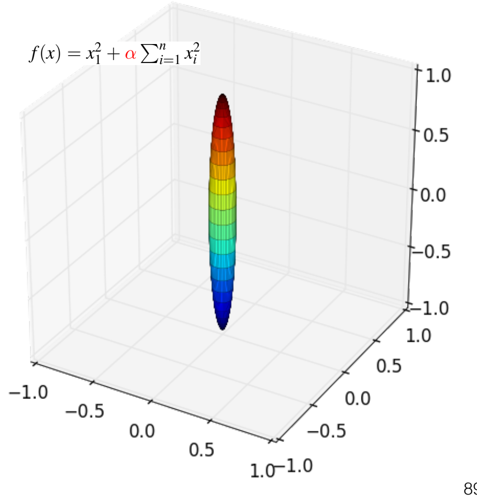
$$f_{\text{TwoAxes}}(x) = \sum_{i=1}^{5} x_i^2 + 10^6 \sum_{i=6}^{10} x_i^2$$

$\lambda = 50$

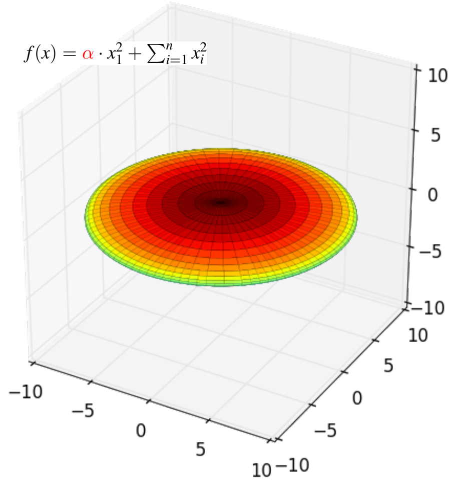
Different Types of Ill-Conditioning

(α: Axes Ratio = 10)

Cigar Type:
1 long axis = n-1 short axes



Discus Type:
1 short axis = n-1 long axes



Learning a Short/Long Axis

Rank-one and Rank-μ Covariance Matrix Update

$$C \leftarrow (1 - c_1 - c_\mu)C + \underbrace{c_1 p_c p_c^T + c_\mu \sum_{i=1}^{\mu} w_i y_{i:\lambda} y_{i:\lambda}^T}_{\text{nonnegative definite, i.e., increase covariances}}$$

- increases the variance in the directions of p_c and $y_{i:\lambda}$; cumulation is very effective for this purpose
- decrease the variance only by multiplying the factor $(1 - c_1 - c_\mu)$

Learning a Short/Long Axis

Rank-one and Rank-μ Covariance Matrix Update

$$C \leftarrow (1 - c_1 - c_\mu)C + \underbrace{c_1 p_c p_c^T + c_\mu \sum_{i=1}^{\mu} w_i y_{i:\lambda} y_{i:\lambda}^T}_{\text{nonnegative definite, i.e., increase covariances}}$$

- increases the variance in the directions of p_c and $y_{i:\lambda}$; cumulation is very effective for this purpose
- decrease the variance only by multiplying the factor $(1 - c_1 - c_\mu)$

Learning a Short/Long Axis

Rank-one and Rank-μ Covariance Matrix Update

$$C \leftarrow (1 - c_1 - c_\mu)C + \underbrace{c_1 p_c p_c^T + c_\mu \sum_{i=1}^{\mu} w_i y_{i:\lambda} y_{i:\lambda}^T}_{\text{nonnegative definite, i.e., increase covariances}}$$

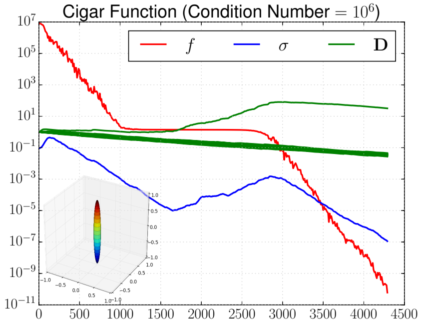
- increases the variance in the directions of p_c and $y_{i:\lambda}$; cumulation is very effective for this purpose
- decrease the variance only by multiplying the factor $(1 - c_1 - c_\mu)$

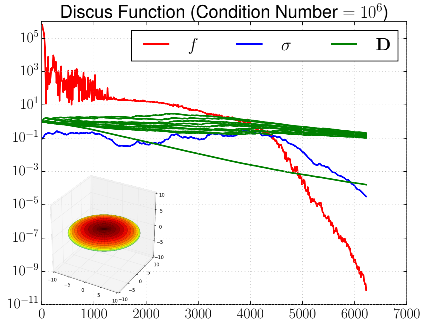
Learning a Short/Long Axis

Rank-one and Rank-μ Covariance Matrix Update

$$C \leftarrow (1 - c_1 - c_\mu)C + c_1 p_c p_c{}^T + c_\mu \sum_{i=1}^{\mu} w_i y_{i:\lambda} y_{i:\lambda}^T$$

nonnegative definite, i.e., increase covariances





n=10

Active Update

utilize negative weights [Jastrebski and Arnold, 2006]

Rank-one and Rank-μ Active Covariance Matrix Update

$$C \leftarrow (1 - c_1 - c_\mu \sum_{i=1}^{\lambda} w_i)C + c_1 p_c p_c{}^T + c_\mu \sum_{i=1}^{\lambda} w_i y_{i:\lambda} y_{i:\lambda}^T$$

where $w_i \in \mathbb{R}$.

Before: $w_i > 0$ $(i = 1, \dots, \mu)$, $w_i = 0$ $(i > \mu)$ and $\sum_{i=1}^{\mu} w_i = 1$.

Examples

$$w_i = \begin{cases} \frac{1}{\mu} & i \leq \mu \\ 0 & \text{otherwise} \end{cases} \Rightarrow C \leftarrow (1 - c_1 - c_\mu)C + c_1 p_c p_c{}^T + c_\mu \sum_{i=1}^{\lambda} w_i y_{i:\lambda} y_{i:\lambda}^T$$

$$w_i = \begin{cases} \frac{1}{\mu} & i \leq \mu \\ -\frac{1}{\mu}\left(1 + \frac{c_1}{c_\mu}\right) & i \geq \lambda - \mu \\ 0 & \text{otherwise} \end{cases} \Rightarrow C \leftarrow C + c_1 p_c p_c{}^T + c_\mu \sum_{i=1}^{\lambda} w_i y_{i:\lambda} y_{i:\lambda}^T$$

[Jastrebski and Arnold, 2006] Jastrebski, G. and Arnold, D. V. (2006). Improving Evolution Strategies through Active Covariance Matrix Adaptation. In 2006 IEEE Congress on Evolutionary Computation, pages 9719–9726.

Active Update

utilize negative weights [Jastrebski and Arnold, 2006]

Active Update (rewriting)

decreasing the variances in unpromising directions

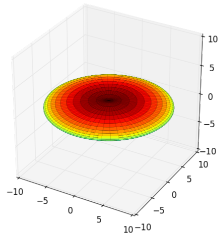
$$C \leftarrow C + c_1 p_c p_c{}^T + c_\mu \sum_{i=1}^{\mu} w_i y_{i:\lambda} y_{i:\lambda}^T - c_\mu \sum_{i=\lambda-\mu}^{\lambda} |w_i| y_{i:\lambda} y_{i:\lambda}^T$$

increasing the variances in promising directions

- increases the variance in the directions of p_c and promising steps $y_{i:\lambda}$ $(i \leq \mu)$
- decrease the variance in the directions of unpromising steps $y_{i:\lambda}$ $(i \geq \lambda - \mu)$
- keep the variance in the subspace orthogonal to the above

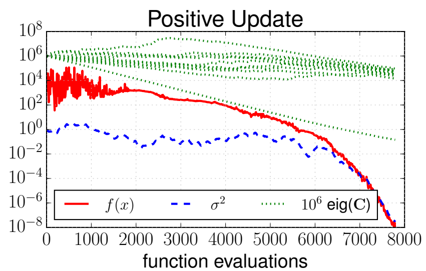
[Jastrebski and Arnold, 2006] Jastrebski, G. and Arnold, D. V. (2006). Improving Evolution Strategies through Active Covariance Matrix Adaptation. In 2006 IEEE Congress on Evolutionary Computation, pages 9719–9726.

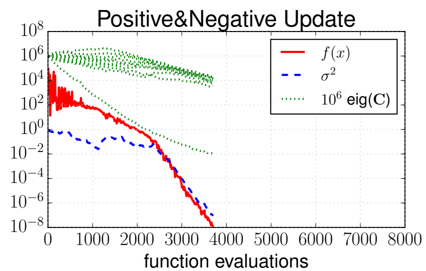
On 10D Discus Function



10D Discus Function (axis ratio: $\alpha = 10^3$)

$$f(x) = \alpha^2 \cdot x_1^2 + \sum_{i=1}^{n} x_i^2$$

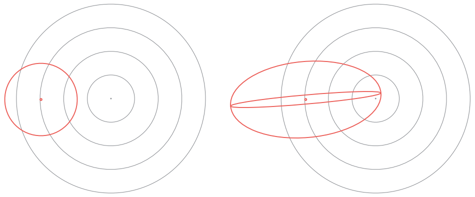




- Positive: wait for the smallest eig(C) decreasing
- Active: decrease the smallest eig(C) actively

On 10D Sphere Function

Where C converges towards when m is fixed?



Summary

Active Covariance Matrix Adaptation + Cumulation

$$C \leftarrow (1 - c_1 - c_\mu \textstyle\sum_{i=1}^{\lambda} w_i) C + c_1 p_c p_c{}^T + c_\mu \displaystyle\sum_{i=1}^{\lambda} w_i y_{i:\lambda} y_{i:\lambda}^{\mathrm{T}}$$

- w_i is positive for $i \le \mu$, negative for $i \ge \lambda - \mu$
- our choice: $\sum_{i=1}^{\mu} w_i = 1$, $\sum_{i=\lambda-\mu}^{\lambda} |w_i| = 1 + c_1/c_\mu$

These components compensate each other

- cumulation: excels to learn a long axis, but inefficient for a large λ
- rank-μ update: efficient for a large λ
- active update: effective to learn short axes

An important yet solvable issue of active update

- The positive definiteness of C will be violated if $c_\mu \sum_{i=\lambda-\mu}^{\lambda} |w_i|$ is not small enough
- The positive definiteness can be guaranteed w.p.1 by controlling w_i at each iteration

Summary of Equations

The Covariance Matrix Adaptation Evolution Strategy

Input: $m \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, λ (problem dependent)

Initialize: $\mathbf{C} = \mathbf{I}$, and $p_c = \mathbf{0}$, $p_\sigma = \mathbf{0}$,

Set: $c_c \approx 4/n$, $c_\sigma \approx 4/n$, $c_1 \approx 2/n^2$, $c_\mu \approx \mu_w/n^2$, $c_1 + c_\mu \le 1$, $d_\sigma \approx 1$

and $w_{i=1\ldots\lambda}$ such that $\mu_w = \frac{1}{\sum_{i=1}^{\mu} w_i{}^2} \approx 0.3\,\lambda$

While not terminate

$$x_i = m + \sigma y_i, \quad y_i \sim \mathcal{N}_i(\mathbf{0}, \mathbf{C}), \quad \text{for } i = 1, \ldots, \lambda \qquad \text{sampling}$$

$$m \leftarrow \textstyle\sum_{i=1}^{\mu} w_i x_{i:\lambda} = m + \sigma y_w \quad \text{where } y_w = \sum_{i=1}^{\mu} w_i y_{i:\lambda} \qquad \text{update mean}$$

$$p_c \leftarrow (1 - c_c) p_c + \mathbb{1}_{\{\|p_\sigma\| < 1.5\sqrt{n}\}} \sqrt{1 - (1 - c_c)^2} \sqrt{\mu_w}\, y_w \qquad \text{cumulation for } \mathbf{C}$$

possibly replaced with TPA or other step-size control

$$C \leftarrow (1 - c_1 - c_\mu \textstyle\sum_{i=1}^{\lambda} w_i) C + c_1 p_c p_c{}^T + c_\mu \sum_{i=1}^{\lambda} w_i y_{i:\lambda} y_{i:\lambda}^{\mathrm{T}} \qquad \text{update } \mathbf{C}$$

use TPA for example when ineffective axes exist

Not covered on this slide: termination, restarts, useful output, boundaries and encoding

Summary

Further problem difficulties

- ruggedness
  - well-structured
    a large population size
  - weakly-structured
    restart with different initialization
  - noisy
    averaging, increasing population size
- constraint
  - box-constraint
    adaptive penalty
  - linear constraint, black-box constraint
    resampling, repair, augmented Lagrangian
- high dimensionality
  exploiting the structure (restricted covariance matrix)

Natural Gradient Descend

- Consider $\arg\min_{\boldsymbol{\theta}} \mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta})$ under the sampling distribution $\boldsymbol{x} \sim p(.|\boldsymbol{\theta})$

  we could improve $\mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta})$ by following the gradient $\nabla_{\boldsymbol{\theta}}\mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta})$:

  $$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta}), \qquad \eta > 0$$

  $\nabla_{\boldsymbol{\theta}}$ depends on the parameterization of the distribution, therefore

- Consider the natural gradient of the expected transformed fitness

  $$\widetilde{\nabla}_{\boldsymbol{\theta}} \mathrm{E}(w \circ P_f(f(\boldsymbol{x}))|\boldsymbol{\theta}) = F_{\boldsymbol{\theta}}^{-1} \nabla_{\boldsymbol{\theta}} \mathrm{E}(w \circ P_f(f(\boldsymbol{x}))|\boldsymbol{\theta}) = \mathrm{E}(w \circ P_f(f(\boldsymbol{x})) F_{\boldsymbol{\theta}}^{-1} \nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{x}|\boldsymbol{\theta}))$$

  using the Fisher information matrix $F_{\boldsymbol{\theta}} = \left(\left(E \frac{\partial^2 \log p(x|\theta)}{\partial\theta_i \partial\theta_j}\right)\right)_{ij}$ of the density p.

  The natural gradient is invariant under re-parameterization of the distribution.

- A Monte-Carlo approximation reads

  $$\widetilde{\nabla}_{\boldsymbol{\theta}} \widehat{\mathrm{E}}(\widehat{w}(f(\boldsymbol{x}))|\boldsymbol{\theta}) = \sum_{i=1}^{\lambda} w_i F_{\boldsymbol{\theta}}^{-1} \nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{x}_{i:\lambda}|\boldsymbol{\theta}), \quad w_i = \widehat{w}(f(\boldsymbol{x}_{i:\lambda})|\boldsymbol{\theta})$$

Theoretical Foundations

Natural Gradient Descend

- Consider $\arg\min_{\boldsymbol{\theta}} \mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta})$ under the sampling distribution $\boldsymbol{x} \sim p(.|\boldsymbol{\theta})$

  we could improve $\mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta})$ by following the gradient $\nabla_{\boldsymbol{\theta}}\mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta})$:

  $$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta}), \qquad \eta > 0$$

  $\nabla_{\boldsymbol{\theta}}$ depends on the parameterization of the distribution, therefore

- Consider the natural gradient of the expected transformed fitness

  $$\widetilde{\nabla}_{\boldsymbol{\theta}} \mathrm{E}(w \circ P_f(f(\boldsymbol{x}))|\boldsymbol{\theta}) = F_{\boldsymbol{\theta}}^{-1} \nabla_{\boldsymbol{\theta}} \mathrm{E}(w \circ P_f(f(\boldsymbol{x}))|\boldsymbol{\theta}) = \mathrm{E}(w \circ P_f(f(\boldsymbol{x})) F_{\boldsymbol{\theta}}^{-1} \nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{x}|\boldsymbol{\theta}))$$

  using the Fisher information matrix $F_{\boldsymbol{\theta}} = \left(\left(E \frac{\partial^2 \log p(x|\theta)}{\partial\theta_i \partial\theta_j}\right)\right)_{ij}$ of the density p.

  The natural gradient is invariant under re-parameterization of the distribution.

- A Monte-Carlo approximation reads

  $$\widetilde{\nabla}_{\boldsymbol{\theta}} \widehat{\mathrm{E}}(\widehat{w}(f(\boldsymbol{x}))|\boldsymbol{\theta}) = \sum_{i=1}^{\lambda} w_i F_{\boldsymbol{\theta}}^{-1} \nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{x}_{i:\lambda}|\boldsymbol{\theta}), \quad w_i = \widehat{w}(f(\boldsymbol{x}_{i:\lambda})|\boldsymbol{\theta})$$

Theoretical Foundations

Natural Gradient Descend

- Consider $\arg\min_{\boldsymbol{\theta}} \mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta})$ under the sampling distribution $\boldsymbol{x} \sim p(.|\boldsymbol{\theta})$

  we could improve $\mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta})$ by following the gradient $\nabla_{\boldsymbol{\theta}}\mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta})$:

  $$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta}), \qquad \eta > 0$$

  $\nabla_{\boldsymbol{\theta}}$ depends on the parameterization of the distribution, therefore

- Consider the natural gradient of the expected transformed fitness

  $$\widetilde{\nabla}_{\boldsymbol{\theta}} \mathrm{E}(w \circ P_f(f(\boldsymbol{x}))|\boldsymbol{\theta}) = F_{\boldsymbol{\theta}}^{-1} \nabla_{\boldsymbol{\theta}} \mathrm{E}(w \circ P_f(f(\boldsymbol{x}))|\boldsymbol{\theta}) = \mathrm{E}(w \circ P_f(f(\boldsymbol{x})) F_{\boldsymbol{\theta}}^{-1} \nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{x}|\boldsymbol{\theta}))$$

  using the Fisher information matrix $F_{\boldsymbol{\theta}} = \left(\left(E \frac{\partial^2 \log p(x|\theta)}{\partial\theta_i \partial\theta_j}\right)\right)_{ij}$ of the density p.

  The natural gradient is invariant under re-parameterization of the distribution.

- A Monte-Carlo approximation reads

  $$\widetilde{\nabla}_{\boldsymbol{\theta}} \widehat{\mathrm{E}}(\widehat{w}(f(\boldsymbol{x}))|\boldsymbol{\theta}) = \sum_{i=1}^{\lambda} w_i F_{\boldsymbol{\theta}}^{-1} \nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{x}_{i:\lambda}|\boldsymbol{\theta}), \quad w_i = \widehat{w}(f(\boldsymbol{x}_{i:\lambda})|\boldsymbol{\theta})$$

Natural Gradient Descend

- Consider $\arg\min_{\boldsymbol{\theta}} \mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta})$ under the sampling distribution $\boldsymbol{x} \sim p(.|\boldsymbol{\theta})$ we could improve $\mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta})$ by following the gradient $\nabla_{\boldsymbol{\theta}}\mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta})$:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\nabla_{\boldsymbol{\theta}}\mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta}), \qquad \eta > 0$$

  $\nabla_{\boldsymbol{\theta}}$ depends on the parameterization of the distribution, therefore

- Consider the natural gradient of the expected transformed fitness

$$\tilde{\nabla}_{\boldsymbol{\theta}}\,\mathrm{E}(w \circ P_f(f(\boldsymbol{x}))|\boldsymbol{\theta}) = F_{\boldsymbol{\theta}}^{-1}\nabla_{\boldsymbol{\theta}}\mathrm{E}(w \circ P_f(f(\boldsymbol{x}))|\boldsymbol{\theta})$$
$$= \mathrm{E}(w \circ P_f(f(\boldsymbol{x}))F_{\boldsymbol{\theta}}^{-1}\nabla_{\boldsymbol{\theta}}\ln p(\boldsymbol{x}|\boldsymbol{\theta}))$$

  using the Fisher information matrix $F_{\boldsymbol{\theta}} = \left(\left(E\frac{\partial^2 \log p(x|\theta)}{\partial\theta_i\partial\theta_j}\right)\right)_{ij}$ of the density p.

  The natural gradient is invariant under re-parameterization of the distribution.

- A Monte-Carlo approximation reads

$$\tilde{\nabla}_{\boldsymbol{\theta}}\,\widehat{\mathrm{E}}(\widehat{w}(f(\boldsymbol{x}))|\boldsymbol{\theta}) = \sum_{i=1}^{\lambda} w_i\,F_{\boldsymbol{\theta}}^{-1}\nabla_{\boldsymbol{\theta}}\ln p(\boldsymbol{x}_{i:\lambda}|\boldsymbol{\theta}), \quad w_i = \widehat{w}(f(\boldsymbol{x}_{i:\lambda})|\boldsymbol{\theta})$$

Theoretical Foundations

Natural Gradient Descend

- Consider $\arg\min_{\boldsymbol{\theta}} \mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta})$ under the sampling distribution $\boldsymbol{x} \sim p(.|\boldsymbol{\theta})$ we could improve $\mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta})$ by following the gradient $\nabla_{\boldsymbol{\theta}}\mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta})$:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\nabla_{\boldsymbol{\theta}}\mathrm{E}(f(\boldsymbol{x})|\boldsymbol{\theta}), \qquad \eta > 0$$

  $\nabla_{\boldsymbol{\theta}}$ depends on the parameterization of the distribution, therefore

- Consider the natural gradient of the expected transformed fitness

$$\tilde{\nabla}_{\boldsymbol{\theta}}\,\mathrm{E}(w \circ P_f(f(\boldsymbol{x}))|\boldsymbol{\theta}) = F_{\boldsymbol{\theta}}^{-1}\nabla_{\boldsymbol{\theta}}\mathrm{E}(w \circ P_f(f(\boldsymbol{x}))|\boldsymbol{\theta})$$
$$= \mathrm{E}(w \circ P_f(f(\boldsymbol{x}))F_{\boldsymbol{\theta}}^{-1}\nabla_{\boldsymbol{\theta}}\ln p(\boldsymbol{x}|\boldsymbol{\theta}))$$

  using the Fisher information matrix $F_{\boldsymbol{\theta}} = \left(\left(E\frac{\partial^2 \log p(x|\theta)}{\partial\theta_i\partial\theta_j}\right)\right)_{ij}$ of the density p.

  The natural gradient is invariant under re-parameterization of the distribution.

- A Monte-Carlo approximation reads

$$\tilde{\nabla}_{\boldsymbol{\theta}}\,\widehat{\mathrm{E}}(\widehat{w}(f(\boldsymbol{x}))|\boldsymbol{\theta}) = \sum_{i=1}^{\lambda} w_i\,F_{\boldsymbol{\theta}}^{-1}\nabla_{\boldsymbol{\theta}}\ln p(\boldsymbol{x}_{i:\lambda}|\boldsymbol{\theta}), \quad w_i = \widehat{w}(f(\boldsymbol{x}_{i:\lambda})|\boldsymbol{\theta})$$

CMA-ES = Natural Evolution Strategy + Cumulation

Natural gradient descend using the MC approximation and the normal distribution

- Rewriting the update of the distribution mean

$$\boldsymbol{m}_{\text{new}} \leftarrow \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda} = \boldsymbol{m} + \underbrace{\sum_{i=1}^{\mu} w_i(\boldsymbol{x}_{i:\lambda} - \boldsymbol{m})}_{\text{natural gradient for mean } \frac{\tilde{\partial}}{\partial \boldsymbol{m}}\widehat{\mathrm{E}}(w \circ P_f(f(\boldsymbol{x}))|\boldsymbol{m},\mathbf{C})}$$

- Rewriting the update of the covariance matrix[13]

$$\mathbf{C}_{\text{new}} \leftarrow \mathbf{C} + c_1(\overbrace{\boldsymbol{p}_c\boldsymbol{p}_c^{\mathrm{T}}}^{\text{rank one}} - \mathbf{C})$$
$$+ \underbrace{\frac{c_\mu}{\sigma^2}\sum_{i=1}^{\mu} w_i\left(\overbrace{(\boldsymbol{x}_{i:\lambda} - \boldsymbol{m})\,(\boldsymbol{x}_{i:\lambda} - \boldsymbol{m})^{\mathrm{T}}}^{\text{rank-}\mu} - \sigma^2\mathbf{C}\right)}_{\text{natural gradient for covariance matrix } \frac{\tilde{\partial}}{\partial \mathbf{C}}\widehat{\mathrm{E}}(w \circ P_f(f(\boldsymbol{x}))|\boldsymbol{m},\mathbf{C})}$$

[13] Akimoto et.al. (2010): Bidirectional Relation between CMA Evolution Strategies and Natural Evolution Strategies, PPSN XI

Maximum Likelihood Update

The new distribution mean $\boldsymbol{m}$ maximizes the log-likelihood

$$\boldsymbol{m}_{\text{new}} = \arg\max_{\boldsymbol{m}} \sum_{i=1}^{\mu} w_i \log p_{\mathcal{N}}(\boldsymbol{x}_{i:\lambda}|\boldsymbol{m})$$

independently of the given covariance matrix

$$\log p_{\mathcal{N}}(\boldsymbol{x}|\boldsymbol{m},\mathbf{C}) = -\tfrac{1}{2}\log\det(2\pi\mathbf{C}) - \tfrac{1}{2}(\boldsymbol{x} - \boldsymbol{m})^{\mathrm{T}}\mathbf{C}^{-1}(\boldsymbol{x} - \boldsymbol{m})$$

$p_{\mathcal{N}}$ is the density of the multi-variate normal distribution

Maximum Likelihood Update

The new distribution mean $\boldsymbol{m}$ maximizes the log-likelihood

$$\boldsymbol{m}_{\text{new}} = \arg\max_{\boldsymbol{m}} \sum_{i=1}^{\mu} w_i \log p_{\mathcal{N}}(\boldsymbol{x}_{i:\lambda} | \boldsymbol{m})$$

independently of the given covariance matrix

The rank-μ update matrix $\mathbf{C}_\mu$ maximizes the log-likelihood

$$\mathbf{C}_\mu = \arg\max_{\mathbf{C}} \sum_{i=1}^{\mu} w_i \log p_{\mathcal{N}} \left(\frac{\boldsymbol{x}_{i:\lambda} - \boldsymbol{m}_{\text{old}}}{\sigma} \middle| \boldsymbol{m}_{\text{old}}, \mathbf{C} \right)$$

$\log p_{\mathcal{N}}(\boldsymbol{x}|\boldsymbol{m}, \mathbf{C}) = -\frac{1}{2}\log\det(2\pi\mathbf{C}) - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{m})^{\mathrm{T}}\mathbf{C}^{-1}(\boldsymbol{x} - \boldsymbol{m})$
$p_{\mathcal{N}}$ is the density of the multi-variate normal distribution

CMA-ES as a Natural Gradient

$\mathcal{O}(\lambda N^2)$ floating point (FP) multiplication + $\mathcal{O}(N^2 + \lambda N)$ FP memory. ($\lambda \in o(N)$)

1. $\sqrt{\boldsymbol{C}^{(t)}} = \textsc{MatrixSqrt}(\boldsymbol{C}^{(t)})$ (perform every $\mathcal{O}(N/\lambda)$ iter.)
2. $z_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ for $i = 1, \ldots, \lambda$
3. $x_i = \boldsymbol{m}^{(t)} + \sigma^{(t)}\sqrt{\boldsymbol{C}^{(t)}} z_i$

$\mathcal{O}(\lambda N^2)$

4. $(x_{i:\lambda})_{i=1,\ldots,\lambda} = \textsc{SortW.r.t.}^f((x_i)_{i=1,\ldots,\lambda})$

5. $\boldsymbol{p_c}^{(t+1)} = (1 - c_c)\boldsymbol{p_c}^{(t)} + \sqrt{c_c(2 - c_c)/(\sum w_i^2)} \sum_{i=1}^{\lambda} w_i(x_{i:\lambda} - \boldsymbol{m}^{(t)})/\sigma^{(t)}$
6. $\boldsymbol{p_\sigma}^{(t+1)} = (1 - c_\sigma)\boldsymbol{p_\sigma}^{(t)} + \sqrt{c_\sigma(2 - c_\sigma)/(\sum w_i^2)} \sum_{i=1}^{\lambda} w_i z_{i:\lambda}$
7. $\sigma^{(t+1)} = \sigma^{(t)} \exp\left(\frac{c_\sigma}{d_\sigma} \left(\frac{\|\boldsymbol{p_\sigma}^{(t+1)}\|}{\mathrm{E}\|\mathcal{N}(\mathbf{0}, \boldsymbol{I})\|} - 1 \right) \right)$

$\mathcal{O}(\lambda N)$

8. $\boldsymbol{m}^{(t+1)} = \boldsymbol{m}^{(t)} + \sum_{i=1}^{\lambda} w_i \tilde{\nabla} J(x_{i:\lambda})$
9. $\boldsymbol{C}^{(t+1)} = \boldsymbol{C}^{(t)} + c_\mu \sum_{i=1}^{\lambda} w_i \tilde{\nabla} J(x_{i:\lambda}) + c_1 \tilde{\nabla} J(\boldsymbol{m}^{(t)} + \sigma^{(t)} \boldsymbol{p_c}^{(t+1)})$

$\mathcal{O}(\lambda N^2)$

Bottleneck of the CMA-ES

1. $\mathcal{O}(N^2)$ Time and Space Complexities
   - to store and update $\boldsymbol{C} \in \mathbb{R}^{N \times N}$
   - to compute the eigen decomposition of $\boldsymbol{C}$
2. $\mathcal{O}(1/N^2)$ Learning Rates for $\boldsymbol{C}$-Update
   - $c_\mu \approx \mu_w / N^2$
   - $c_1 \approx 2/N^2$

$\Rightarrow$ Design a variant of CMA-ES for high dimensional search space

Variants with Restricted Covariance Matrix

CMA-ES Variants with Restricted Covariance Matrices

- Sep-CMA [Ros and Hansen, 2008]
  - $\boldsymbol{C} = \mathbf{D}$. $\mathbf{D}$: Diagonal
- VD-CMA [Akimoto et al., 2014]
  - $\boldsymbol{C} = \mathbf{D}(\mathbf{I} + \boldsymbol{v}\boldsymbol{v}^{\mathrm{T}})\mathbf{D}$. $\mathbf{D}$: Diagonal, $\boldsymbol{v} \in \mathbb{R}^N$.
- LM-CMA [Loshchilov, 2014]
  - $\boldsymbol{C} = \mathbf{I} + \sum_{i=1}^{k} \boldsymbol{v}_i\boldsymbol{v}_i^{\mathrm{T}}$. $\boldsymbol{v}_i \in \mathbb{R}^N$.
- VkD-CMA [Akimoto and Hansen, 2016]
  - $\boldsymbol{C} = \mathbf{D}(\mathbf{I} + \sum_{i=1}^{k} \boldsymbol{v}_i\boldsymbol{v}_i^{\mathrm{T}})\mathbf{D}$. $\boldsymbol{v}_i \in \mathbb{R}^N$.

[Ros and Hansen, 2008] Ros, R. and Hansen, N. (2008). A simple modification in CMA-ES achieving linear time and space complexity. In Parallel Problem Solving from Nature - PPSN X, pages 296–305. Springer.
[Akimoto et al., 2014] Akimoto, Y., Auger, A., and Hansen, N. (2014). Comparison-based natural gradient optimization in high dimension. In Proceedings of Genetic and Evolutionary Computation Conference, pages 373–380, Vancouver, BC, Canada.
[Loshchilov, 2014] Loshchilov, I. (2014). A computationally efficient limited memory cma-es for large scale optimization. In Proceedings of Genetic and Evolutionary Computation Conference, pages 397–404.
[Akimoto and Hansen, 2016] Akimoto, Y. and Hansen, N. (2016). Projection-based restricted covariance matrix adaptation for high dimension. In Genetic and Evolutionary Computation Conference, GECCO 2016, Denver, Colorado, USA, July 20-24, 2016, page (accepted). ACM.

CMA-ES as a Natural Gradient

1. $\sqrt{C^{(t)}} = \textsc{MatrixSqrt}(C^{(t)})$ (perform every $\mathcal{O}(N/\lambda)$ iter.)
2. $z_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ for $i = 1, \dots, \lambda$
3. $x_i = m^{(t)} + \sigma^{(t)} \sqrt{C^{(t)}} z_i$
4. $(x_{i:\lambda})_{i=1,\dots,\lambda} = \textsc{SortW.r.t.}^f((x_i)_{i=1,\dots,\lambda})$
5. $p_c^{(t+1)} = (1 - c_c) p_c^{(t)} + \sqrt{c_c(2 - c_c)/(\sum w_i^2)} \sum_{i=1}^{\lambda} w_i (x_{i:\lambda} - m^{(t)})/\sigma^{(t)}$
6. $p_\sigma^{(t+1)} = (1 - c_\sigma) p_\sigma^{(t)} + \sqrt{c_\sigma(2 - c_\sigma)/(\sum w_i^2)} \sum_{i=1}^{\lambda} w_i z_{i:\lambda}$
7. $\sigma^{(t+1)} = \sigma^{(t)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma^{(t+1)}\|}{\mathsf{E}\|\mathcal{N}(\mathbf{0},\mathbf{I})\|} - 1\right)\right)$
8. $m^{(t+1)} = m^{(t)} + \sum_{i=1}^{\lambda} w_i \tilde{\nabla} J(x_{i:\lambda})$
9. $C^{(t+1)} = C^{(t)} + c_\mu \sum_{i=1}^{\lambda} w_i \tilde{\nabla} J(x_{i:\lambda}) + c_1 \tilde{\nabla} J(m^{(t)} + \sigma^{(t)} p_c^{(t+1)})$

Theoretical Foundations | Design Variants of CMA

CMA-ES as a Natural Gradient

$\mathcal{O}(\lambda N^2)$ floating point (FP) multiplication + $\mathcal{O}(N^2 + \lambda N)$ FP memory. ($\lambda \in o(N)$)

1. $\sqrt{C^{(t)}} = \textsc{MatrixSqrt}(C^{(t)})$ (perform every $\mathcal{O}(N/\lambda)$ iter.)
2. $z_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ for $i = 1, \dots, \lambda$
3. $x_i = m^{(t)} + \sigma^{(t)} \sqrt{C^{(t)}} z_i$

$\mathcal{O}(\lambda N^2)$

4. $(x_{i:\lambda})_{i=1,\dots,\lambda} = \textsc{SortW.r.t.}^f((x_i)_{i=1,\dots,\lambda})$
5. $p_c^{(t+1)} = (1 - c_c) p_c^{(t)} + \sqrt{c_c(2 - c_c)/(\sum w_i^2)} \sum_{i=1}^{\lambda} w_i (x_{i:\lambda} - m^{(t)})/\sigma^{(t)}$
6. $p_\sigma^{(t+1)} = (1 - c_\sigma) p_\sigma^{(t)} + \sqrt{c_\sigma(2 - c_\sigma)/(\sum w_i^2)} \sum_{i=1}^{\lambda} w_i z_{i:\lambda}$
7. $\sigma^{(t+1)} = \sigma^{(t)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma^{(t+1)}\|}{\mathsf{E}\|\mathcal{N}(\mathbf{0},\mathbf{I})\|} - 1\right)\right)$

$\mathcal{O}(\lambda N)$

8. $m^{(t+1)} = m^{(t)} + \sum_{i=1}^{\lambda} w_i (x_{i:\lambda} - m^{(t)})$
9. $C^{(t+1)} = C^{(t)} + c_\mu \sum_{i=1}^{\lambda} w_i \left(\frac{x_{i:\lambda} - m^{(t)}}{\sigma^{(t)}} \frac{(x_{i:\lambda} - m^{(t)})^{\mathrm{T}}}{\sigma^{(t)}} - C^{(t)}\right) + c_1 (p_c^{(t+1)} p_c^{(t+1)\mathrm{T}} - C^{(t)})$

$\mathcal{O}(\lambda N^2)$

Theoretical Foundations | Design Variants of CMA

Sep-CMA-ES as a Natural Gradient

$\mathcal{O}(\lambda N)$ floating point (FP) multiplication + $\mathcal{O}(N + \lambda N)$ FP memory. ($\lambda \in o(N)$)

1. $\sqrt{C^{(t)}} = \textsc{MatrixSqrt}(C^{(t)})$
2. $z_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ for $i = 1, \dots, \lambda$
3. $x_i = m^{(t)} + \sigma^{(t)} \sqrt{C^{(t)}} z_i$

$\mathcal{O}(\lambda N)$

4. $(x_{i:\lambda})_{i=1,\dots,\lambda} = \textsc{SortW.r.t.}^f((x_i)_{i=1,\dots,\lambda})$
5. $p_c^{(t+1)} = (1 - c_c) p_c^{(t)} + \sqrt{c_c(2 - c_c)/(\sum w_i^2)} \sum_{i=1}^{\lambda} w_i (x_{i:\lambda} - m^{(t)})/\sigma^{(t)}$
6. $p_\sigma^{(t+1)} = (1 - c_\sigma) p_\sigma^{(t)} + \sqrt{c_\sigma(2 - c_\sigma)/(\sum w_i^2)} \sum_{i=1}^{\lambda} w_i z_{i:\lambda}$
7. $\sigma^{(t+1)} = \sigma^{(t)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma^{(t+1)}\|}{\mathsf{E}\|\mathcal{N}(\mathbf{0},\mathbf{I})\|} - 1\right)\right)$

$\mathcal{O}(\lambda N)$

8. $m^{(t+1)} = m^{(t)} + \sum_{i=1}^{\lambda} w_i (x_{i:\lambda} - m^{(t)})$
9. $C^{(t+1)} = C^{(t)} + c_\mu \sum_{i=1}^{\lambda} w_i \mathrm{diag}\left(\frac{x_{i:\lambda} - m^{(t)}}{\sigma^{(t)}} \frac{(x_{i:\lambda} - m^{(t)})^{\mathrm{T}}}{\sigma^{(t)}} - C^{(t)}\right) + c_1 \mathrm{diag}(p_c^{(t+1)} p_c^{(t+1)\mathrm{T}} - C^{(t)})$

$\mathcal{O}(\lambda N)$

Plug diagonal C and compute the natural gradient

The Continuous Search Problem

Difficulties of a non-linear optimization problem are

- dimensionality and non-separabitity
  demands to exploit problem structure, e.g. neighborhood
  cave: design of benchmark functions
- ill-conditioning
  demands to acquire a second order model
- ruggedness
  demands a non-local (stochastic? population based?) approach

Main Characteristics of (CMA) Evolution Strategies

1. Multivariate normal distribution to generate new search points
   follows the maximum entropy principle
2. Rank-based selection
   implies invariance, same performance on $g(f(\boldsymbol{x}))$ for any increasing g
   more invariance properties are featured
3. Step-size control facilitates fast (log-linear) convergence and possibly linear scaling with the dimension
   in CMA-ES based on an evolution path (a non-local trajectory)
4. *Covariance matrix adaptation (CMA)* increases the likelihood of previously successful steps and can improve performance by orders of magnitude
   the update follows the natural gradient
   $\mathbf{C} \propto \boldsymbol{H}^{-1} \iff$ adapts a variable metric
   $\iff$ new (rotated) problem representation
   $\implies f : \boldsymbol{x} \mapsto g(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x})$ reduces to $\boldsymbol{x} \mapsto \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}$

Limitations

of CMA Evolution Strategies

- internal CPU-time: $10^{-8}n^2$ seconds per function evaluation on a 2GHz PC, tweaks are available
  1 000 000 f-evaluations in 100-D take 100 seconds *internal* CPU-time
- better methods are presumably available in case of
  - partly separable problems
  - specific problems, for example with cheap gradients
    specific methods
  - small dimension ($n \ll 10$)
    for example Nelder-Mead
  - small running times (number of f-evaluations $< 100n$)
    model-based methods

Thank You

Source code for CMA-ES in C, Java, Matlab, Octave, Python, Scilab is available at `http://www.lri.fr/~hansen/cmaes_inmatlab.html`