Stochastic Optimization in Continuous Domain: Challenges and Approaches

Nikolaus Hansen

July 2007

Content

Einstein once spoke of the "unreasonable effectiveness of mathematics" in describing how the natural world works. Whether one is talking about basic physics, about the increasingly important environmental sciences, or the transmission of disease, mathematics is never any more, or any less, than a way of thinking clearly. As such, it always has been and always will be a valuable tool, but only valuable when it is part of a larger arsenal embracing analytic experiments and, above all, wide-ranging imagination.

Lord Kay

Problem Statement

Continuous Domain Search/Optimization

- Task: minimize a 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

- Task: minimize a 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

- Task: minimize a 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

- Task: minimize a 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

- Task: minimize a objective function $f : \mathcal{X} \subseteq \mathbb{R}^n \to \mathbb{R}, \; \boldsymbol{x} \mapsto f(\boldsymbol{x})$
- Goal
  - fast convergence to the global optimum
    ... or to a robust solution $\boldsymbol{x}$
  - solution $\boldsymbol{x}$ with small function value with least search cost
    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
- Problem
  - exhaustive search is infeasible
  - deterministic search is often not successful
  - naive random search takes too long

Approach: stochastic search, Evolutionary Algorithms

Problem Statement

Continuous Domain Search/Optimization

- Task: minimize a objective function $f : \mathcal{X} \subseteq \mathbb{R}^n \to \mathbb{R}, \ \boldsymbol{x} \mapsto f(\boldsymbol{x})$
- Goal
  - fast convergence to the global optimum
    ... or to a robust solution $\boldsymbol{x}$
  - solution $\boldsymbol{x}$ with small function value with least search cost
    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
- Problem
  - exhaustive search is infeasible
  - deterministic search is often not successful
  - naive random search takes too long

Approach: stochastic search, Evolutionary Algorithms

Problem Statement

Continuous Domain Search/Optimization

- Task: minimize a objective function $f : \mathcal{X} \subseteq \mathbb{R}^n \to \mathbb{R},\ \boldsymbol{x} \mapsto f(\boldsymbol{x})$
- Goal
  - fast convergence to the global optimum

    … or to a robust solution $\boldsymbol{x}$
  - solution $\boldsymbol{x}$ with small function value with least search cost

    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
- Problem
  - exhaustive search is infeasible
  - deterministic search is often not successful
  - naive random search takes too long

Approach: stochastic search, Evolutionary Algorithms

Problem Statement

Continuous Domain Search/Optimization

- Task: minimize a objective function $f : \mathcal{X} \subseteq \mathbb{R}^n \to \mathbb{R},\ \boldsymbol{x} \mapsto f(\boldsymbol{x})$
- Goal
  - fast convergence to the global optimum

    ...or to a robust solution $\boldsymbol{x}$
  - solution $\boldsymbol{x}$ with small function value with least search cost

    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
- Problem
  - exhaustive search is infeasible
  - deterministic search is often not successful
  - naive random search takes too long

Approach: stochastic search, Evolutionary Algorithms

Analogies

Evolutionary Computation		Optimization
individual, offspring, parent	$\longleftrightarrow$	candidate solution decision variables design variables object variables
population	$\longleftrightarrow$	set of candidate solutions
fitness function	$\longleftrightarrow$	objective function loss function cost function
generation	$\longleftrightarrow$	iteration

. . . function properties

Objective Function Properties

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

Additionally, f can be

- multimodal

  there are eventually many local optima

- non-smooth

  derivatives do not exist

- discontinuous
- 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 have at least moderate dimensionality, say $n \not\ll 10$, and to be *non-linear, non-convex, and non-separable*.

Additionally, f can be

- multimodal

  there are eventually many local optima
- non-smooth

  derivatives do not exist
- discontinuous
- 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 have at least moderate dimensionality, say $n \not\ll 10$, and to be *non-linear, non-convex, and non-separable*.

Additionally, f can be

- multimodal

  there are eventually many local optima

- non-smooth

  derivatives do not exist

- discontinuous
- 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?

- ruggedness

  non-smooth, discontinuous, multimodal, and/or noisy function

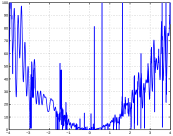
- dimensionality

  (considerably) larger than three

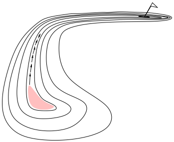
- non-separability

  dependencies between the objective variables

- ill-conditioning



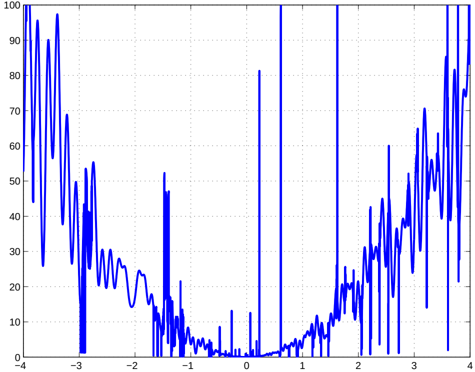
cut from 3-D example, solvable with CMA-ES



a narrow ridge

What Makes a Function Difficult to Solve?

Why stochastic search?



cut from 3-D example, solvable with CMA-ES

Curse of Dimensionality

The term *Curse of dimensionality* (Richard Bellman) refers to problems caused by the rapid increase in volume associated with adding extra dimensions to a (mathematical) space.

Consider placing 100 points onto a real interval, say $[-1, 1]$. To get similar coverage, in terms of distance between adjacent points, of the 10-dimensional space $[-1, 1]^{10}$ would require $100^{10} = 10^{20}$ points. A 100 points appear now as isolated points in a vast empty space.

Consequently, a search policy (e.g. exhaustive search) that is valuable in small dimensions might be useless in moderate or large dimensional search spaces.

Separable Problems

Definition (Separable Problem)

A function f is separable if

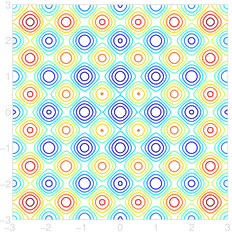
$$\arg\min_{(x_1,\ldots,x_n)} f(x_1,\ldots,x_n) = \left(\arg\min_{x_1} f(x_1,\ldots),\ldots,\arg\min_{x_n} f(\ldots,x_n)\right)$$

$\Rightarrow$ it follows that f can be optimized in a sequence of n independent 1-D optimization processes

Example: Additively decomposable functions

$$f(x_1,\ldots,x_n) = \sum_{i=1}^{n} f_i(x_i)$$

Rastrigin function



Separable Problems

Definition (Separable Problem)

A function f is separable if

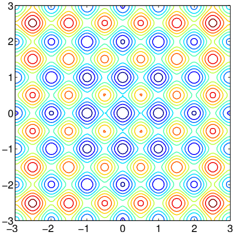
$$\arg\min_{(x_1,\ldots,x_n)} f(x_1,\ldots,x_n) = \left(\arg\min_{x_1} f(x_1,\ldots),\ldots,\arg\min_{x_n} f(\ldots,x_n)\right)$$

$\Rightarrow$ it follows that f can be optimized in a sequence of n independent 1-D optimization processes

Example: Additively decomposable functions

$$f(x_1,\ldots,x_n) = \sum_{i=1}^{n} f_i(x_i)$$

Rastrigin function



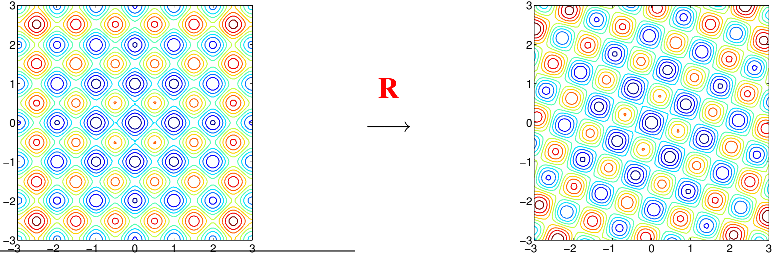
Non-Separable Problems

Building a non-separable problem from a separable one

Rotating the coordinate system

- $f : \boldsymbol{x} \mapsto f(\boldsymbol{x})$ separable
- $f : \boldsymbol{x} \mapsto f(\mathbf{R}\boldsymbol{x})$ non-separable

$\mathbf{R}$ rotation matrix



12

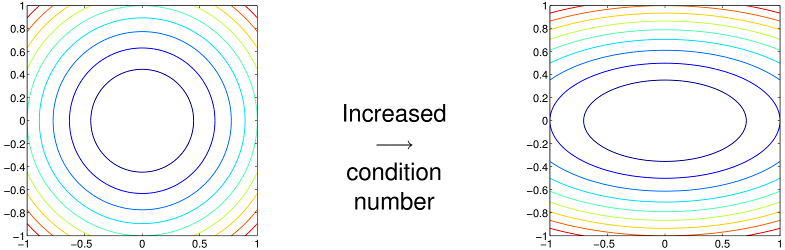
[1] Hansen, Ostermeier, Gawelczyk (1995). On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. Sixth ICGA, pp. 57-64, Morgan Kaufmann

[2] Salomon (1996). "Reevaluating Genetic Algorithm Performance under Coordinate Rotation of Benchmark Functions; A survey of some theoretical and practical aspects of genetic algorithms." BioSystems, 39(3):263-278

Ill-Conditioned Problems

If f is quadratic, $f : \boldsymbol{x} \mapsto \boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x}$, ill-conditioned means a high condition number of Hessian Matrix $\boldsymbol{H}$

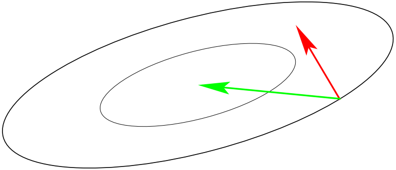
ill-conditioned means "squeezed" lines of equal function value



consider the curvature of iso-fitness lines

The Benefit of Second Order Information

Consider the convex quadratic function $f(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^*)^T \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}^*)$



gradient direction $-f'(\boldsymbol{x})^{\mathrm{T}}$

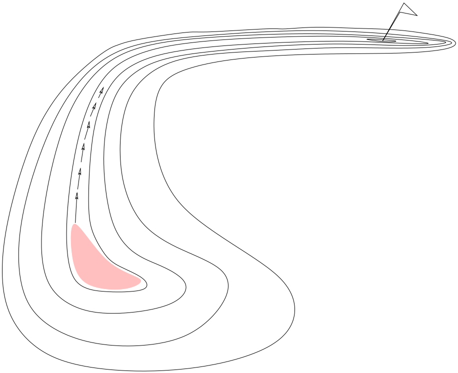
Newton direction $-\boldsymbol{H}^{-1}f'(\boldsymbol{x})^{\mathrm{T}}$

Condition number equals nine here. Condition numbers between 100 and even 10^6 can be observed in real world problems.

If $\boldsymbol{H} \approx \mathbf{I}$ (small condition number of $\boldsymbol{H}$) first order information (e.g. the gradient) is sufficient. Otherwise second order information (estimation of $\boldsymbol{H}^{-1}$) is required.

Ill-Conditioned Problems

Example: A Narrow Ridge



Volume oriented search ends up in the pink area.
To approach the optimum an ill-conditioned problem needs to be solved (e.g. by following the narrow bent ridge).[3]

[3] Whitley, Lunacek, Knight 2004. Ruffled by Ridges: How Evolutionary Algorithms Can Fail, *GECCO*

Second Order Approaches

Examples

- quasi-Newton method
- conjugate gradients
- trust region methods
- surrogate model methods
- linkage learning
- correlated mutations (self-adaptation)
- estimation of distribution algorithms

The mutual idea

capture dependencies between variables, a second-order model

. . . summary

What Makes a Function Difficult to Solve?

. . . and what can be done

The Problem	What can be done
Ruggedness	non-local search, large sampling width (step-size) as large as possible while preserving a reasonable convergence speed stochastic, non-elitistic, population-based method recombination operator serves as repair mechanism
Dimensionality, Non-Separability	exploiting the problem structure locality, neighborhood, encoding
Ill-conditioning	second order approach changes the neighborhood metric

. . . interface to real world problems

What Makes a Function Difficult to Solve?

. . . and what can be done

The Problem	What can be done
Ruggedness	non-local search, large sampling width (step-size) as large as possible while preserving a reasonable convergence speed
	stochastic, non-elitistic, population-based method recombination operator serves as repair mechanism
Dimensionality, Non-Separability	exploiting the problem structure locality, neighborhood, encoding
Ill-conditioning	second order approach changes the neighborhood metric

. . . interface to real world problems

What Makes a Function Difficult to Solve?

. . . and what can be done

The Problem	What can be done
Ruggedness	non-local search, large sampling width (step-size) as large as possible while preserving a reasonable convergence speed
	stochastic, non-elitistic, population-based method recombination operator serves as repair mechanism
Dimensionality, Non-Separability	exploiting the problem structure locality, neighborhood, encoding
Ill-conditioning	second order approach changes the neighborhood metric

. . . interface to real world problems

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

Remark: a population of solutions is used

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 $P(\boldsymbol{x}|\boldsymbol{\theta}) \to \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda \in \mathbb{R}^n$
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 Evolutionary Algorithms the distribution P is often implicitly defined via operators on a population, in particular, selection, recombination and mutation

Remark: a population of solutions is used

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

Remark: a population of solutions is used

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 $P(\boldsymbol{x}|\boldsymbol{\theta}) \to \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda \in \mathbb{R}^n$
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 Evolutionary Algorithms the distribution P is often implicitly defined via operators on a population, in particular, selection, recombination and mutation

Remark: a population of solutions is used

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

Remark: a population of solutions is used

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

Remark: a population of solutions is used

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 $P(\boldsymbol{x}|\boldsymbol{\theta}) \to \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda \in \mathbb{R}^n$
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 Evolutionary Algorithms the distribution P is often implicitly defined via operators on a population, in particular, selection, recombination and mutation

Remark: a population of solutions is used

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

Remark: a population of solutions is used

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

Remark: a population of solutions is used

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 $\boldsymbol{\theta} \leftarrow F_\theta(\boldsymbol{\theta}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda, f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_\lambda))$

In the following

- P is a multi-variate normal distribution $\mathcal{N}(\boldsymbol{m}, \sigma^2\mathbf{C}) \sim \boldsymbol{m} + \sigma\mathcal{N}(\mathbf{0}, \mathbf{C})$
- $\boldsymbol{\theta} = \{\boldsymbol{m}, \mathbf{C}, \sigma\} \in \mathbb{R}^n \times \mathbb{R}^{n \times n} \times \mathbb{R}_+$
- $F_\theta = F_\theta(\boldsymbol{\theta}, \boldsymbol{x}_{1:\lambda}, \ldots, \boldsymbol{x}_{\mu:\lambda})$, where $\mu \leq \lambda$ and $\boldsymbol{x}_{i:\lambda}$ is the i-th best of the λ points

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 $\boldsymbol{\theta} \leftarrow F_\theta(\boldsymbol{\theta}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda, f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_\lambda))$

In the following

- P is a multi-variate normal distribution $\mathcal{N}(\boldsymbol{m}, \sigma^2\mathbf{C}) \sim \boldsymbol{m} + \sigma\mathcal{N}(\mathbf{0}, \mathbf{C})$
- $\boldsymbol{\theta} = \{\boldsymbol{m}, \mathbf{C}, \sigma\} \in \mathbb{R}^n \times \mathbb{R}^{n\times n} \times \mathbb{R}_+$
- $F_\theta = F_\theta(\boldsymbol{\theta}, \boldsymbol{x}_{1:\lambda}, \ldots, \boldsymbol{x}_{\mu:\lambda})$, where $\mu \leq \lambda$ and $\boldsymbol{x}_{i:\lambda}$ is the i-th best of the λ points

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

In the following

- P is a multi-variate normal distribution $\mathcal{N}(\boldsymbol{m}, \sigma^2 \mathbf{C}) \sim \boldsymbol{m} + \sigma \mathcal{N}(\mathbf{0}, \mathbf{C})$
- $\boldsymbol{\theta} = \{\boldsymbol{m}, \mathbf{C}, \sigma\} \in \mathbb{R}^n \times \mathbb{R}^{n \times n} \times \mathbb{R}_+$
- $F_\theta = F_\theta(\boldsymbol{\theta}, \boldsymbol{x}_{1:\lambda}, \dots, \boldsymbol{x}_{\mu:\lambda})$, where $\mu \leq \lambda$ and $\boldsymbol{x}_{i:\lambda}$ is the i-th best of the λ points

Why Normal Distributions?

1. most convenient way to generate isotropic search points

   the isotropic distribution does not (unfoundedly) favor any direction, supports rotational invariance

2. maximum entropy distribution with finite variance

   there are the least possible assumptions on f in the distribution shape

3. only stable distribution with finite variance

   stable means the sum of normal variates is again normal, helpful in design and analysis of algorithms

4. widely observed in nature, for example with phenotypic traits

Why Normal Distributions?

1. most convenient way to generate isotropic search points

   the isotropic distribution does not (unfoundedly) favor any direction, supports rotational invariance

2. maximum entropy distribution with finite variance

   there are the least possible assumptions on f in the distribution shape

3. only stable distribution with finite variance

   stable means the sum of normal variates is again normal, helpful in design and analysis of algorithms

4. widely observed in nature, for example with phenotypic traits

Why Normal Distributions?

1. most convenient way to generate isotropic search points
   the isotropic distribution does not (unfoundedly) favor any direction, supports rotational invariance
2. maximum entropy distribution with finite variance
   there are the least possible assumptions on f in the distribution shape
3. only stable distribution with finite variance
   stable means the sum of normal variates is again normal, helpful in design and analysis of algorithms
4. widely observed in nature, for example with phenotypic traits

Why Normal Distributions?

1. most convenient way to generate isotropic search points

   the isotropic distribution does not (unfoundedly) favor any direction, supports rotational invariance

2. maximum entropy distribution with finite variance

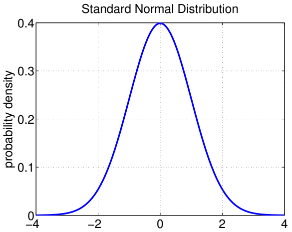
   there are the least possible assumptions on f in the distribution shape

3. only stable distribution with finite variance

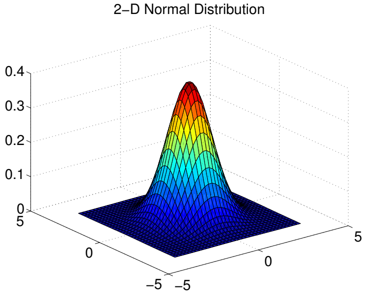
   stable means the sum of normal variates is again normal, helpful in design and analysis of algorithms

4. widely observed in nature, for example with phenotypic traits

Normal Distribution



probability density of 1-D standard normal distribution



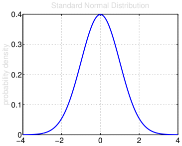
probability density of 2-D normal distribution

The Multi-Variate (n-Dimensional) Normal Distribution

Any multi-variate normal distribution $\mathcal{N}(\boldsymbol{m}, \mathbf{C})$ is uniquely determined by its mean value $\boldsymbol{m} \in \mathbb{R}^n$ and its symmetric positive definite $n \times n$ covariance matrix $\mathbf{C}$.

The mean value $\boldsymbol{m}$

- determines the displacement (translation)
- is the value with the largest density (modal value)
- the distribution is symmetric about the distribution mean

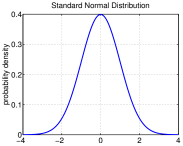


The Multi-Variate (n-Dimensional) Normal Distribution

Any multi-variate normal distribution $\mathcal{N}(\boldsymbol{m}, \mathbf{C})$ is uniquely determined by its mean value $\boldsymbol{m} \in \mathbb{R}^n$ and its symmetric positive definite $n \times n$ covariance matrix $\mathbf{C}$.

The mean value $\boldsymbol{m}$

- determines the displacement (translation)
- is the value with the largest density (modal value)
- the distribution is symmetric about the distribution mean



The covariance matrix $\mathbf{C}$ determines the shape. It has a valuable geometrical interpretation: any covariance matrix can be uniquely identified with the iso-density ellipsoid $\{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x}^{\mathrm{T}}\mathbf{C}^{-1}\boldsymbol{x} = 1\}$

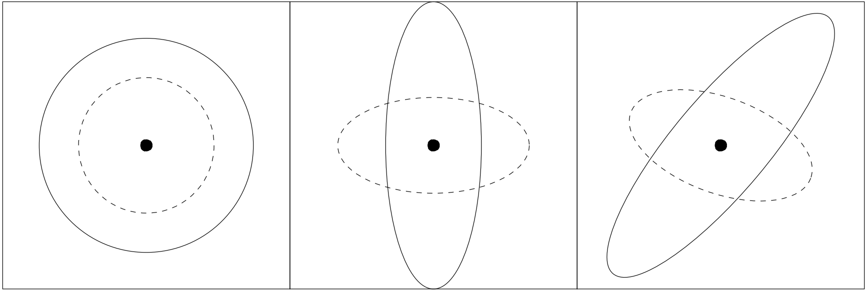
Lines of Equal Density

$\mathcal{N}(\boldsymbol{m}, \sigma^2\mathbf{I}) \sim \boldsymbol{m} + \sigma\mathcal{N}(\mathbf{0}, \mathbf{I})$	$\mathcal{N}(\boldsymbol{m}, \mathbf{D}^2) \sim \boldsymbol{m} + \mathbf{D}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$	$\mathcal{N}(\boldsymbol{m}, \mathbf{C}) \sim \boldsymbol{m} + \mathbf{C}^{\frac{1}{2}}\mathcal{N}(\mathbf{0}, \mathbf{I})$
one free parameter σ	n free parameters	$(n^2 + n)/2$ free parameters
components of $\mathcal{N}(\mathbf{0}, \mathbf{I})$ are independent standard normally distributed	components are independent, scaled	components are correlated

where $\mathbf{I}$ is the identity matrix (isotropic case) and $\mathbf{D}$ is a diagonal matrix (reasonable for separable problems) and $\mathbf{A} \times \mathcal{N}(\mathbf{0}, \mathbf{I}) \sim \mathcal{N}(\mathbf{0}, \mathbf{A}\mathbf{A}^{\mathrm{T}})$ holds for all $\mathbf{A}$.

The covariance matrix $\mathbf{C}$ determines the shape. It has a valuable geometrical interpretation: any covariance matrix can be uniquely identified with the iso-density ellipsoid $\{\boldsymbol{x} \in \mathbb{R}^n \,|\, \boldsymbol{x}^{\mathrm{T}}\mathbf{C}^{-1}\boldsymbol{x} = 1\}$

Lines of Equal Density

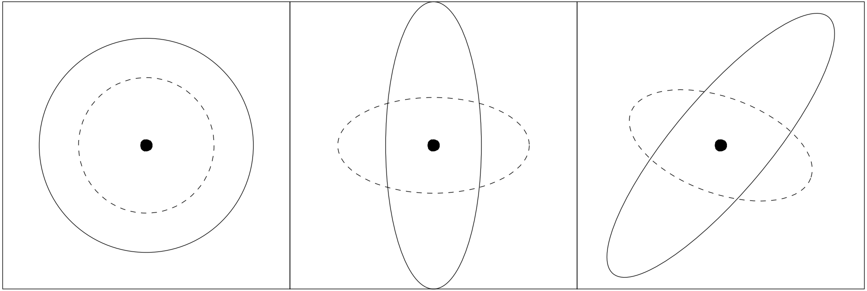


$\mathcal{N}\left(\boldsymbol{m}, \sigma^2\mathbf{I}\right) \sim \boldsymbol{m} + \sigma\mathcal{N}(\mathbf{0}, \mathbf{I})$	$\mathcal{N}\left(\boldsymbol{m}, \mathbf{D}^2\right) \sim \boldsymbol{m} + \mathbf{D}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$	$\mathcal{N}(\boldsymbol{m}, \mathbf{C}) \sim \boldsymbol{m} + \mathbf{C}^{\frac{1}{2}}\mathcal{N}(\mathbf{0}, \mathbf{I})$
one free parameter σ	n free parameters	$(n^2 + n)/2$ free parameters
components of $\mathcal{N}(\mathbf{0}, \mathbf{I})$ are independent standard normally distributed	components are independent, scaled	components are correlated

where $\mathbf{I}$ is the identity matrix (isotropic case) and $\mathbf{D}$ is a diagonal matrix (reasonable for separable problems) and $\mathbf{A} \times \mathcal{N}(\mathbf{0}, \mathbf{I}) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{A}\mathbf{A}^{\mathrm{T}}\right)$ holds for all $\mathbf{A}$.

The covariance matrix $\mathbf{C}$ determines the shape. It has a valuable geometrical interpretation: any covariance matrix can be uniquely identified with the iso-density ellipsoid $\{\boldsymbol{x} \in \mathbb{R}^n \,|\, \boldsymbol{x}^{\mathrm{T}}\mathbf{C}^{-1}\boldsymbol{x} = 1\}$

Lines of Equal Density

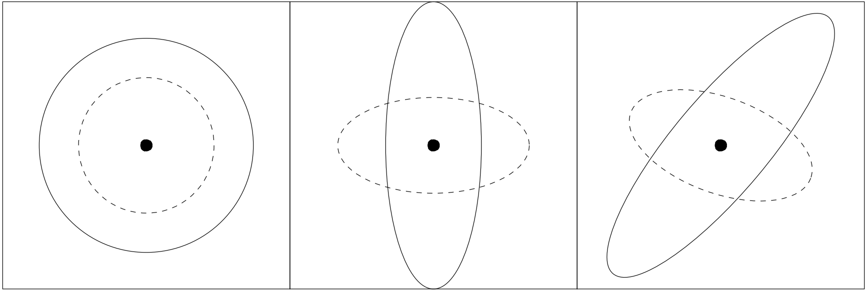


$\mathcal{N}\left(\boldsymbol{m}, \sigma^2\mathbf{I}\right) \sim \boldsymbol{m} + \sigma\mathcal{N}(\mathbf{0}, \mathbf{I})$	$\mathcal{N}\left(\boldsymbol{m}, \mathbf{D}^2\right) \sim \boldsymbol{m} + \mathbf{D}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$	$\mathcal{N}(\boldsymbol{m}, \mathbf{C}) \sim \boldsymbol{m} + \mathbf{C}^{\frac{1}{2}}\mathcal{N}(\mathbf{0}, \mathbf{I})$
one free parameter σ	n free parameters	$(n^2 + n)/2$ free parameters
components of $\mathcal{N}(\mathbf{0}, \mathbf{I})$ are independent standard normally distributed	components are independent, scaled	components are correlated

where $\mathbf{I}$ is the identity matrix (isotropic case) and $\mathbf{D}$ is a diagonal matrix (reasonable for separable problems) and $\mathbf{A} \times \mathcal{N}(\mathbf{0}, \mathbf{I}) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{A}\mathbf{A}^{\mathrm{T}}\right)$ holds for all $\mathbf{A}$.

The covariance matrix $\mathbf{C}$ determines the shape. It has a valuable geometrical interpretation: any covariance matrix can be uniquely identified with the iso-density ellipsoid $\{\boldsymbol{x} \in \mathbb{R}^n \,|\, \boldsymbol{x}^{\mathrm{T}}\mathbf{C}^{-1}\boldsymbol{x} = 1\}$

Lines of Equal Density



$\mathcal{N}\left(\boldsymbol{m}, \sigma^2\mathbf{I}\right) \sim \boldsymbol{m} + \sigma\mathcal{N}(\mathbf{0}, \mathbf{I})$	$\mathcal{N}\left(\boldsymbol{m}, \mathbf{D}^2\right) \sim \boldsymbol{m} + \mathbf{D}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$	$\mathcal{N}(\boldsymbol{m}, \mathbf{C}) \sim \boldsymbol{m} + \mathbf{C}^{\frac{1}{2}}\mathcal{N}(\mathbf{0}, \mathbf{I})$
one free parameter σ	n free parameters	$(n^2 + n)/2$ free parameters
components of $\mathcal{N}(\mathbf{0}, \mathbf{I})$ are independent standard normally distributed	components are independent, scaled	components are correlated

where $\mathbf{I}$ is the identity matrix (isotropic case) and $\mathbf{D}$ is a diagonal matrix (reasonable for separable problems) and $\mathbf{A} \times \mathcal{N}(\mathbf{0}, \mathbf{I}) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{A}\mathbf{A}^{\mathrm{T}}\right)$ holds for all $\mathbf{A}$.

Sampling New Search Points

The Mutation Operator

New search points are sampled normally distributed

$$\boldsymbol{x}_i \sim \mathcal{N}_i\left(\boldsymbol{m}, \sigma^2\mathbf{C}\right) = \boldsymbol{m} + \sigma\,\mathcal{N}_i(\mathbf{0}, \mathbf{C}) \qquad \text{for } i = 1, \ldots, \lambda$$

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

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

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

Sampling New Search Points
The Mutation Operator

New search points are sampled normally distributed

$$\boldsymbol{x}_i \sim \mathcal{N}_i\left(\boldsymbol{m}, \sigma^2\mathbf{C}\right) = \boldsymbol{m} + \sigma\,\mathcal{N}_i(\mathbf{0}, \mathbf{C}) \qquad \text{for } i = 1, \ldots, \lambda$$

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

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

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

Sampling New Search Points

The Mutation Operator

New search points are sampled normally distributed

$$\boldsymbol{x}_i \sim \mathcal{N}_i\left(\boldsymbol{m}, \sigma^2 \mathbf{C}\right) = \boldsymbol{m} + \sigma\, \mathcal{N}_i(\mathbf{0}, \mathbf{C}) \qquad \text{for } i = 1, \ldots, \lambda$$

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

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

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

Update of the Distribution Mean m

Selection and Recombination

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

Let $x_{i:\lambda}$ the i-th ranked solution point, such that $f(x_{1:\lambda}) \leq \cdots \leq f(x_{\lambda:\lambda})$.

The new mean reads

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

where

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

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

Update of the Distribution Mean $\boldsymbol{m}$

Selection and Recombination

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

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

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

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

Update of the Distribution Mean $\boldsymbol{m}$

Selection and Recombination

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

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

$$\boldsymbol{m} \leftarrow \sum_{i=1}^{\mu} w_i \boldsymbol{x}_{i:\lambda} = \boldsymbol{m} + \sigma \underbrace{\sum_{i=1}^{\mu} w_i \boldsymbol{z}_{i:\lambda}}_{=: \langle \boldsymbol{z} \rangle_{\mathrm{sel}}}$$

where

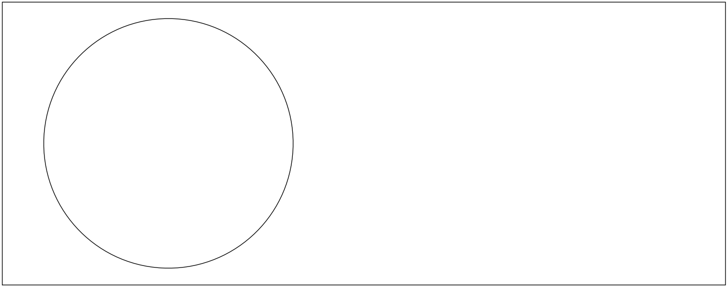
$$w_1 \geq \cdots \geq w_\mu > 0, \quad \sum_{i=1}^{\mu} w_i = 1$$

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

Covariance Matrix Adaptation

Rank-One Update

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

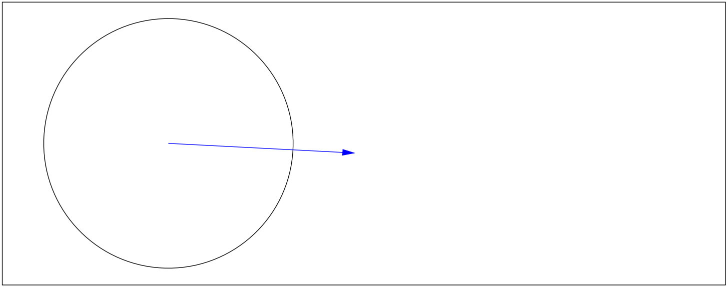


initial distribution, $\mathbf{C} = \mathbf{I}$

Covariance Matrix Adaptation

Rank-One Update

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

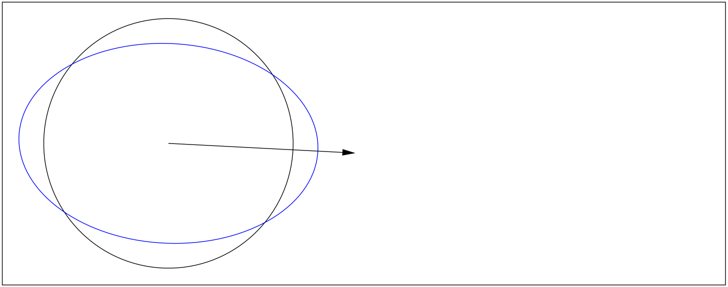


$\langle \boldsymbol{z} \rangle_{\text{sel}}$, movement of the population mean $\boldsymbol{m}$ (disregarding σ)

Covariance Matrix Adaptation

Rank-One Update

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



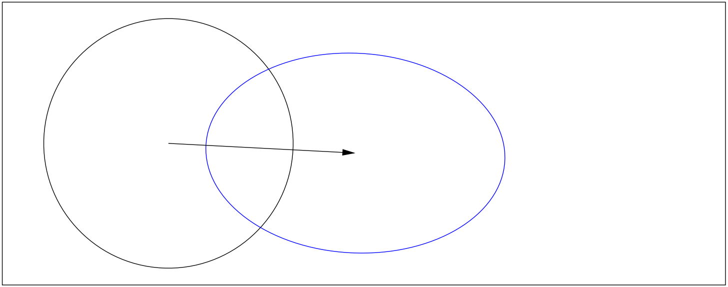
mixture of distribution $\mathbf{C}$ and step $\langle \boldsymbol{z} \rangle_{\text{sel}}$,

$\mathbf{C} \leftarrow 0.8 \times \mathbf{C} + 0.2 \times \langle \boldsymbol{z} \rangle_{\text{sel}} \langle \boldsymbol{z} \rangle_{\text{sel}}^{\mathrm{T}}$

Covariance Matrix Adaptation

Rank-One Update

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

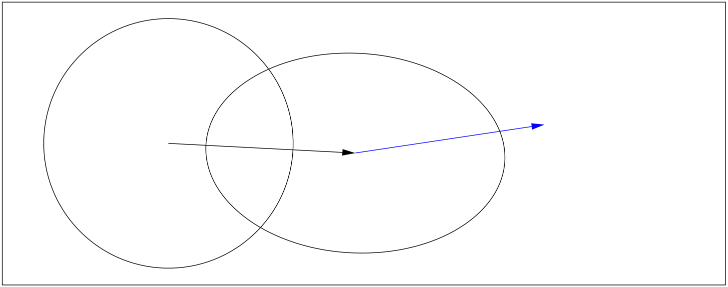


new distribution (disregarding σ)

Covariance Matrix Adaptation

Rank-One Update

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

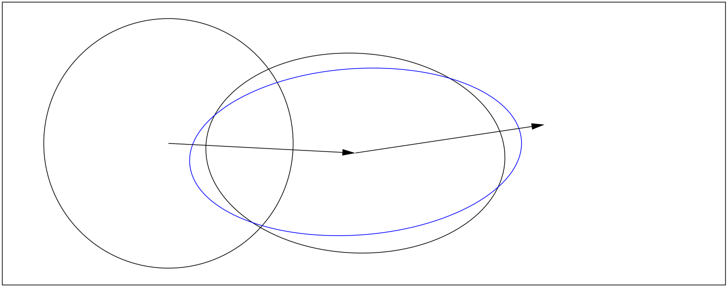


movement of the population mean $\boldsymbol{m}$

Covariance Matrix Adaptation

Rank-One Update

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

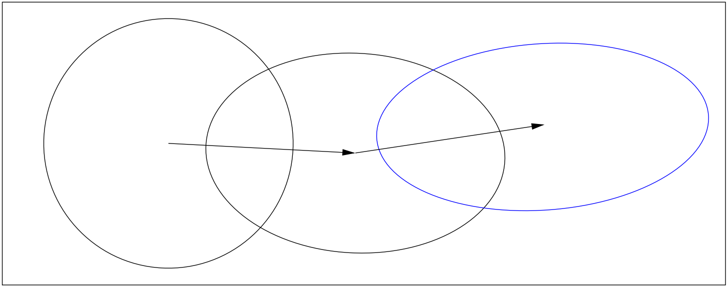


mixture of distribution $\mathbf{C}$ and step $\langle \boldsymbol{z} \rangle_{\text{sel}}$,
$\mathbf{C} \leftarrow 0.8 \times \mathbf{C} + 0.2 \times \langle \boldsymbol{z} \rangle_{\text{sel}} \langle \boldsymbol{z} \rangle_{\text{sel}}^{\mathrm{T}}$

Covariance Matrix Adaptation

Rank-One Update

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



new distribution,

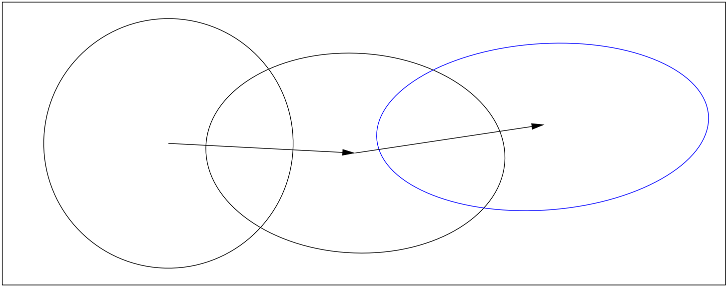
$\mathbf{C} \leftarrow 0.8 \times \mathbf{C} + 0.2 \times \langle \boldsymbol{z} \rangle_{\text{sel}} \langle \boldsymbol{z} \rangle_{\text{sel}}^{\mathrm{T}}$

the ruling principle: the adaptation increases the probability of successful steps, $\langle \boldsymbol{z} \rangle_{\text{sel}}$, to appear again

Covariance Matrix Adaptation

Rank-One Update

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



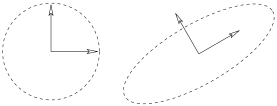
new distribution,

$\mathbf{C} \leftarrow 0.8 \times \mathbf{C} + 0.2 \times \langle \boldsymbol{z} \rangle_{\text{sel}} \langle \boldsymbol{z} \rangle_{\text{sel}}^{\mathrm{T}}$

the ruling principle: the adaptation increases the probability of successful steps, $\langle \boldsymbol{z} \rangle_{\text{sel}}$, to appear again

The covariance matrix adaptation

- learns all pairwise dependencies between variables
  off-diagonal entries in the covariance matrix reflect the dependencies



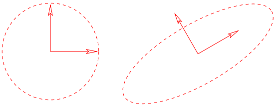
- learns a rotated problem representation (according to the principle axes of the mutation ellipsoid)
  components are independent (only) in the new representation
- learns a new metric according to the scaling of the independent components in the new representation
- conducts a principle component analysis (PCA) of steps $\langle z \rangle_{\text{sel}}$, sequentially in time and space
  eigenvectors of the covariance matrix $\mathbf{C}$ are the principle components / the principle axes of the mutation ellipsoid
- approximates the inverse Hessian on quadratic functions
- is equivalent with an adaptive (general) linear encoding[4]

. . . equations

[4] Hansen 2000, Invariance, Self-Adaptation and Correlated Mutations in Evolution Strategies, PPSN VI

The covariance matrix adaptation

- learns all pairwise dependencies between variables
  off-diagonal entries in the covariance matrix reflect the dependencies



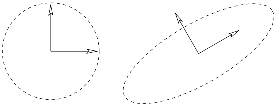
- learns a rotated problem representation (according to the principle axes of the mutation ellipsoid)
  components are independent (only) in the new representation
- learns a new metric according to the scaling of the independent components
  in the new representation
- conducts a principle component analysis (PCA) of steps $\langle z \rangle_{\text{sel}}$, sequentially in time and space
  eigenvectors of the covariance matrix $\mathbf{C}$ are the principle components / the principle axes of the mutation ellipsoid
- approximates the inverse Hessian on quadratic functions
- is equivalent with an adaptive (general) linear encoding[4]

. . . equations

[4] Hansen 2000, Invariance, Self-Adaptation and Correlated Mutations in Evolution Strategies, PPSN VI

The covariance matrix adaptation

- learns all pairwise dependencies between variables
  off-diagonal entries in the covariance matrix reflect the dependencies



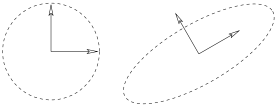
- learns a rotated problem representation (according to the principle axes of the mutation ellipsoid)
  components are independent (only) in the new representation
- learns a new metric according to the scaling of the independent components
  in the new representation
- conducts a principle component analysis (PCA) of steps $\langle z \rangle_{\mathrm{sel}}$, sequentially in time and space
  eigenvectors of the covariance matrix $\mathbf{C}$ are the principle components / the principle axes of the mutation ellipsoid
- approximates the inverse Hessian on quadratic functions
- is equivalent with an adaptive (general) linear encoding[4]

. . . equations

[4] Hansen 2000, Invariance, Self-Adaptation and Correlated Mutations in Evolution Strategies, PPSN VI

The covariance matrix adaptation

- learns all pairwise dependencies between variables
  off-diagonal entries in the covariance matrix reflect the dependencies



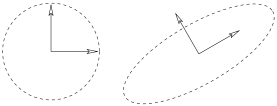
- learns a rotated problem representation (according to the principle axes of the mutation ellipsoid)
  components are independent (only) in the new representation
- learns a new metric according to the scaling of the independent components
  in the new representation
- conducts a principle component analysis (PCA) of steps $\langle z \rangle_{\mathrm{sel}}$, sequentially in time and space
  eigenvectors of the covariance matrix $\mathbf{C}$ are the principle components / the principle axes of the mutation ellipsoid
- approximates the inverse Hessian on quadratic functions
- is equivalent with an adaptive (general) linear encoding[4]

...equations

[4] Hansen 2000, Invariance, Self-Adaptation and Correlated Mutations in Evolution Strategies, PPSN VI

The covariance matrix adaptation

- learns all pairwise dependencies between variables
  off-diagonal entries in the covariance matrix reflect the dependencies



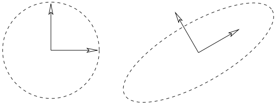
- learns a rotated problem representation (according to the principle axes of the mutation ellipsoid)
  components are independent (only) in the new representation
- learns a new metric according to the scaling of the independent components
  in the new representation
- conducts a principle component analysis (PCA) of steps $\langle z \rangle_{\text{sel}}$, sequentially in time and space
  eigenvectors of the covariance matrix $\mathbf{C}$ are the principle components / the principle axes of the mutation ellipsoid
- approximates the inverse Hessian on quadratic functions
- is equivalent with an adaptive (general) linear encoding[4]

. . . equations

[4] Hansen 2000, Invariance, Self-Adaptation and Correlated Mutations in Evolution Strategies, PPSN VI

The covariance matrix adaptation

- learns all pairwise dependencies between variables
  off-diagonal entries in the covariance matrix reflect the dependencies



- learns a rotated problem representation (according to the principle axes of the mutation ellipsoid)
  components are independent (only) in the new representation
- learns a new metric according to the scaling of the independent components
  in the new representation
- conducts a principle component analysis (PCA) of steps $\langle z \rangle_{\mathrm{sel}}$, sequentially in time and space
  eigenvectors of the covariance matrix $\mathbf{C}$ are the principle components / the principle axes of the mutation ellipsoid
- approximates the inverse Hessian on quadratic functions
- is equivalent with an adaptive (general) linear encoding[4]

. . . equations

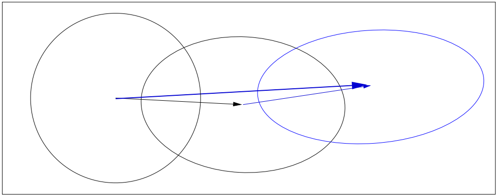
[4] Hansen 2000, Invariance, Self-Adaptation and Correlated Mutations in Evolution Strategies, PPSN VI

Cumulation

The Evolution Path

Evolution Path

Conceptually, the evolution path is the 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 $\langle z \rangle_{\text{sel}}$ is used

$$\boldsymbol{p}_c \propto \sum_{i=0}^{g} \underbrace{(1 - c_c)^{g-i}}_{\substack{\text{exponentially} \\ \text{fading weights}}} \langle z \rangle_{\text{sel}}^{(i)}$$

The recursive construction of the evolution path (cumulation):

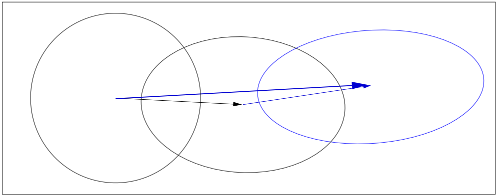
$$\boldsymbol{p}_c \quad \leftarrow \quad \underbrace{(1 - c_c)}_{\text{decay factor}} \boldsymbol{p}_c + \underbrace{\sqrt{1 - (1 - c_c)^2} \sqrt{\mu_{\text{eff}}}}_{\text{normalization factor}} \underbrace{\langle z \rangle_{\text{sel}}}_{\text{input}}$$

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

Cumulation
The Evolution Path

Evolution Path

Conceptually, the evolution path is the 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 $\langle z \rangle_{\mathrm{sel}}$ is used

$$\boldsymbol{p}_{\mathrm{c}} \propto \sum_{i=0}^{g} \underbrace{(1 - c_{\mathbf{c}})^{g-i}}_{\text{exponentially fading weights}} \langle z \rangle_{\mathrm{sel}}^{(i)}$$

The recursive construction of the evolution path (cumulation):

$$\boldsymbol{p}_{\mathrm{c}} \leftarrow \underbrace{(1 - c_{\mathbf{c}})}_{\text{decay factor}} \boldsymbol{p}_{\mathrm{c}} + \underbrace{\sqrt{1-(1-c_{\mathbf{c}})^2}\sqrt{\mu_{\mathrm{eff}}}}_{\text{normalization factor}} \underbrace{\langle z \rangle_{\mathrm{sel}}}_{\text{input}}$$

where $\mu_{\mathrm{eff}} = \frac{1}{\sum {w_i}^2}$, $c_{\mathbf{c}} \ll 1$. History information is accumulated in the evolution path.

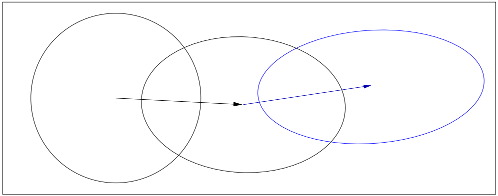
"Cumulation" is a widely used technique and also know as

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

Cumulation

Utilizing the Evolution Path

We used $\langle z\rangle_{\mathrm{sel}}\langle z\rangle_{\mathrm{sel}}^{\mathrm{T}}$ for updating $\mathbf{C}$. Because $\langle z\rangle_{\mathrm{sel}}\langle z\rangle_{\mathrm{sel}}^{\mathrm{T}} = -\langle z\rangle_{\mathrm{sel}}(-\langle z\rangle_{\mathrm{sel}})^{\mathrm{T}}$ the sign of $\langle z\rangle_{\mathrm{sel}}$ is neglected. The sign information is (re-)introduced by using the *evolution path*.



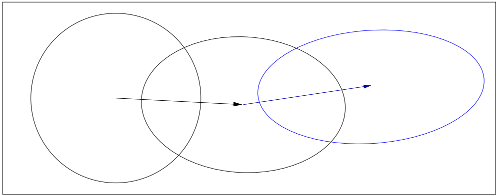
$$p_c \leftarrow \underbrace{(1-c_c)}_{\text{decay factor}} p_c + \underbrace{\sqrt{1-(1-c_c)^2}\sqrt{\mu_{\mathrm{eff}}}}_{\text{normalization factor}} \langle z\rangle_{\mathrm{sel}}$$

where $\mu_{\mathrm{eff}} = \frac{1}{\sum w_i^2}$, $c_c \ll 1$.

Cumulation

Utilizing the Evolution Path

We used $\langle z\rangle_{\mathrm{sel}}\langle z\rangle_{\mathrm{sel}}^{\mathrm{T}}$ for updating $\mathbf{C}$. Because $\langle z\rangle_{\mathrm{sel}}\langle z\rangle_{\mathrm{sel}}^{\mathrm{T}} = -\langle z\rangle_{\mathrm{sel}}(-\langle z\rangle_{\mathrm{sel}})^{\mathrm{T}}$ the sign of $\langle z\rangle_{\mathrm{sel}}$ is neglected. The sign information is (re-)introduced by using the *evolution path*.

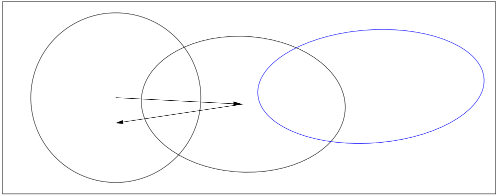


$$\boldsymbol{p}_{\mathrm{c}} \quad \leftarrow \quad \underbrace{(1-c_{\mathrm{c}})}_{\text{decay factor}}\boldsymbol{p}_{\mathrm{c}} + \underbrace{\sqrt{1-(1-c_{\mathrm{c}})^2}\sqrt{\mu_{\mathrm{eff}}}}_{\text{normalization factor}}\langle z\rangle_{\mathrm{sel}}$$

where $\mu_{\mathrm{eff}} = \frac{1}{\sum w_i{}^2}$, $c_{\mathrm{c}} \ll 1$.

Cumulation
Utilizing the Evolution Path

We used $\langle z\rangle_{\text{sel}}\langle z\rangle_{\text{sel}}^{\mathrm{T}}$ for updating $\mathbf{C}$. Because $\langle z\rangle_{\text{sel}}\langle z\rangle_{\text{sel}}^{\mathrm{T}} = -\langle z\rangle_{\text{sel}}(-\langle z\rangle_{\text{sel}})^{\mathrm{T}}$ the sign of $\langle z\rangle_{\text{sel}}$ is neglected. The sign information is (re-)introduced by using the *evolution path*.



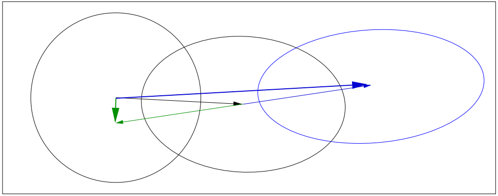
$$p_c \quad \leftarrow \quad \underbrace{(1-c_c)}_{\text{decay factor}}\, p_c + \underbrace{\sqrt{1-(1-c_c)^2}\sqrt{\mu_{\text{eff}}}}_{\text{normalization factor}}\langle z\rangle_{\text{sel}}$$

where $\mu_{\text{eff}} = \frac{1}{\sum w_i{}^2}$, $c_c \ll 1$.

Cumulation

Utilizing the Evolution Path

We used $\langle z\rangle_{\mathrm{sel}}\langle z\rangle_{\mathrm{sel}}^{\mathrm{T}}$ for updating $\mathbf{C}$. Because $\langle z\rangle_{\mathrm{sel}}\langle z\rangle_{\mathrm{sel}}^{\mathrm{T}} = -\langle z\rangle_{\mathrm{sel}}(-\langle z\rangle_{\mathrm{sel}})^{\mathrm{T}}$ the sign of $\langle z\rangle_{\mathrm{sel}}$ is neglected. The sign information is (re-)introduced by using the *evolution path*.



$$p_{\mathrm{c}} \quad \leftarrow \quad \underbrace{(1 - c_{\mathrm{c}})}_{\text{decay factor}} p_{\mathrm{c}} + \underbrace{\sqrt{1-(1-c_{\mathrm{c}})^2}\sqrt{\mu_{\mathrm{eff}}}}_{\text{normalization factor}} \langle z\rangle_{\mathrm{sel}}$$

where $\mu_{\mathrm{eff}} = \frac{1}{\sum {w_i}^2}$, $c_{\mathrm{c}} \ll 1$.

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 $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$.[a]

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

...rank μ update

Rank-μ Update

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

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 matrix

$$\mathbf{Z} = \sum_{i=1}^{\mu} w_i\, \boldsymbol{z}_{i:\lambda} \boldsymbol{z}_{i:\lambda}^{\mathrm{T}}$$

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

The rank-μ update then reads

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

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

Rank-μ Update

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

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 matrix

$$\mathbf{Z} = \sum_{i=1}^{\mu} w_i\, \boldsymbol{z}_{i:\lambda} \boldsymbol{z}_{i:\lambda}^{\mathrm{T}}$$

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

$$\mathbf{C} \leftarrow (1 - c_{\text{cov}})\, \mathbf{C} + c_{\text{cov}}\, \mathbf{Z}$$

where $c_{\text{cov}} \approx \mu_{\text{eff}} / n^2 \leq 1$.

Rank-μ Update

$$\begin{array}{rclrcl} \boldsymbol{x}_i & = & \boldsymbol{m} + \sigma\, \boldsymbol{z}_i, & \boldsymbol{z}_i & \sim & \mathcal{N}_i(\mathbf{0}, \mathbf{C})\,, \\ \boldsymbol{m} & \leftarrow & \boldsymbol{m} + \sigma \langle \boldsymbol{z} \rangle_{\text{sel}} & \langle \boldsymbol{z} \rangle_{\text{sel}} & = & \sum_{i=1}^{\mu} w_i\, \boldsymbol{z}_{i:\lambda} \end{array}$$

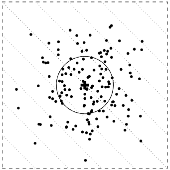
The rank-μ update extends the update rule for large population sizes λ using $\mu > 1$ vectors to update $\mathbf{C}$ at each generation step.
The matrix

$$\mathbf{Z} = \sum_{i=1}^{\mu} w_i\, \boldsymbol{z}_{i:\lambda} \boldsymbol{z}_{i:\lambda}^{\mathrm{T}}$$

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

$$\mathbf{C} \leftarrow (1 - c_{\text{cov}})\, \mathbf{C} + c_{\text{cov}}\, \mathbf{Z}$$

where $c_{\text{cov}} \approx \mu_{\text{eff}} / n^2 \leq 1$.



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

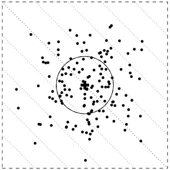
sampling of $\lambda = 150$ solutions where $\mathbf{C} = \mathbf{I}$ and $\sigma = 1$

$$\mathbf{Z} = \frac{1}{\mu} \sum z_{i:\lambda} z_{i:\lambda}^{\mathrm{T}}$$
$$\mathbf{C} \leftarrow (1-1) \times \mathbf{C} + 1 \times \mathbf{Z}$$

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

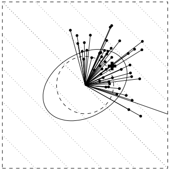
$$m_{\text{new}} \leftarrow \boldsymbol{m} + \frac{1}{\mu} \sum z_{i:\lambda}$$

new distribution



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

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

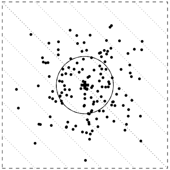


$$\mathbf{Z} = \frac{1}{\mu} \sum \boldsymbol{z}_{i:\lambda} \boldsymbol{z}_{i:\lambda}^{\mathrm{T}}$$
$$\mathbf{C} \leftarrow (1-1) \times \mathbf{C} + 1 \times \mathbf{Z}$$

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

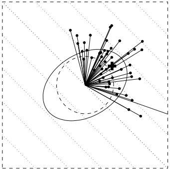
$$\boldsymbol{m}_{\text{new}} \leftarrow \boldsymbol{m} + \frac{1}{\mu} \sum \boldsymbol{z}_{i:\lambda}$$

new distribution



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

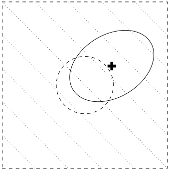
sampling of $\lambda = 150$ solutions where $\mathbf{C} = \mathbf{I}$ and $\sigma = 1$



$\mathbf{Z} \;=\; \frac{1}{\mu} \sum z_{i:\lambda} z_{i:\lambda}^{\mathrm{T}}$

$\mathbf{C} \;\leftarrow\; (1 - 1) \times \mathbf{C} + 1 \times \mathbf{Z}$

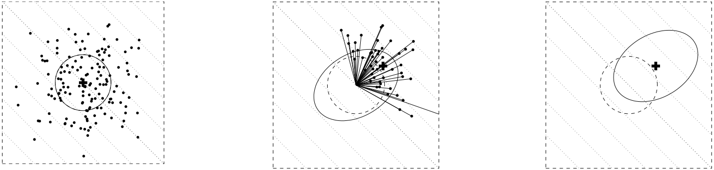
calculating $\mathbf{C}$ where $\mu = 50$, $w_1 = \cdots = w_\mu = \frac{1}{\mu}$, and $c_{\mathrm{cov}} = 1$



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

new distribution

rank-μ CMA versus EMNA$_{\text{global}}$ [5]

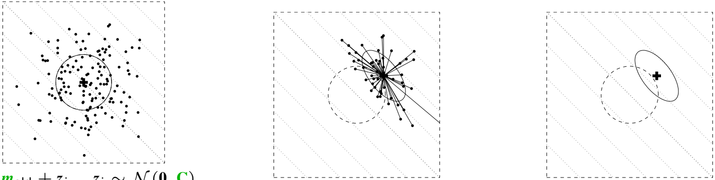


$\boldsymbol{x}_i = \boldsymbol{m}_{\text{old}} + z_i, \quad z_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$

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

$\boldsymbol{m}_{\text{new}} = \boldsymbol{m}_{\text{old}} + \frac{1}{\mu} \sum z_{i:\lambda}$

rank-μ CMA conducts a PCA of steps



$\boldsymbol{x}_i = \boldsymbol{m}_{\text{old}} + z_i, \quad z_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$

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

$\boldsymbol{m}_{\text{new}} = \boldsymbol{m}_{\text{old}} + \frac{1}{\mu} \sum z_{i:\lambda}$

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

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

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

new distribution

The CMA-update yields a larger variance in particular in gradient direction, because $\boldsymbol{m}_{\text{new}}$ is the minimizer for the variances when calculating $\mathbf{C}$

[5] 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_{eff}/n^2
- can reduce the number of necessary generations roughly from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$[6]
  given $\mu_{\text{eff}} \propto \lambda \propto n$

Therefore the rank-μ update is the primary mechanism whenever a large population size is used

say $\lambda \geq 3\,n + 10$

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 $\mathcal{O}(n)$.

[6] 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_{eff}/n^2
- can reduce the number of necessary generations roughly from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$[6]
  given $\mu_{\mathrm{eff}} \propto \lambda \propto n$

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

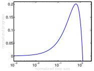
The rank-one update

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

[6] 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

Why Step-Size Control?

1. the covariance matrix update can hardly increase the variance in *all* directions simultaneously

2. There is a relatively small *evolution window* for the step-size .
Given $\mu \not\gg n$ the optimal step length remarkably depends on parent number μ. The $\mathbf{C}$-update cannot achieve close to optimal step lengths for a wide range of μ.

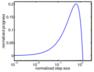
3. The learning rate $c_{\text{cov}} \approx \mu_{\text{eff}}/n^2$ does not comply with the requirements of convergence speed on the sphere model, $f(\boldsymbol{x}) = \sum x_i^2$.

Each single reason would be sufficient to ask for additional step-size control

…methods for step-size control

Why Step-Size Control?

1. the covariance matrix update can hardly increase the variance in *all* directions simultaneously

2. There is a relatively small *evolution window* for the step-size 

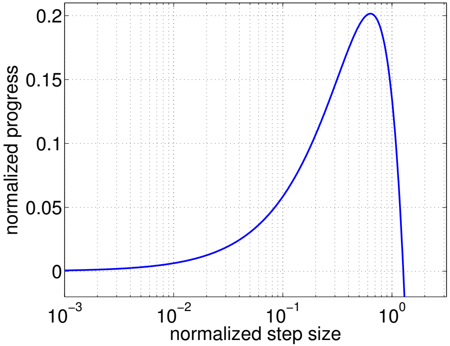
.
Given $\mu \not\gg n$ the optimal step length remarkably depends on parent number μ. The $\mathbf{C}$-update cannot achieve close to optimal step lengths for a wide range of μ.

3. The learning rate $c_{\mathrm{cov}} \approx \mu_{\mathrm{eff}}/n^2$ does not comply with the requirements of convergence speed on the sphere model, $f(\boldsymbol{x}) = \sum x_i^2$.

Each single reason would be sufficient to ask for additional step-size control

…methods for step-size control

Why Step-Size Control?

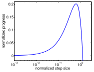


evolution window for the step-size on the sphere function

evolution window refers to the step-size interval where reasonable performance is observed

Why Step-Size Control?

1. the covariance matrix update can hardly increase the variance in *all* directions simultaneously

2. There is a relatively small *evolution window* for the step-size 

. Given $\mu \not\gg n$ the optimal step length remarkably depends on parent number μ. The $\mathbf{C}$-update cannot achieve close to optimal step lengths for a wide range of μ.

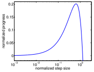
3. The learning rate $c_{\mathrm{cov}} \approx \mu_{\mathrm{eff}}/n^2$ does not comply with the requirements of convergence speed on the sphere model, $f(\boldsymbol{x}) = \sum x_i^2$.

Each single reason would be sufficient to ask for additional step-size control

...methods for step-size control

Why Step-Size Control?

1. the covariance matrix update can hardly increase the variance in *all* directions simultaneously

2. There is a relatively small *evolution window* for the step-size 

. Given $\mu \not\gg n$ the optimal step length remarkably depends on parent number μ. The $\mathbf{C}$-update cannot achieve close to optimal step lengths for a wide range of μ.

3. The learning rate $c_{\mathrm{cov}} \approx \mu_{\mathrm{eff}}/n^2$ does not comply with the requirements of convergence speed on the sphere model, $f(\boldsymbol{x}) = \sum x_i^2$.

Each single reason would be sufficient to ask for additional step-size control

. . . methods for step-size control

Methods for Step-Size Control

- $1/5$-th success rule[a][b], often applied with "+"-selection
- σ-self-adaptation[c], applied with ","-selection
- two-point adaptation, used in Evolutionary Gradient Search[d]
- path length control[e] (Cumulative Step-size Adaptation, CSA)[f], applied with ","-selection

---

[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] Salomon 1998, Evolutionary algorithms and gradient search: Similarities and differences, *IEEE Trans. Evol. Comput., 2(2)*

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

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

Methods for Step-Size Control

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

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

- σ-self-adaptation, applied with ","-selection
- two-point adaptation, used in Evolutionary Gradient Search
- path length control (Cumulative Step-size Adaptation, CSA), applied with ","-selection

Methods for Step-Size Control

- $1/5$-th success rule, often applied with "+"-selection
- σ-self-adaptation, applied with ","-selection

  mutation is applied to the step-size and the better one, according to the objective function value, is selected
- two-point adaptation, used in Evolutionary Gradient Search
- path length control (Cumulative Step-size Adaptation, CSA), applied with ","-selection

Methods for Step-Size Control

- $1/5$-th success rule, often applied with "+"-selection
- σ-self-adaptation, applied with ","-selection
- two-point adaptation, used in Evolutionary Gradient Search

  simplified "global" self-adaptation

- path length control (Cumulative Step-size Adaptation, CSA), applied with ","-selection

Methods for Step-Size Control

- $1/5$-th success rule[a][b], often applied with “+”-selection
- σ-self-adaptation[c], applied with “,”-selection
- two-point adaptation, used in Evolutionary Gradient Search[d]
- path length control[e] (Cumulative Step-size Adaptation, CSA)[f], applied with “,”-selection

---

[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] Salomon 1998, Evolutionary algorithms and gradient search: Similarities and differences, *IEEE Trans. Evol. Comput., 2(2)*

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

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

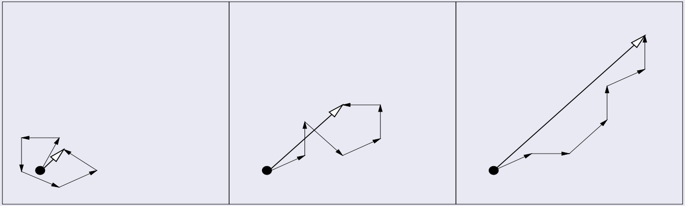
Path Length Control

The Concept

$$x_i = m + \sigma z_i$$
$$m \leftarrow m + \sigma \langle z \rangle_{\mathrm{sel}}$$

Measure the length of the *evolution path*

the pathway of the mean vector m in the generation sequence



$\downarrow$ decrease σ $\qquad\qquad$ $\downarrow$ increase σ

loosely speaking steps are

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

Summary

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) in a Nutshell

1. Multivariate normal distribution to generate new search points
   follows the maximum entropy principle
2. Selection only based on the ranking of the f-values, weighted recombination
   using only the ranking of f-values preserves invariance
3. *Covariance matrix adaptation (CMA)* increases the probability to repeat successful steps
   conducts a sequential PCA
   $\Longrightarrow$ rotated problem representation
   $\Longrightarrow$ learning all pairwise dependencies
4. An evolution path enhances the covariance matrix adaptation
5. *Path length control* to control the step-size
   uses the evolution path,
   aims at conjugate perpendicularity

Summary

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) in a Nutshell

1. Multivariate normal distribution to generate new search points
   follows the maximum entropy principle
2. Selection only based on the ranking of the f-values, weighted recombination
   using only the ranking of f-values preserves invariance
3. *Covariance matrix adaptation (CMA)* increases the probability to repeat successful steps
   conducts a sequential PCA
   $\Longrightarrow$ rotated problem representation
   $\Longrightarrow$ learning all pairwise dependencies
4. An evolution path enhances the covariance matrix adaptation
5. *Path length control* to control the step-size
   uses the evolution path,
   aims at conjugate perpendicularity

Summary

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) in a Nutshell

1. Multivariate normal distribution to generate new search points

   follows the maximum entropy principle

2. Selection only based on the ranking of the f-values, weighted recombination

   using only the ranking of f-values preserves invariance

3. *Covariance matrix adaptation (CMA)* increases the probability to repeat successful steps

   conducts a sequential PCA
   $\Longrightarrow$ rotated problem representation
   $\Longrightarrow$ learning all pairwise dependencies

4. An evolution path enhances the covariance matrix adaptation

5. *Path length control* to control the step-size

   uses the evolution path,
   aims at conjugate perpendicularity

Summary

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) in a Nutshell

1. Multivariate normal distribution to generate new search points

   follows the maximum entropy principle

2. Selection only based on the ranking of the f-values, weighted recombination

   using only the ranking of f-values preserves invariance

3. *Covariance matrix adaptation (CMA)* increases the probability to repeat successful steps

   conducts a sequential PCA
   $\Longrightarrow$ rotated problem representation
   $\Longrightarrow$ learning all pairwise dependencies

4. An evolution path enhances the covariance matrix adaptation

5. *Path length control* to control the step-size

   uses the evolution path,
   aims at conjugate perpendicularity

Summary

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) in a Nutshell

1. Multivariate normal distribution to generate new search points
   follows the maximum entropy principle
2. Selection only based on the ranking of the f-values, weighted recombination
   using only the ranking of f-values preserves invariance
3. *Covariance matrix adaptation (CMA)* increases the probability to repeat successful steps
   conducts a sequential PCA
   $\Longrightarrow$ rotated problem representation
   $\Longrightarrow$ learning all pairwise dependencies
4. An evolution path enhances the covariance matrix adaptation
5. *Path length control* to control the step-size
   uses the evolution path,
   aims at conjugate perpendicularity

Summary of Equations

The Covariance Matrix Adaptation Evolution Strategy

Initialize $\boldsymbol{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, $\mathbf{C} = \mathbf{I}$, and $\boldsymbol{p}_\mathrm{c} = \mathbf{0}$, $\boldsymbol{p}_\sigma = \mathbf{0}$,
set $c_\mathbf{c} \approx 4/n$, $c_\sigma \approx 4/n$, $c_\mathrm{cov} \approx \mu_\mathrm{eff}/n^2$, $\mu_\mathrm{cov} = \mu_\mathrm{eff}$, $d_\sigma \approx 1 + \sqrt{\frac{\mu_\mathrm{eff}}{n}}$,
set λ and $w_i, i = 1, \ldots, \mu$ such that $\mu_\mathrm{eff} \approx 0.3\,\lambda$

While not terminate

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

$$\boldsymbol{m} \leftarrow \boldsymbol{m} + \sigma\langle\boldsymbol{z}\rangle_\mathrm{sel} \quad \text{where } \langle\boldsymbol{z}\rangle_\mathrm{sel} = \sum_{i=1}^{\mu} w_i\,\boldsymbol{z}_{i:\lambda} \qquad \text{update mean}$$

$$\boldsymbol{p}_\mathrm{c} \leftarrow (1 - c_\mathbf{c})\,\boldsymbol{p}_\mathrm{c} + \mathbb{1}_{\{\|\boldsymbol{p}_\sigma\| < 1.5\sqrt{n}\}}\sqrt{1 - (1 - c_\mathbf{c})^2}\sqrt{\mu_\mathrm{eff}}\,\langle\boldsymbol{z}\rangle_\mathrm{sel} \qquad \text{cumulation for } \mathbf{C}$$

$$\mathbf{C} \leftarrow (1 - c_\mathrm{cov})\,\mathbf{C} + c_\mathrm{cov}\,\frac{1}{\mu_\mathrm{cov}}\,\boldsymbol{p}_\mathrm{c}\boldsymbol{p}_\mathrm{c}^\mathrm{T} \qquad \text{update } \mathbf{C}$$

$$+ c_\mathrm{cov}\left(1 - \frac{1}{\mu_\mathrm{cov}}\right)\mathbf{Z} \quad \text{where } \mathbf{Z} = \sum_{i=1}^{\mu} w_i\,\boldsymbol{z}_{i:\lambda}\boldsymbol{z}_{i:\lambda}^\mathrm{T}$$

$$\boldsymbol{p}_\sigma \leftarrow (1 - c_\sigma)\,\boldsymbol{p}_\sigma + \sqrt{1 - (1 - c_\sigma)^2}\sqrt{\mu_\mathrm{eff}}\,\mathbf{C}^{-\frac{1}{2}}\langle\boldsymbol{z}\rangle_\mathrm{sel} \qquad \text{cumulation for } \sigma$$

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

. . . CMA in a nutshell

Experimentum Crucis

What did we specifically want to achieve?

- reduce any convex quadratic function

$$f(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x}$$

to the sphere model

$$f(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}$$

without use of derivatives

- lines of equal density align with lines of equal fitness

$$\mathbf{C} \propto \boldsymbol{H}^{-1}$$

- even true for any $g(f(\boldsymbol{x})) = g\left(\boldsymbol{x}^{\mathrm{T}}\mathbf{H}\boldsymbol{x}\right)$

$g : \mathbb{R} \to \mathbb{R}$ stricly monotonic (order preserving)

Experimentum Crucis

What did we specifically want to achieve?

- reduce any convex quadratic function

$$f(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x}$$

to the sphere model

$$f(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}$$

without use of derivatives

- lines of equal density align with lines of equal fitness

$$\mathbf{C} \propto \boldsymbol{H}^{-1}$$

- even true for any $g(f(\boldsymbol{x})) = g\left(\boldsymbol{x}^{\mathrm{T}}\mathbf{H}\boldsymbol{x}\right)$

$g : \mathbb{R} \to \mathbb{R}$ stricly monotonic (order preserving)

Experimentum Crucis

What did we specifically want to achieve?

- reduce any convex quadratic function

$$f(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x}$$

to the sphere model

$$f(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}$$

without use of derivatives

- lines of equal density align with lines of equal fitness

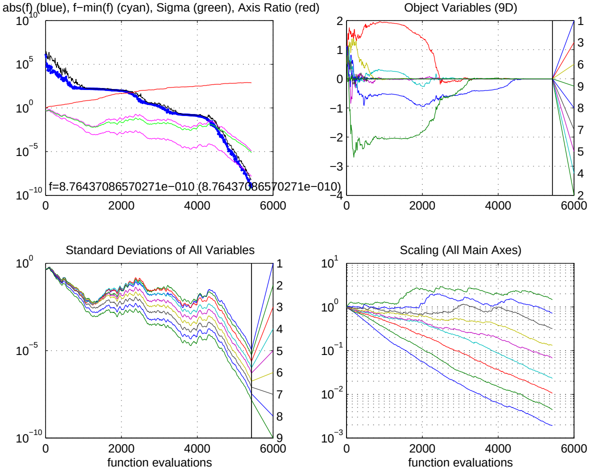
$$\mathbf{C} \propto \boldsymbol{H}^{-1}$$

- even true for any $g(f(\boldsymbol{x})) = g\left(\boldsymbol{x}^{\mathrm{T}}\mathbf{H}\boldsymbol{x}\right)$

$g : \mathbb{R} \to \mathbb{R}$ stricly monotonic (order preserving)

Experimentum Crucis (1)

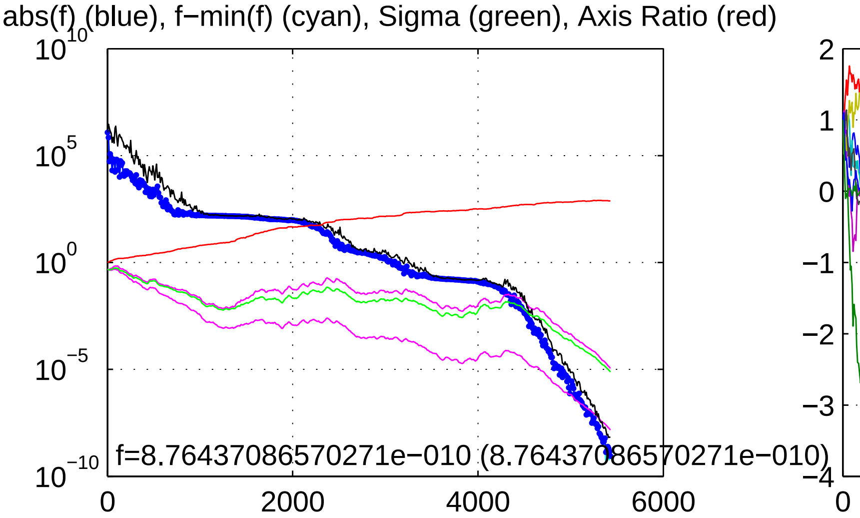
f convex quadratic, separable



$$f(\boldsymbol{x}) = \sum_{i=1}^{n} 10^{6\frac{i-1}{n-1}} x_i^2$$

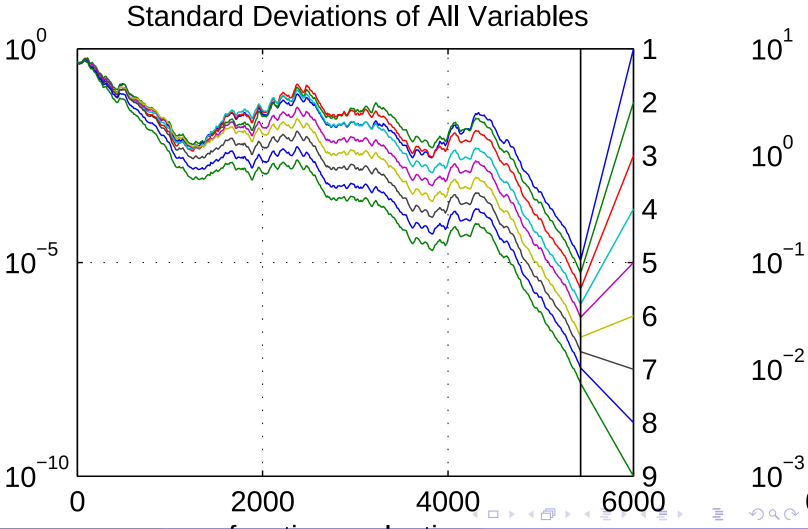
Experimentum Crucis (1)

f convex quadratic, separable



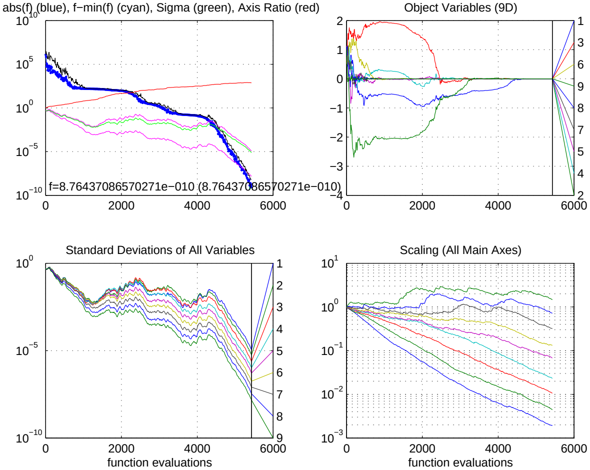
Experimentum Crucis (1)

f convex quadratic, separable



Experimentum Crucis (1)

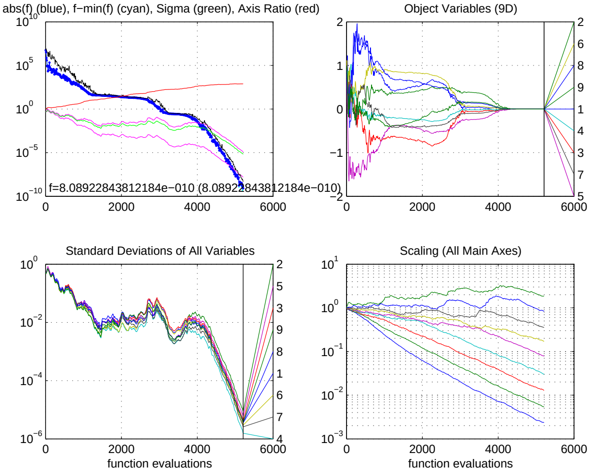
f convex quadratic, separable



$$f(\boldsymbol{x}) = \sum_{i=1}^{n} 10^{6\frac{i-1}{n-1}} x_i^2$$

Experimentum Crucis (2)

f convex quadratic, non-separable (rotated)



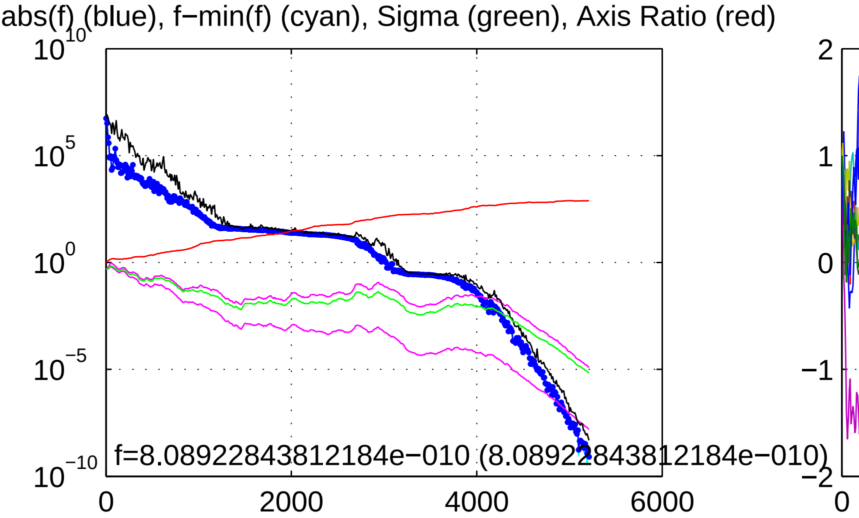
$\mathbf{C} \propto \boldsymbol{H}^{-1}$ for all $g, \mathbf{H}$

$f(\boldsymbol{x}) = g\left(\boldsymbol{x}^{\mathrm{T}}\mathbf{H}\boldsymbol{x}\right),\ g : \mathbb{R} \to \mathbb{R}$ stricly monotonic

... internal parameters

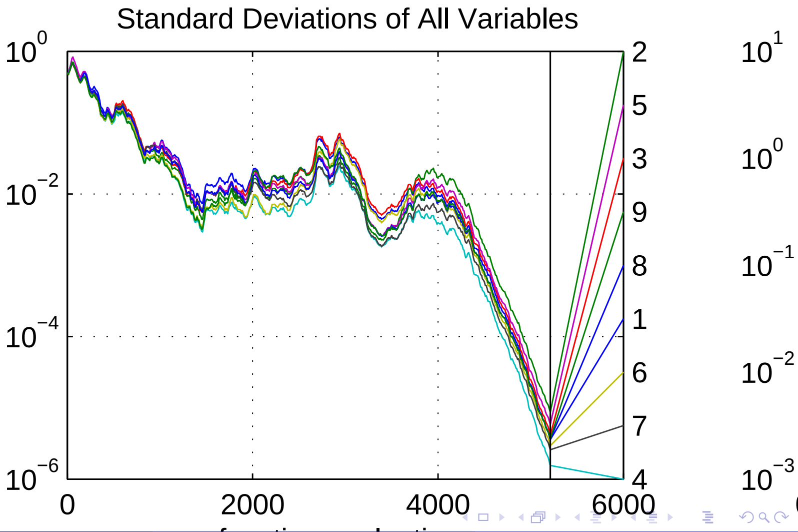
Experimentum Crucis (2)

f convex quadratic, non-separable (rotated)



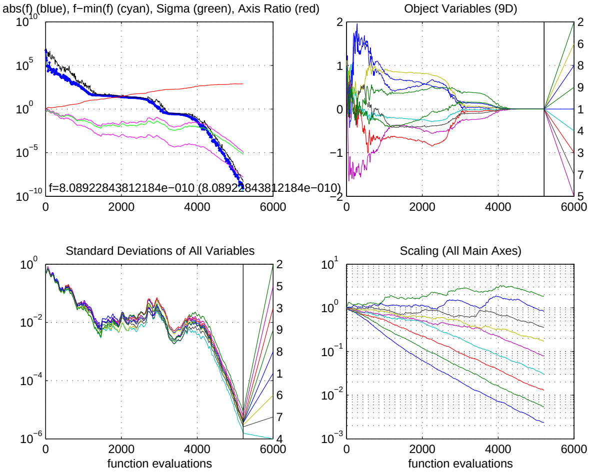
Experimentum Crucis (2)

f convex quadratic, non-separable (rotated)



Experimentum Crucis (2)

f convex quadratic, non-separable (rotated)



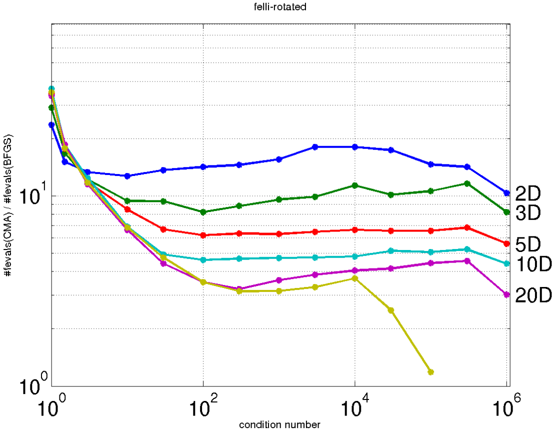
$\mathbf{C} \propto \boldsymbol{H}^{-1}$ for all $g, \mathbf{H}$

$f(\boldsymbol{x}) = g\left(\boldsymbol{x}^{\mathrm{T}}\mathbf{H}\boldsymbol{x}\right)$, $g : \mathbb{R} \to \mathbb{R}$ stricly monotonic

... internal parameters

Comparison to BFGS

f convex quadratic, non-separable (rotated)



$$f(\boldsymbol{x}) = \boldsymbol{x}^{\mathrm{T}}\mathbf{H}\boldsymbol{x}$$

shown are $\frac{\text{function evaluations CMA-ES}}{\text{function evaluations BFGS}}$ until to reach $f = 10^{-6}$ versus condition number

...population size, invariance

Invariance

Motivation

The grand aim of all science is to cover the greatest number of empirical facts by logical deduction from the smallest number of hypotheses or axioms.

— Albert Einstein

- empirical performance results, for example
  - from benchmark functions,
  - from solved real world problems,

  are only useful if they do generalize to other problems

- Invariance is a statement about the feasibility of generalization

  generalizes performance from a single function to a class of functions

Invariance

Motivation

The grand aim of all science is to cover the greatest number of empirical facts by logical deduction from the smallest number of hypotheses or axioms.

— Albert Einstein

- empirical performance results, for example
  - from benchmark functions,
  - from solved real world problems,

  are only useful if they do generalize to other problems

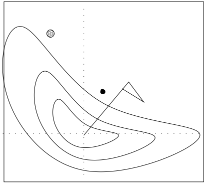
- Invariance is a statement about the feasibility of generalization

  generalizes performance from a single function to a class of functions

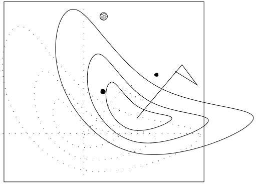
Basic Invariance in Search Space

translation invariance, for example

applies to most optimization algorithms



$$f(\boldsymbol{x}) \leftrightarrow f(\boldsymbol{x}-\boldsymbol{a})$$



Identical behavior on f and f_a

$$\begin{aligned} f: \quad & \boldsymbol{x} \mapsto f(\boldsymbol{x}), & \boldsymbol{x}^{(t=0)} &= \boldsymbol{x}_0 \\ f_a: \quad & \boldsymbol{x} \mapsto f(\boldsymbol{x}-\boldsymbol{a}), & \boldsymbol{x}^{(t=0)} &= \boldsymbol{x}_0 + \boldsymbol{a} \end{aligned}$$

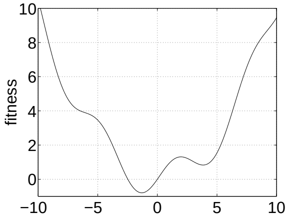
No difference can be observed w.r.t. the argument of f

Only useful if the initial point is not decisive

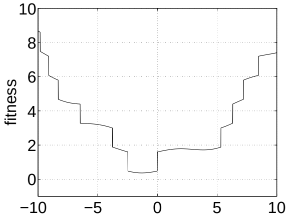
Invariance in Function Space

invariance to order preserving transformations

preserved by ranking based selection



$f(\boldsymbol{x}) \leftrightarrow g(f(\boldsymbol{x}))$



Identical behavior on f and $g \circ f$ for all order preserving $g : \mathbb{R} \to \mathbb{R}$ (strictly monotonically increasing g)

$$\begin{aligned} f: \quad & \boldsymbol{x} \mapsto f(\boldsymbol{x}), & \boldsymbol{x}^{(t=0)} = \boldsymbol{x}_0 \\ g \circ f: \quad & \boldsymbol{x} \mapsto g(f(\boldsymbol{x})), & \boldsymbol{x}^{(t=0)} = \boldsymbol{x}_0 \end{aligned}$$

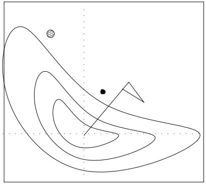
No difference can be observed w.r.t. the argument of f

Rotational Invariance in Search Space

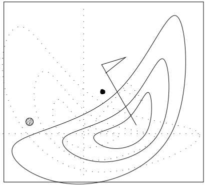
invariance to an orthogonal transformation $\mathbf{R}$, where $\mathbf{R}\mathbf{R}^{\mathrm{T}} = \mathbf{I}$

e.g. true for simple evolution strategies

recombination operators might jeopardize rotational invariance



$$f(\boldsymbol{x}) \leftrightarrow f(\mathbf{R}\boldsymbol{x})$$



Identical behavior on f and $f_{\mathbf{R}}$

$$\begin{aligned} f: \quad & \boldsymbol{x} \mapsto f(\boldsymbol{x}), & \boldsymbol{x}^{(t=0)} &= \boldsymbol{x}_0 \\ f_{\mathbf{R}}: \quad & \boldsymbol{x} \mapsto f(\mathbf{R}\boldsymbol{x}), & \boldsymbol{x}^{(t=0)} &= \mathbf{R}^{-1}(\boldsymbol{x}_0) \end{aligned}$$

No difference can be observed w.r.t. the argument of f

Invariances in Search Space

- invariance to any rigid (scalar product preserving) transformation in search space $\boldsymbol{x} \mapsto \mathbf{R}\boldsymbol{x} - \boldsymbol{a}$, where $\mathbf{R}\mathbf{R}^{\mathrm{T}} = \mathbf{I}$

e.g. true for simple evolution strategies

Identical behavior on f and $f_{\mathbf{R}}$

$$\begin{aligned} f: &\quad \boldsymbol{x} \mapsto f(\boldsymbol{x}), & \boldsymbol{x}^{(t=0)} &= \boldsymbol{x}_0 \\ f_{\mathbf{R}}: &\quad \boldsymbol{x} \mapsto f(\mathbf{R}\boldsymbol{x}), & \boldsymbol{x}^{(t=0)} &= \mathbf{R}^{-1}(\boldsymbol{x}_0) \end{aligned}$$

No difference can be observed w.r.t. the argument of f

Invariances in Search Space

- invariance to any rigid (scalar product preserving) transformation in search space $\boldsymbol{x} \mapsto \mathbf{R}\boldsymbol{x} - \boldsymbol{a}$, where $\mathbf{R}\mathbf{R}^{\mathrm{T}} = \mathbf{I}$

  e.g. true for simple evolution strategies

- scale invariance (scalar multiplication)

  exploited by step-size control

Identical behavior on f and f_α

$$\begin{aligned} f: \quad & \boldsymbol{x} \mapsto f(\boldsymbol{x}), & \boldsymbol{x}^{(t=0)} &= \boldsymbol{x}_0, & \sigma^{(t=0)} &= \sigma_0 \\ f_\alpha: \quad & \boldsymbol{x} \mapsto f(\alpha\boldsymbol{x}), & \boldsymbol{x}^{(t=0)} &= \boldsymbol{x}_0/\alpha, & \sigma^{(t=0)} &= \sigma_0/\alpha \end{aligned}$$

No difference can be observed w.r.t. the argument of f

Only useful with an effective step-size control

Invariances in Search Space

- invariance to any rigid (scalar product preserving) transformation in search space $\boldsymbol{x} \mapsto \mathbf{R}\boldsymbol{x} - \boldsymbol{a}$, where $\mathbf{R}\mathbf{R}^{\mathrm{T}} = \mathbf{I}$
  e.g. true for simple evolution strategies
- scale invariance (scalar multiplication)
  exploited by step-size control
- invariance to a general linear transformation $\mathbf{G}$
  exploited by CMA

Identical behavior on f and $f_{\mathbf{G}}$

$$\begin{array}{lll} f: \quad \boldsymbol{x} \mapsto f(\boldsymbol{x}), & \boldsymbol{x}^{(t=0)} = \boldsymbol{x}_0, & \mathbf{C}^{(t=0)} = \mathbf{I} \\ f_{\mathbf{G}}: \quad \boldsymbol{x} \mapsto f(\mathbf{G}(\boldsymbol{x} - \boldsymbol{b})), & \boldsymbol{x}^{(t=0)} = \mathbf{G}^{-1}\boldsymbol{x}_0 + \boldsymbol{b}, & \mathbf{C}^{(t=0)} = \mathbf{G}^{-1}{\mathbf{G}^{-1}}^{\mathrm{T}} \end{array}$$

No difference can be observed w.r.t. the argument of f

Only useful with an effective adaptation of $\mathbf{C}$

Invariance of the CMA Evolution Strategy

- The CMA Evolution Strategy inherits all invariances from simple evolution strategies

  to *rigid transformations* of the search space and
  to *order preserving transformations* of the function value

- The Covariance Matrix Adaptation adds invariance to general linear transformations

  useful *only together* with an effective adaptation of the covariance matrix

. . . strategy internal parameters

Invariance of the CMA Evolution Strategy

- The CMA Evolution Strategy inherits all invariances from simple evolution strategies

  to *rigid transformations* of the search space and
  to *order preserving transformations* of the function value

- The Covariance Matrix Adaptation adds invariance to general linear transformations

  useful *only together* with an effective adaptation of the covariance matrix

. . . strategy internal parameters

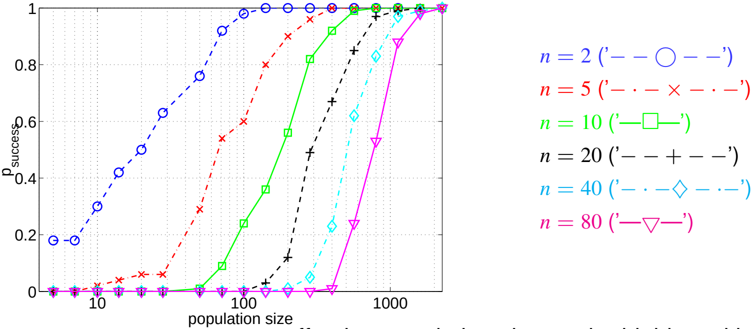
Strategy Internal Parameters

- related to selection and recombination
  - λ, offspring number, new solutions sampled, population size
  - μ, parent number, solutions involved in updates of $\boldsymbol{m}$, $\mathbf{C}$, and σ
  - $w_{i=1,\dots,\mu}$, recombination weights
- related to $\mathbf{C}$-update
  - c_{cov}, learning rate for $\mathbf{C}$-update
  - $c_{\mathbf{c}}$, learning rate for the evolution path
  - μ_{cov}, weight for rank-μ update versus rank-one update
- related to σ-update
  - c_σ, learning rate of the evolution path
  - d_σ, damping for σ-change

Parameters were identified in carefully chosen experimental set ups. Parameters do not in the first place depend on the objective function and are not meant to be in the users choice.
Only(?) the population size λ might be reasonably varied in a wide range, *depending on the objective function*

Population Size on Multi-Modal Functions

Success Probability to Find the Global Optimum



Shown: success rate versus offspring population size on the highly multi-modal Rastrigins function[7]

On multi-modal functions increasing the population size can sharply increase the success probability to find the global optimum

[7] Hansen & Kern 2004. Evaluating the CMA Evolution Strategy on Multimodal Test Functions. PPSN VIII, Springer-Verlag, pp. 282-291.

Multi-Start With Increasing Population Size

Increase by a Factor of Two Each Restart

1. no performance loss, where small population size is sufficient (e.g. on unimodal functions)
2. moderate performance loss, if large population size is necessary
   loss has, in principle, an upper bound

This results in a quasi parameter free search algorithm.[8]

...empirical evaluation

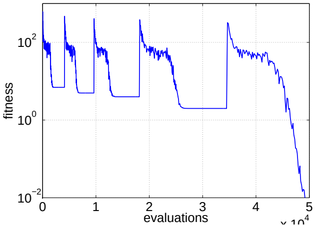
[8] Auger & Hansen 2005. A Restart CMA Evolution Strategy With Increasing Population Size. IEEE Congress on Evolutionary Computation.

Multi-Start With Increasing Population Size

Increase by a Factor of Two Each Restart

1. no performance loss, where small population size is sufficient (e.g. on unimodal functions)
2. moderate performance loss, if large population size is necessary

loss has, in principle, an upper bound



for a factor between successive runs of ≥ 1.5 we have a performance loss smaller than $\sum_{k=0}^{\infty} 1/1.5^k = 3$

This results in a quasi parameter free search algorithm.[8]

...empirical evaluation

[8] Auger & Hansen 2005. A Restart CMA Evolution Strategy With Increasing Population Size. IEEE Congress on Evolutionary Computation.

Performance Evaluation

Evaluation of the performance of a search algorithm needs

- meaningful quantitative measure on benchmark functions or real world problems
- acknowlegde invariance properties
- account for meta-parameter tuning
- account for algorithm internal cost

  often negligible, depending on the objective function cost

Comparison of 11 Evolutionary Algorithms

A Performance Meta-Study

- Task: black-box optimization of 25 benchmark functions and submission of results to the *Congress of Evolutionary Computation*
- Performance measure: cost (number of function evaluations) to reach the target function value, where the maximum number of function evaluations was $\mathrm{FE}_{\mathrm{max}} = \begin{cases} 10^5 & \text{for } n = 10 \\ 3 \times 10^5 & \text{for } n = 30 \end{cases}$

  Remark: the setting of $\mathrm{FE}_{\mathrm{max}}$ has a remarkable influence on the results, if the target function value can be reached only for a (slightly) larger number of function evaluations with a high probability. Where $\mathrm{FEs} \geq \mathrm{FE}_{\mathrm{max}}$ the result must be taken with great care.
- The competitors included Differential Evolution (DE), Particle Swarm Optimization (PSO), real-coded GAs, Estimation of Distribution Algorithm (EDA), and hybrid methods combined e.g. with quasi-Newton BFGS.

Comparison of 11 Evolutionary Algorithms

A Performance Meta-Study

- Task: black-box optimization of 25 benchmark functions and submission of results to the *Congress of Evolutionary Computation*
- Performance measure: cost (number of function evaluations) to reach the target function value, where the maximum number of function evaluations was $\mathtt{FE}_{\text{max}} = \begin{cases} 10^5 & \text{for } n = 10 \\ 3 \times 10^5 & \text{for } n = 30 \end{cases}$

  Remark: the setting of $\mathtt{FE}_{\text{max}}$ has a remarkable influence on the results, if the target function value can be reached only for a (slightly) larger number of function evaluations with a high probability.
  Where $\mathtt{FEs} \geq \mathtt{FE}_{\text{max}}$ the result must be taken with great care.
- The competitors included Differential Evolution (DE), Particle Swarm Optimization (PSO), real-coded GAs, Estimation of Distribution Algorithm (EDA), and hybrid methods combined e.g. with quasi-Newton BFGS.

Comparison of 11 Evolutionary Algorithms

A Performance Meta-Study

- Task: black-box optimization of 25 benchmark functions and submission of results to the *Congress of Evolutionary Computation*
- Performance measure: cost (number of function evaluations) to reach the target function value, where the maximum number of function evaluations was $\mathtt{FE}_{\text{max}} = \begin{cases} 10^5 & \text{for } n = 10 \\ 3 \times 10^5 & \text{for } n = 30 \end{cases}$

  Remark: the setting of $\mathtt{FE}_{\text{max}}$ has a remarkable influence on the results, if the target function value can be reached only for a (slightly) larger number of function evaluations with a high probability. Where $\mathtt{FEs} \geq \mathtt{FE}_{\text{max}}$ the result must be taken with great care.
- The competitors included Differential Evolution (DE), Particle Swarm Optimization (PSO), real-coded GAs, Estimation of Distribution Algorithm (EDA), and hybrid methods combined e.g. with quasi-Newton BFGS.

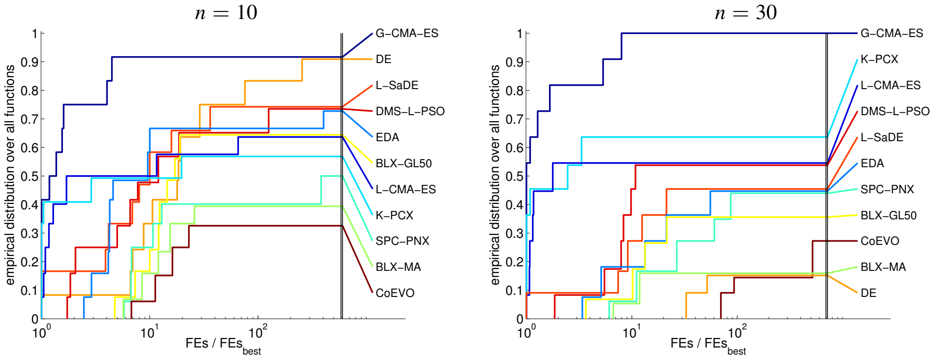
References to Algorithms

BLX-GL50	García-Martínez and Lozano (Hybrid Real-Coded. . .)
BLX-MA	Molina et al. (Adaptive Local Search. . .)
CoEVO	Pošík (Real-Parameter Optimization. . .)
DE	Rönkkönen et al. (Real-Parameter Optimization. . .)
DMS-L-PSO	Liang and Suganthan (Dynamic Multi-Swarm. . .)
EDA	Yuan and Gallagher (Experimental Results. . .)
G-CMA-ES	Auger and Hansen (A Restart CMA. . .)
K-PCX	Sinha et al. (A Population-Based,. . .)
L-CMA-ES	Auger and Hansen (Performance Evaluation. . .)
L-SaDE	Qin and Suganthan (Self-Adaptive Differential. . .)
SPC-PNX	Ballester et al. (Real-Parameter Optimization. . .)

In: CEC 2005 IEEE Congress on Evolutionary Computation, Proceedings

Summarized Results

Empirical Distribution of Normalized Success Performance



$\texttt{FEs} = \texttt{mean}(\#fevals) \times \frac{\#\text{all runs (25)}}{\#\text{successful runs}}$, where $\#fevals$ includes only successful runs.

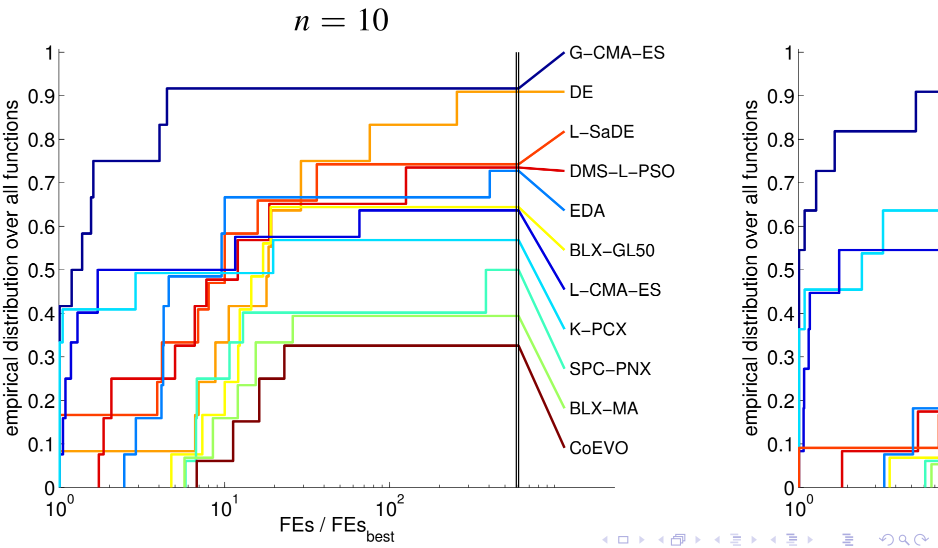
Shown: **empirical distribution function** of the Success Performance $\texttt{FEs}$ divided by $\texttt{FEs}$ of the best algorithm on the respective function.

Results of all functions are used where at least one algorithm was successful at least once, i.e. where the target function value was reached in at least one experiment (out of 11×25 experiments).

Small values for $\texttt{FEs}$ and therefore large (cumulative frequency) values in the graphs are preferable.

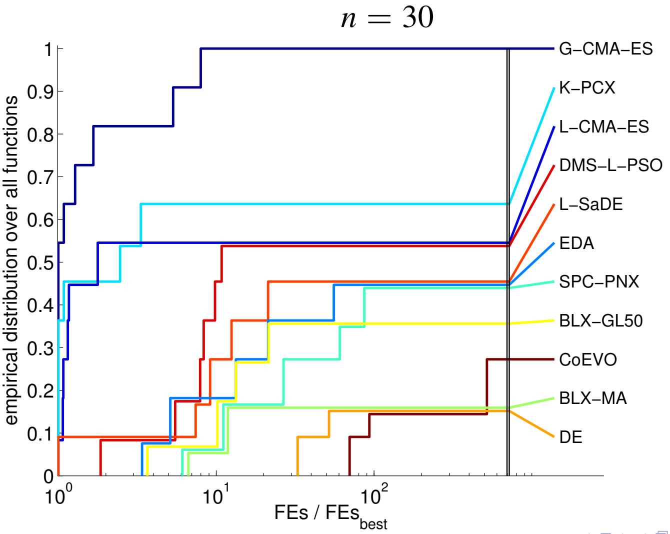
Summarized Results

Empirical Distribution of Normalized Success Performance



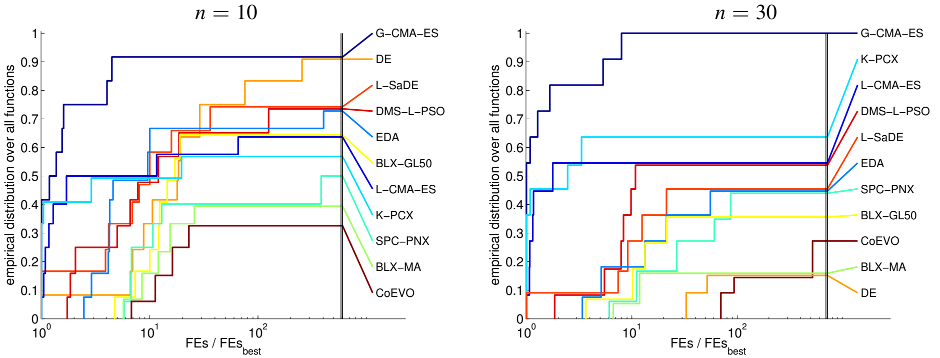
Summarized Results

Empirical Distribution of Normalized Success Performance



Summarized Results

Empirical Distribution of Normalized Success Performance



$\mathtt{FEs} = \mathtt{mean}(\#\mathit{fevals}) \times \frac{\#\text{all runs (25)}}{\#\text{successful runs}}$, where $\#\mathit{fevals}$ includes only successful runs.

Shown: **empirical distribution function** of the Success Performance FEs divided by FEs of the best algorithm on the respective function.

Results of all functions are used where at least one algorithm was successful at least once, i.e. where the target function value was reached in at least one experiment (out of 11×25 experiments).

Small values for FEs and therefore large (cumulative frequency) values in the graphs are preferable.

Function Sets

We split the function set into three subsets

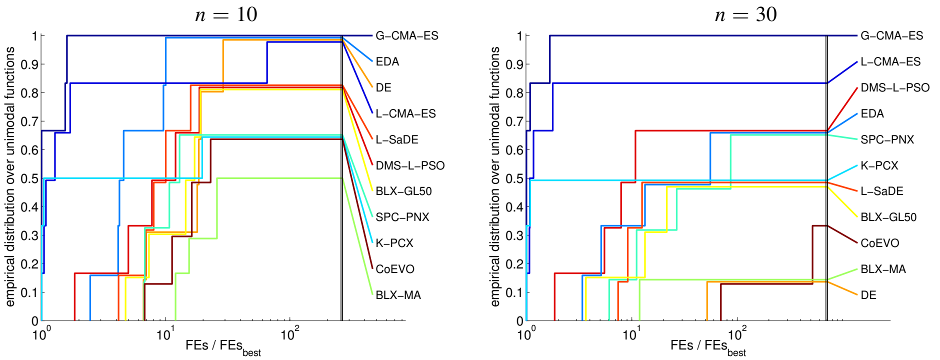
- unimodal functions
- solved multimodal functions

  at least one algorithm conducted at least one successful run
- unsolved multimodal functions

  no single run was successful for any algorithm

Unimodal Functions

Empirical Distribution of Normalized Success Performance



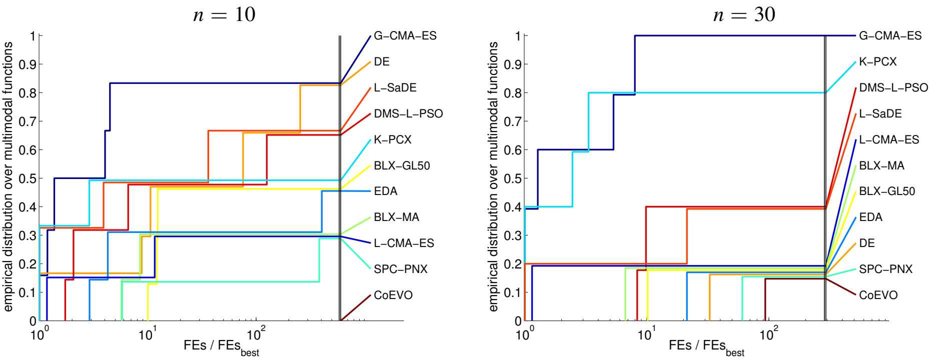
Empirical distribution function of the Success Performance FEs divided by FEs of the best algorithm (table entries of last slides).

$\texttt{FEs} = \texttt{mean}(\#fevals) \times \frac{\#\text{all runs (25)}}{\#\text{successful runs}}$, where $\#fevals$ includes only successful runs.

Small values of FEs and therefore large values in the empirical distribution graphs are preferable.

Multimodal Functions

Empirical Distribution of Normalized Success Performance



Empirical distribution function of the Success Performance $\texttt{FEs}$ divided by $\texttt{FEs}$ of the best algorithm (table entries of last slides).

$\texttt{FEs} = \texttt{mean}(\#fevals) \times \frac{\#\text{all runs (25)}}{\#\text{successful runs}}$, where $\#fevals$ includes only successful runs.

Small values of $\texttt{FEs}$ and therefore large values in the empirical distribution graphs are preferable.

Comparison Study

Conclusion

The CMA-ES with multi-start and increasing population size

- performs best over all functions
- performs best on the function subsets
  - unimodal functions
  - solved multimodal functions
  - unsolved multimodal functions
- no parameter tuning were conducted
- G-CMA-ES, L-CMA-ES, and EDA have the most invariance properties
- on two separable problems G-CMA-ES is considerably outperformed

Comparison Study

Conclusion

The CMA-ES with multi-start and increasing population size

- performs best over all functions
- performs best on the function subsets
  - unimodal functions
  - solved multimodal functions
  - unsolved multimodal functions
- no parameter tuning were conducted
- G-CMA-ES, L-CMA-ES, and EDA have the most invariance properties
- on two separable problems G-CMA-ES is considerably outperformed

Comparison Study

Conclusion

The CMA-ES with multi-start and increasing population size

- performs best over all functions
- performs best on the function subsets
    - unimodal functions
    - solved multimodal functions
    - unsolved multimodal functions
- no parameter tuning were conducted
- G-CMA-ES, L-CMA-ES, and EDA have the most invariance properties
- on two separable problems G-CMA-ES is considerably outperformed

Comparison Study

Conclusion

The CMA-ES with multi-start and increasing population size

- performs best over all functions
- performs best on the function subsets
  - unimodal functions
  - solved multimodal functions
  - unsolved multimodal functions
- no parameter tuning were conducted
- G-CMA-ES, L-CMA-ES, and EDA have the most invariance properties
- on two separable problems G-CMA-ES is considerably outperformed

Comparison Study
Conclusion

The CMA-ES with multi-start and increasing population size

- performs best over all functions
- performs best on the function subsets
  - unimodal functions
  - solved multimodal functions
  - unsolved multimodal functions
- no parameter tuning were conducted
- G-CMA-ES, L-CMA-ES, and EDA have the most invariance properties
- on two separable problems G-CMA-ES is considerably outperformed

Conclusion

The Take Home Message

Difficulties of a non-linear optimization problem are

- ruggedness

  demands a non-local (stochastic?) approach

- dimensionality and non-separabitity

  demands to exploit problem structure, e.g. neighborhood

- ill-conditioning

  demands to acquire a second order model

The CMA-ES addresses these difficulties and is

- a robust local search algorithm

  BFGS is roughly ten times faster on convex quadratic f

- a robust global search algorithm

  empirically outperformes plain or hybrid EAs on most functions

- successfully applied to many real-world applications

  easily applicable as quasi parameter free

Conclusion

The Take Home Message

Difficulties of a non-linear optimization problem are

- ruggedness

  demands a non-local (stochastic?) approach

- dimensionality and non-separabitity

  demands to exploit problem structure, e.g. neighborhood

- ill-conditioning

  demands to acquire a second order model

The CMA-ES addresses these difficulties and is

- a robust local search algorithm

  BFGS is roughly ten times faster on convex quadratic f

- a robust global search algorithm

  empirically outperformes plain or hybrid EAs on most functions

- successfully applied to many real-world applications

  easily applicable as quasi parameter free

Thank You

`http://www.bionik.tu-berlin.de/user/niko/cmaesintro.html`
or google NIKOLAUS HANSEN

Strategy Internal CPU Consumption

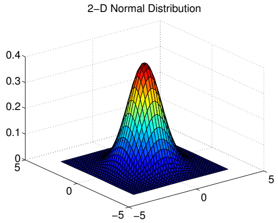
On a 2.5GHz processor our CMA-ES implementation needs

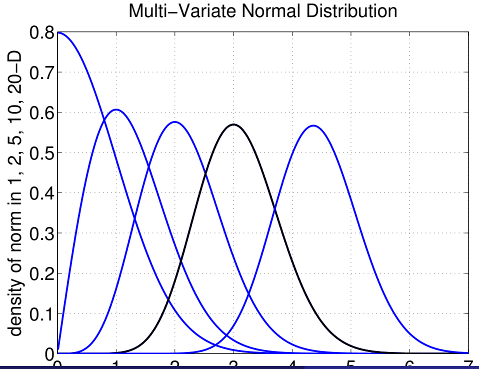
- roughly $3 \times 10^{-8}(n+4)^2$ seconds per function evaluation
- for one million function evaluations roughly

n	time
10	5s
30	30s
100	300s

Normal Distribution Revisited

While the maximum likelihood of the multi-variate normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ is at zero, the distribution of its norm $\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|$ reveals a different, surprising picture.

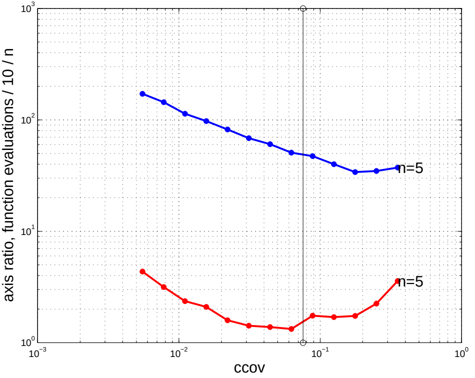




- In 10-D (black) the usual step length is about $3 \times \sigma$ and step lengths smaller than $1 \times \sigma$ virtually never occur
- Remind: this norm-density shape maximizes the distribution entropy

Determining Learning Rates

Learning rate for the covariance matrix



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

optimal condition number for $\mathbf{C}$ is one,

initial condition number of $\mathbf{C}$ equals 10^4

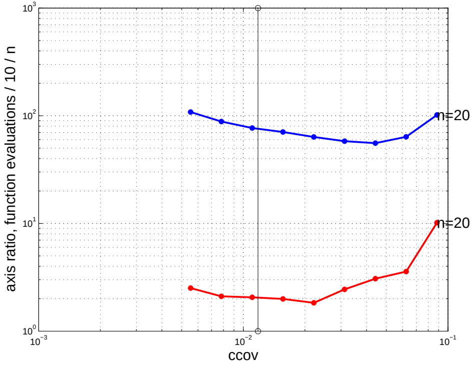
shown are single runs

x-axis: learning rate for the covariance matrix

y-axis: square root of final condition number of $\mathbf{C}$ (red), number of function evaluations to reach f_{stop} (blue)

Determining Learning Rates

Learning rate for the covariance matrix



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

optimal condition number for $\mathbf{C}$ is one,

initial condition number of $\mathbf{C}$ equals 10^4

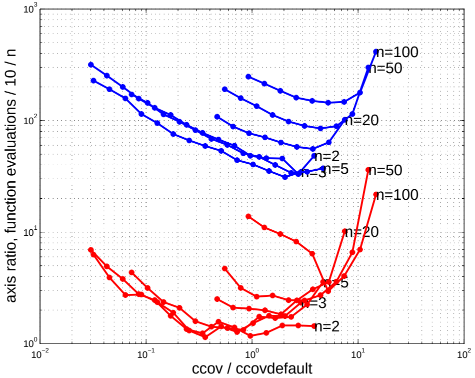
shown are single runs

x-axis: learning rate for the covariance matrix

y-axis: square root of final condition number of $\mathbf{C}$ (red), number of function evaluations to reach f_{stop} (blue)

Determining Learning Rates

Learning rate for the covariance matrix



- learning rates can be identified on simple functions

  exploiting invariance properties
- the outcome depends on the problem dimensionality
- the specific objective function is rather insignificant

x-axis: factor for learning rate for the covariance matrix

y-axis: square root of final condition number of $\mathbf{C}$ (red),

number of function evaluations to reach f_{stop} (blue)

. . . step size control

EMNA versus CMA

Both algorithms use the same sample distribution

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

In EMNA$_{\text{global}}$ $\sigma \equiv 1$ and

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

$$\mathbf{C} \leftarrow \frac{1}{\mu} \sum_{i=1}^{\mu} (\boldsymbol{x}_{i:\lambda} - \boldsymbol{m})(\boldsymbol{x}_{i:\lambda} - \boldsymbol{m})^{\mathrm{T}}$$

In CMA, for $c_{\text{cov}} = 1$, with rank-μ update only

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

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

where $\boldsymbol{z}_{i:\lambda} = \frac{\boldsymbol{x}_{i:\lambda} - \boldsymbol{m}_{\text{old}}}{\sigma}$

EMNA versus CMA

Both algorithms use the same sample distribution

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

In EMNA$_{\text{global}}$ $\sigma \equiv 1$ and

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

$$\mathbf{C} \leftarrow \frac{1}{\mu} \sum_{i=1}^{\mu} (\boldsymbol{x}_{i:\lambda} - \boldsymbol{m})(\boldsymbol{x}_{i:\lambda} - \boldsymbol{m})^{\mathrm{T}}$$

In CMA, for $c_{\text{cov}} = 1$, with rank-μ update only

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

$$\mathbf{C} \leftarrow \sum_{i=1}^{\mu} w_i z_{i:\lambda} z_{i:\lambda}^{\mathrm{T}}$$

where $z_{i:\lambda} = \frac{\boldsymbol{x}_{i:\lambda} - \boldsymbol{m}_{\text{old}}}{\sigma}$

EMNA versus CMA

Both algorithms use the same sample distribution

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

In EMNA$_{\text{global}}$ $\sigma \equiv 1$ and

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

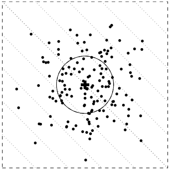
$$\mathbf{C} \leftarrow \frac{1}{\mu} \sum_{i=1}^{\mu} (\boldsymbol{x}_{i:\lambda} - \boldsymbol{m})(\boldsymbol{x}_{i:\lambda} - \boldsymbol{m})^{\mathrm{T}}$$

In CMA, for $c_{\text{cov}} = 1$, with rank-μ update only

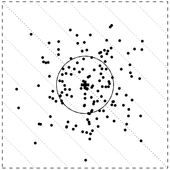
$$\boldsymbol{m} \leftarrow \sum_{i=1}^{\mu} w_i\, \boldsymbol{x}_{i:\lambda}$$

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

where $\boldsymbol{z}_{i:\lambda} = \frac{\boldsymbol{x}_{i:\lambda} - \boldsymbol{m}_{\text{old}}}{\sigma}$



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



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

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

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

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

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

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

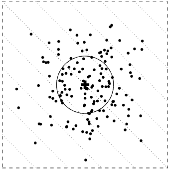
$m_{\text{new}} = m_{\text{old}} + \frac{1}{\mu} \sum z_{i:\lambda}$

new distribution

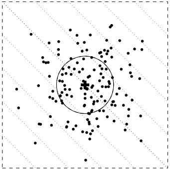
rank-μ CMA conducts a PCA of steps

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

the CMA-update yields a larger variance in particular in gradient direction

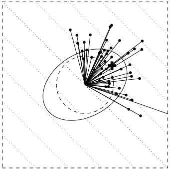


$x_i = m_{\text{old}} + z_i, \quad z_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$

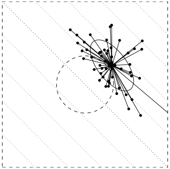


$x_i = m_{\text{old}} + z_i, \quad z_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$

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



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



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

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

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

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

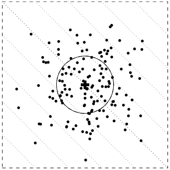
new distribution

rank-μ CMA conducts a PCA of steps

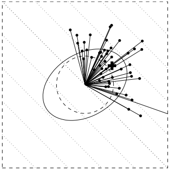
EMNA$_{\text{global}}$ conducts a PCA of points

the CMA-update yields a larger variance in particular in gradient direction

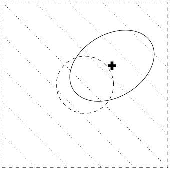
m ... is the minimizer for the variances when calculating $\mathbf{C}$



$\boldsymbol{x}_i = \boldsymbol{m}_{\text{old}} + z_i, \quad z_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$

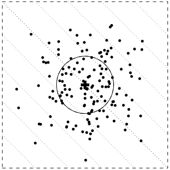


$\mathbf{C} \leftarrow$
$\frac{1}{\mu}\sum(x_{i:\lambda} - \boldsymbol{m}_{\text{old}})(x_{i:\lambda} - \boldsymbol{m}_{\text{old}})^{\mathrm{T}}$

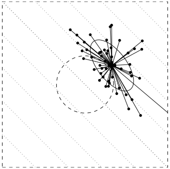


$\boldsymbol{m}_{\text{new}} = \boldsymbol{m}_{\text{old}} + \frac{1}{\mu}\sum z_{i:\lambda}$

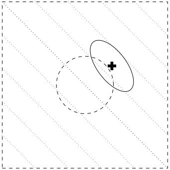
rank-μ CMA conducts a PCA of steps



$\boldsymbol{x}_i = \boldsymbol{m}_{\text{old}} + z_i, \quad z_i \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$



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



$\boldsymbol{m}_{\text{new}} = \boldsymbol{m}_{\text{old}} + \frac{1}{\mu}\sum z_{i:\lambda}$

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

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

the CMA-update yields a larger variance in particular in gradient direction

$\boldsymbol{m}$ is the minimizer for the variances when calculating $\mathbf{C}$

Population Size on Unimodal Functions

On unimodal functions the performance degrades at most linearly with increasing population size.

most often a small population size, $\lambda \leq 10$, is optimal

Problem Formulation

A real world problem requires

- a representation; the encoding of problem parameters into $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^n$
- the definition of a objective function $f : \boldsymbol{x} \mapsto f(\boldsymbol{x})$ to be minimized

One might distinguish two approaches

Natural Encoding

Use a "natural" encoding and design the optimizer with respect to the problem e.g. use of specific "genetic operators"

frequently done in discrete domain

Concerned Encoding (Pure Black Box)

Put problem specific knowledge into the encoding and use a "generic" optimizer

frequently done in continuous domain

Advantage: Sophisticated and well-validated optimizers can be used

Problem Formulation

A real world problem requires

- a representation; the encoding of problem parameters into $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^n$
- the definition of a objective function $f : \boldsymbol{x} \mapsto f(\boldsymbol{x})$ to be minimized

One might distinguish two approaches

Natural Encoding

Use a "natural" encoding and design the optimizer with respect to the problem e.g. use of specific "genetic operators"

frequently done in discrete domain

Concerned Encoding (Pure Black Box)

Put problem specific knowledge into the encoding and use a "generic" optimizer

frequently done in continuous domain

Advantage: Sophisticated and well-validated optimizers can be used

Problem Formulation

A real world problem requires

- a representation; the encoding of problem parameters into $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^n$
- the definition of a objective function $f : \boldsymbol{x} \mapsto f(\boldsymbol{x})$ to be minimized

One might distinguish two approaches

Natural Encoding

Use a "natural" encoding and design the optimizer with respect to the problem e.g. use of specific "genetic operators"

frequently done in discrete domain

Concerned Encoding (Pure Black Box)

Put problem specific knowledge into the encoding and use a "generic" optimizer

frequently done in continuous domain

Advantage: Sophisticated and well-validated optimizers can be used