CMA-ES

Covariance Matrix Adaptation Evolution Strategy

Nikolaus Hansen
Inria & Ecole Polytechnique, France

September 2024

I am happy to answer questions at any time!

Need the slides? Available under Talks at my homepage:
http://www.cmap.polytechnique.fr/~nikolaus.hansen/talks.html

Topics

0. Problem statement

1. What makes an optimization problem difficult to solve?

2. How does the CMA-ES work?

   - Normal Distribution, Rank-Based Recombination
   - Step-Size Adaptation
   - Covariance Matrix Adaptation

3. What can users do?

   - Choice of problem formulation and encoding (not covered)
   - Choice of initial solution and initial step-size
   - Restarts, Increasing Population Size
   - Restricted Covariance Matrix

Problem Statement

Continuous Domain Search/Optimization

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

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

- Black Box scenario (direct search scenario)

  - gradients are not available or not useful
  - problem domain specific knowledge is used only within the black box, e.g. within an appropriate encoding

- Search costs: number of function evaluations

Problem Statement

Continuous Domain Search/Optimization

- Goal
  - fast convergence to the global optimum
  - solution $\boldsymbol{x}$ with small function value $f(\boldsymbol{x})$ with least search cost

    . . . or to a robust solution $\boldsymbol{x}$

    there are two conflicting objectives

- Typical Examples
  - shape optimization (e.g. using CFD) — curve fitting, airfoils
  - model calibration — biological, physical
  - parameter calibration — controller, plants, images

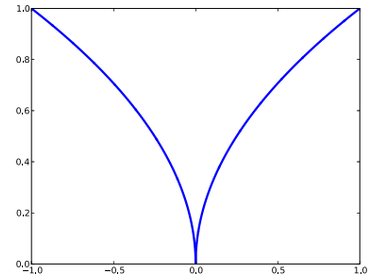
- Difficulties
  - exhaustive search is infeasible
  - naive random search takes too long
  - deterministic search is not successful / takes too long

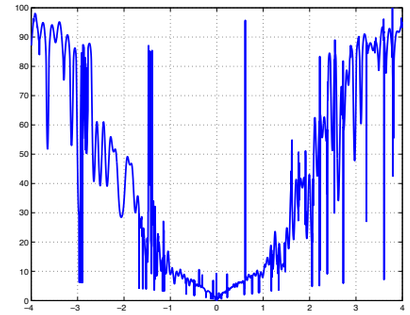
Approach: stochastic search, Evolutionary Algorithms

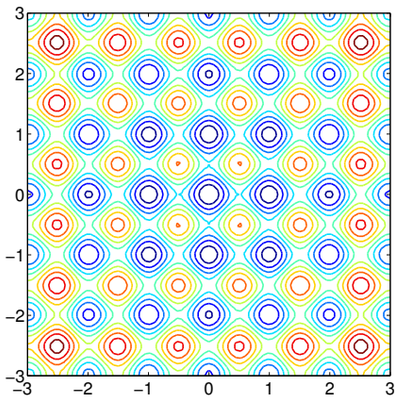
What Makes a Function Difficult to Solve?

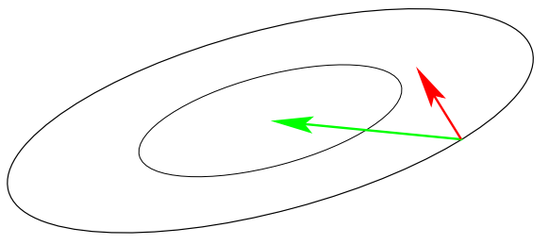
Why stochastic search?

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





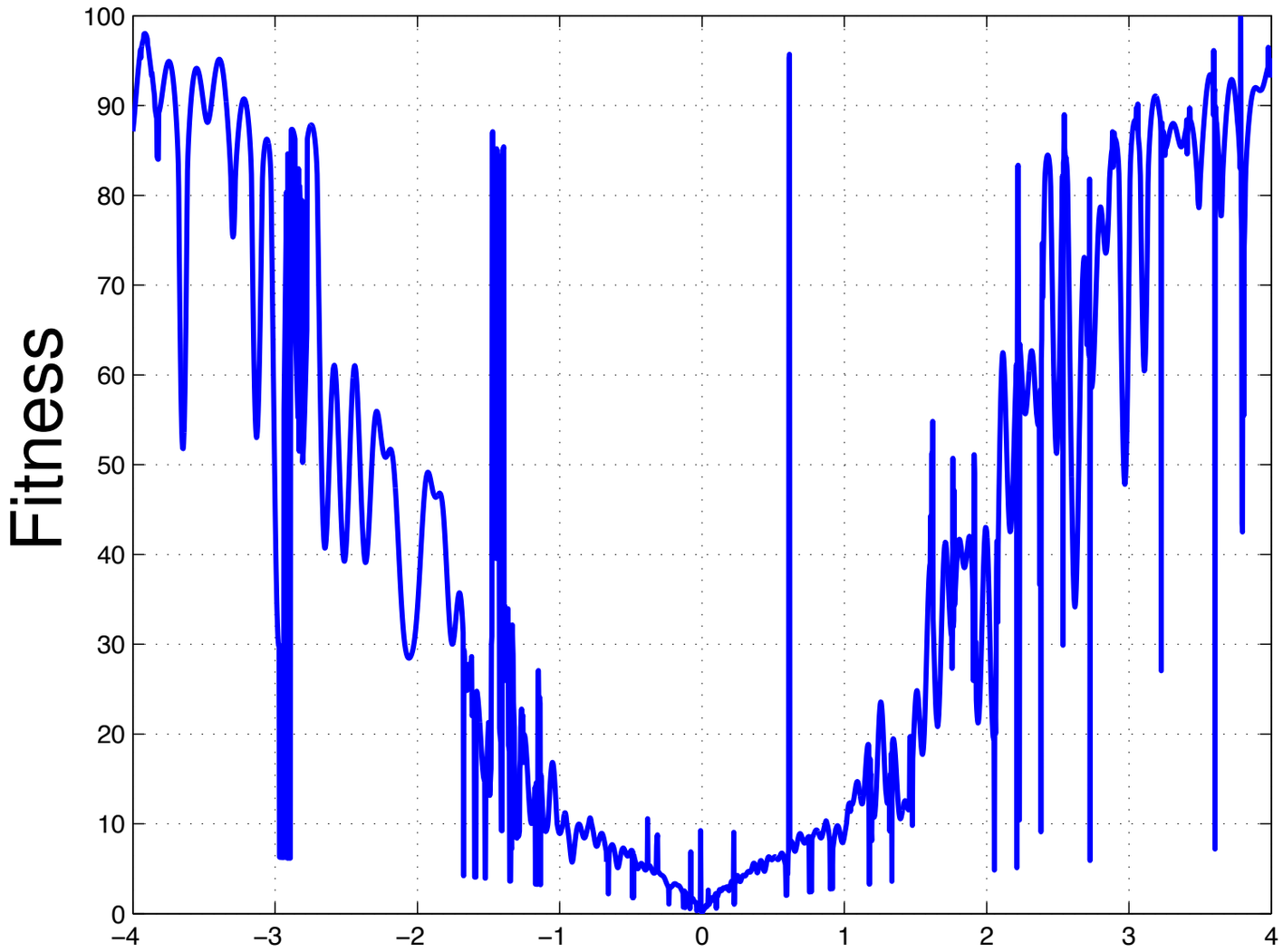




gradient direction Newton direction

Ruggedness

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



cut from a 5-D example, (easily) solvable with evolution strategies

Curse of Dimensionality

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

Example: Consider placing 20 points equally spaced onto the interval $[0, 1]$. Now consider the 10-dimensional space $[0, 1]^{10}$. To get similar coverage in terms of distance between adjacent points requires $20^{10} \approx 10^{13}$ points. 20 points appear now as isolated points in a vast empty space.

Remark: distance measures break down in higher dimensionalities (the central limit theorem kicks in)

Consequence: a search policy that is valuable in small dimensions might be useless in moderate or large dimensional search spaces. Example: exhaustive search.

Separable Problems

Definition (Separable Problem)

A function f is separable if

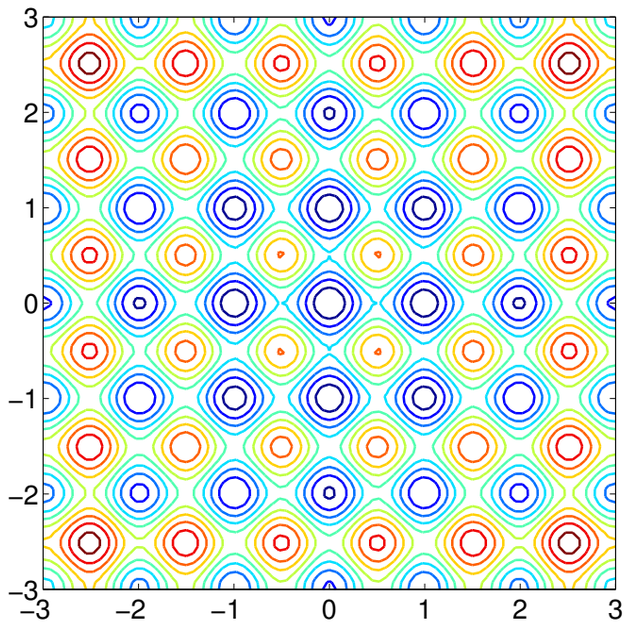
$$\arg\min_{(x_1,\ldots,x_n)} f(x_1,\ldots,x_n) = \left(\arg\min_{x_1} f(x_1,\ldots),\ldots,\arg\min_{x_n} f(\ldots,x_n)\right)$$

$\Rightarrow$ 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



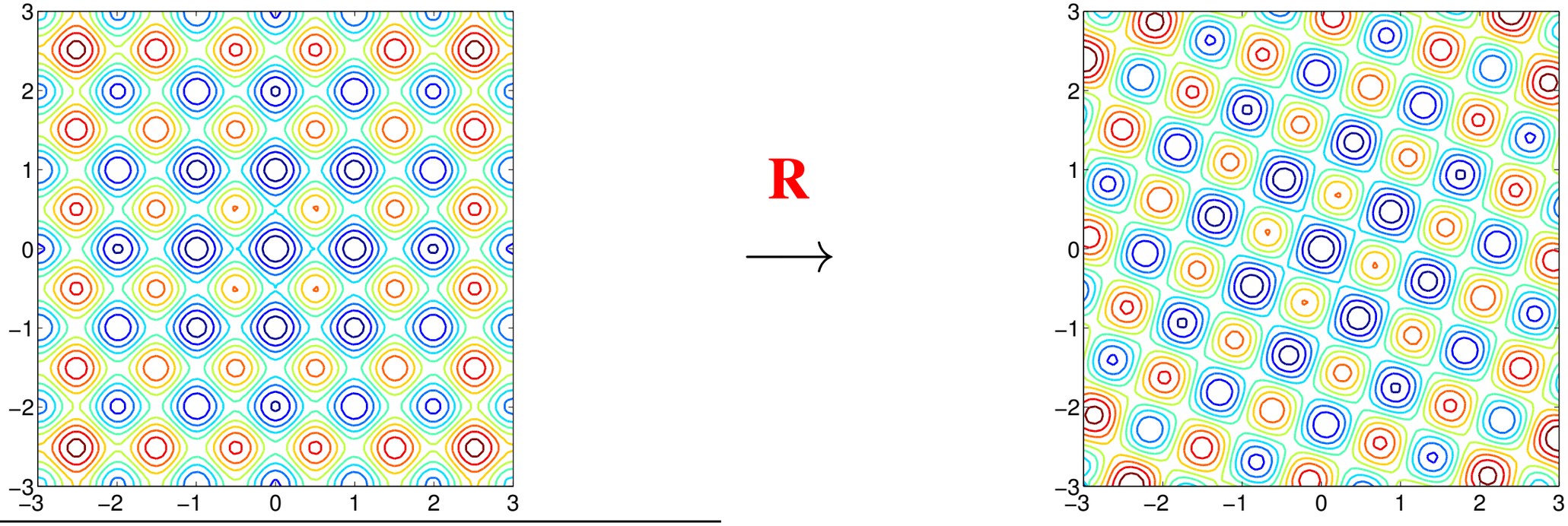
Non-Separable Problems

Building a non-separable problem from a separable one [1,2]

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



[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

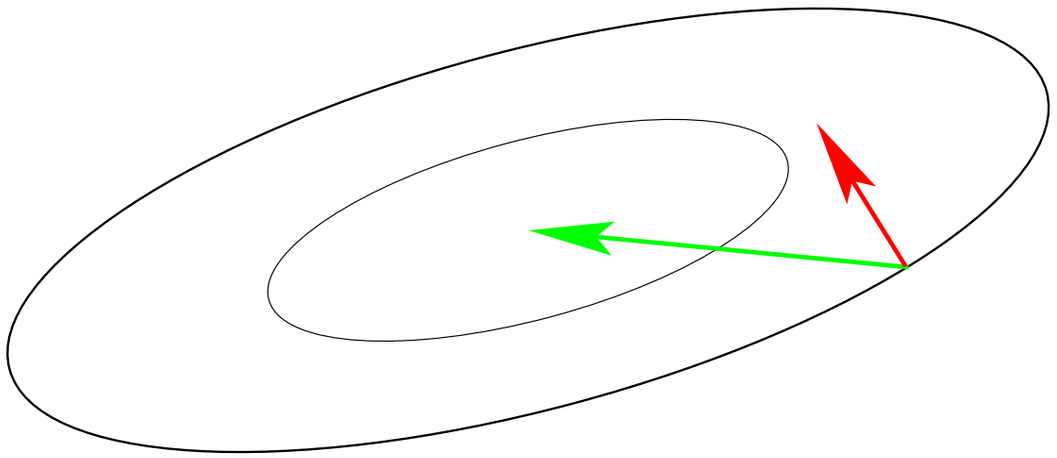
Ill-Conditioned Problems

Curvature of level sets

Consider the convex-quadratic function

$$f(\boldsymbol{x}) = \tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{x}^*)^T\boldsymbol{H}(\boldsymbol{x}-\boldsymbol{x}^*) = \tfrac{1}{2}\sum_i h_{i,i}\,(x_i-x_i^*)^2 + \tfrac{1}{2}\sum_{i\neq j} h_{i,j}\,(x_i-x_i^*)(x_j-x_j^*)$$

$\boldsymbol{H}$ is Hessian matrix of f and symmetric positive definite



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

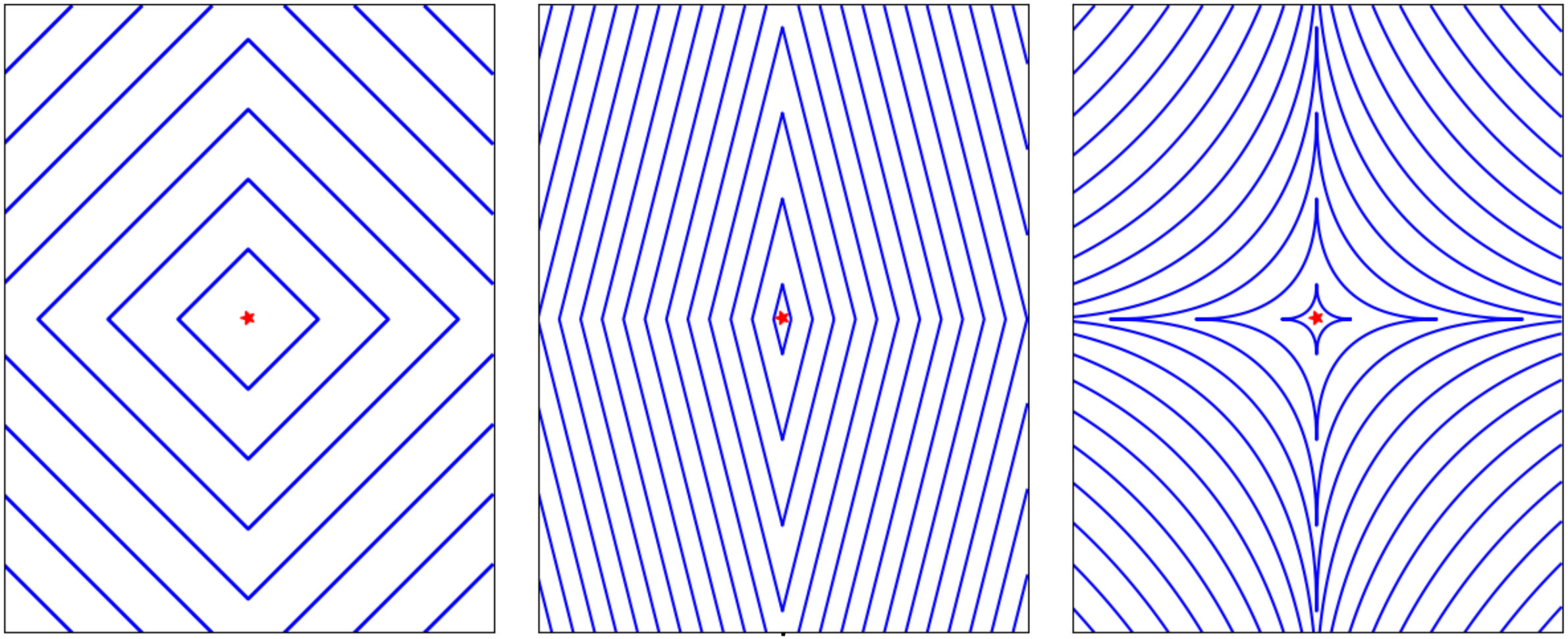
Newton direction $-\boldsymbol{H}^{-1}f'(\boldsymbol{x})^{\mathrm{T}}$

Ill-conditioning means squeezed level sets (high curvature).
Condition number equals nine here. Condition numbers up to 10^{10}
are not unusual 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 necessary.

Non-smooth level sets (sharp ridges)

Similar difficulty **but worse** than ill-conditioning



1-norm | scaled 1-norm | 1/2-norm

opening angle is the crucial parameter

What Makes a Function Difficult to Solve?

. . . and what can be done

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

. . . metaphors

Topics

1. What makes the problem difficult to solve?

2. How does the CMA-ES work?

   - Normal Distribution, Rank-Based Recombination
   - Step-Size Adaptation
   - Covariance Matrix Adaptation

3. What can/should the users do for the CMA-ES to work effectively on their problem?

   - Choice of problem formulation and encoding (not covered)
   - Choice of initial solution and initial step-size
   - Restarts, Increasing Population Size
   - Restricted Covariance Matrix

Stochastic Search

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

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

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

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

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

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

Stochastic Search

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

Initialize distribution parameters $\boldsymbol{\theta}$, set population size $\lambda \in \mathbb{N}$

While not terminate

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

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

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

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

Stochastic Search

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

Initialize distribution parameters $\boldsymbol{\theta}$, set population size $\lambda \in \mathbb{N}$

While not terminate

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

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

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

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

Stochastic Search

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

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

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

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

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

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

Stochastic Search

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

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

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

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

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

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

Stochastic Search

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

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

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

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

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

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

Stochastic Search

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

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

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

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

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

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

Stochastic Search

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

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

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

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

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

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

The CMA-ES

Input: $\boldsymbol{m} \in \mathbb{R}^n$; $\sigma \in \mathbb{R}_+$; $\lambda \in \mathbb{N}_{\geq 2}$, usually $\lambda \geq 5$, default $4 + \lfloor 3 \log n \rfloor$

Set $c_m = 1$; $c_1 \approx 2/n^2$; $c_\mu \approx \mu_w/n^2$; $c_\mathrm{c} \approx 4/n$; $c_\sigma \approx 1/\sqrt{n}$; $d_\sigma \approx 1$; $w_{i=1\ldots\lambda}$ decreasing in i and $\sum_i^\mu w_i = 1$, $w_\mu > 0 \geq w_{\mu+1}$, $\mu_w^{-1} := \sum_{i=1}^\mu w_i^2 \approx 3/\lambda$

Initialize $\mathbf{C} = \mathbf{I}$, and $\boldsymbol{p}_\mathrm{c} = \mathbf{0}$, $\boldsymbol{p}_\sigma = \mathbf{0}$

While not *terminate*

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

$$\boldsymbol{m} \leftarrow \boldsymbol{m} + c_m \sigma \boldsymbol{y}_w, \quad \text{where } \boldsymbol{y}_w = \textstyle\sum_{i=1}^\mu w_i \, \boldsymbol{y}_{\mathrm{rk}^{-1}(i)} \qquad \text{update mean}$$

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

$$\boldsymbol{p}_\mathrm{c} \leftarrow (1 - c_\mathrm{c})\, \boldsymbol{p}_\mathrm{c} + \mathbb{1}_{[0,2n]}\left\{\|\boldsymbol{p}_\sigma\|^2\right\} \sqrt{1 - (1 - c_\mathrm{c})^2}\sqrt{\mu_w}\, \boldsymbol{y}_w \qquad \text{path for } \mathbf{C}$$

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

$$\mathbf{C} \leftarrow \mathbf{C} + c_\mu \textstyle\sum_{i=1}^\lambda w_{\mathrm{rk}(i)} \left(\boldsymbol{y}_i \boldsymbol{y}_i^\mathsf{T} - \mathbf{C}\right) + c_1 \left(\boldsymbol{p}_\mathrm{c} \boldsymbol{p}_\mathrm{c}^\mathsf{T} - \mathbf{C}\right) \qquad \text{update } \mathbf{C}$$

Not covered: termination, restarts, useful output, search boundaries and encoding, corrections for: positive definiteness guaranty, $\boldsymbol{p}_\mathrm{c}$ variance loss, c_σ and d_σ for large λ

Stochastic Search

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

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

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

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

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

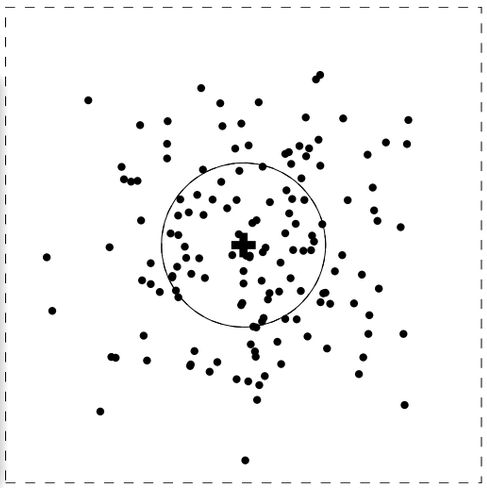
Natural template for (incremental) *Estimation of Distribution Algorithms*

Evolution Strategies

New search points are sampled normally distributed

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

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



where

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

here, all new points are sampled with the same parameters

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

Why Normal Distributions?

1. widely observed in nature, for example as phenotypic traits
2. only stable distribution with finite variance

   stable means that the sum of normal variates is again normal:

$$\mathcal{N}(\boldsymbol{x}, \mathbf{A}) + \mathcal{N}(\boldsymbol{y}, \mathbf{B}) \sim \mathcal{N}(\boldsymbol{x} + \boldsymbol{y},\ \mathbf{A} + \mathbf{B})$$

   helpful in design and analysis of algorithms related to the *central limit theorem*

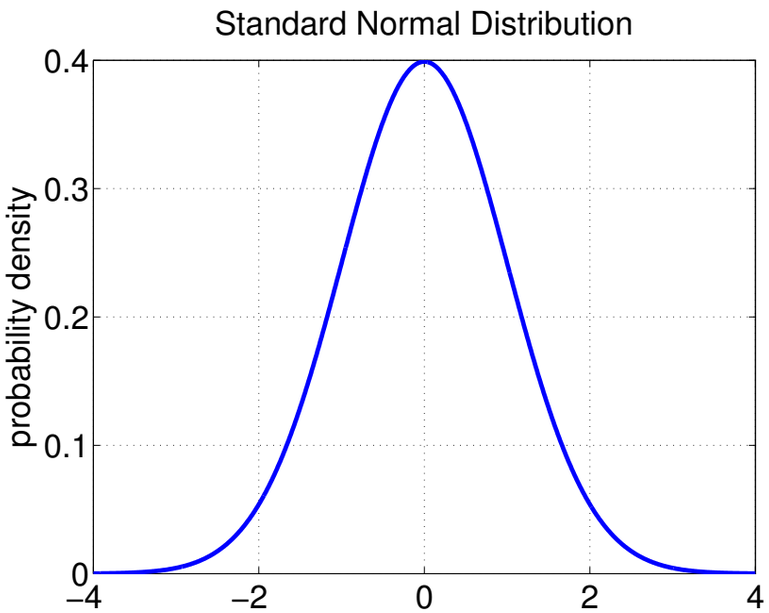
3. most convenient way to generate isotropic search points

   the isotropic distribution does not favor any direction, rotational invariant

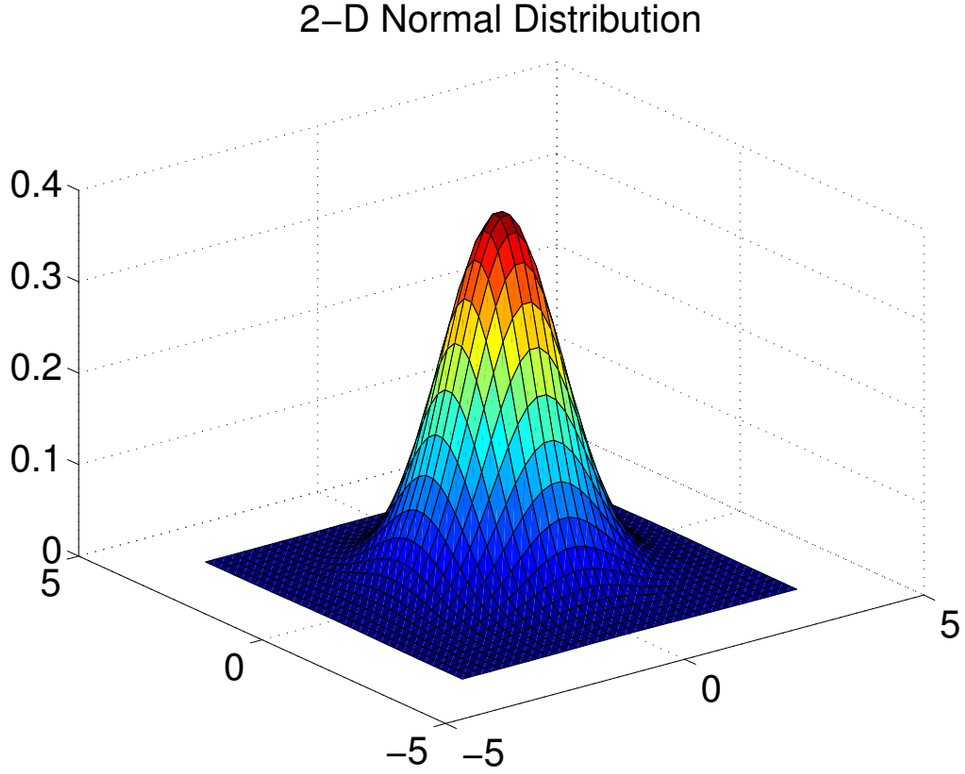
4. maximum entropy distribution with finite variance

   the least possible assumptions on f in the distribution shape

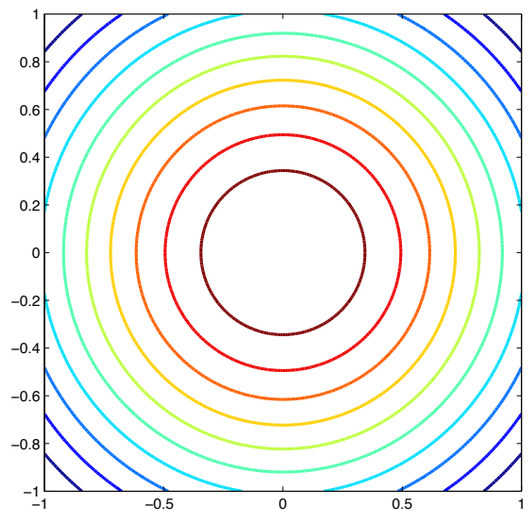
Normal Distribution



probability density of the 1-D standard normal distribution



probability density of a 2-D normal distribution

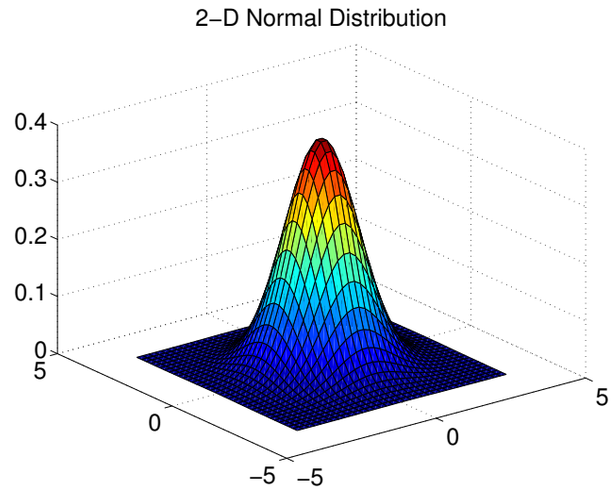


The Multi-Variate (n-Dimensional) Normal Distribution

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 $\boldsymbol{m}$

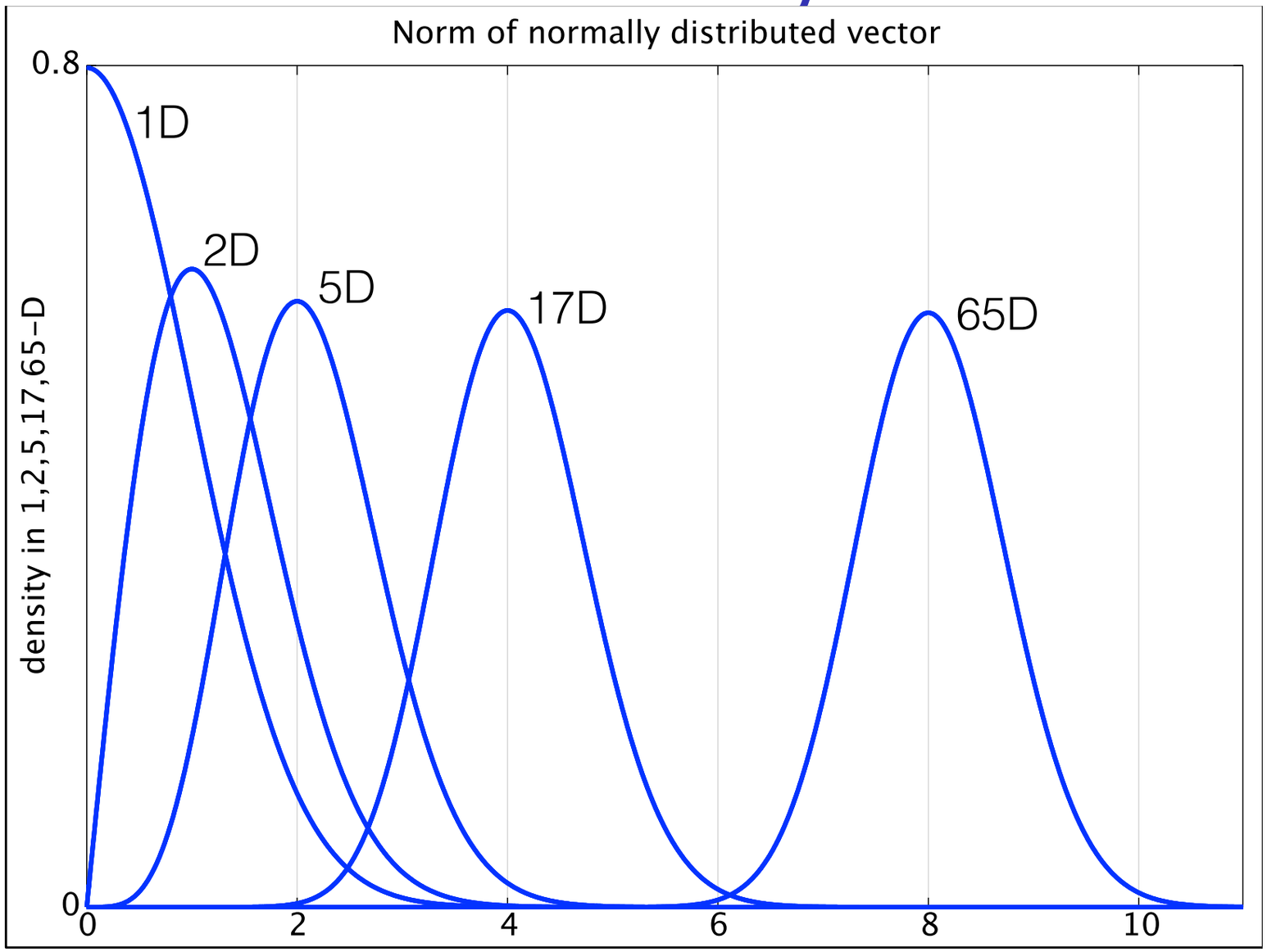
- determines the displacement (translation)
- value with the largest density (modal value)
- the distribution is symmetric about the distribution mean

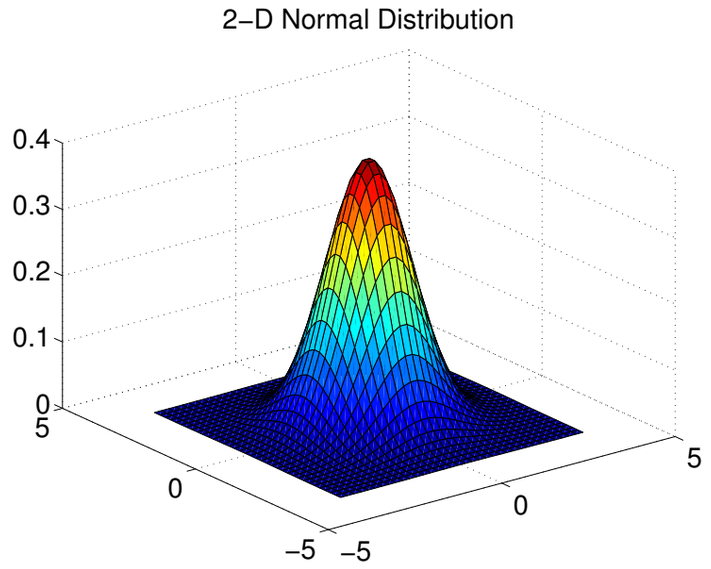


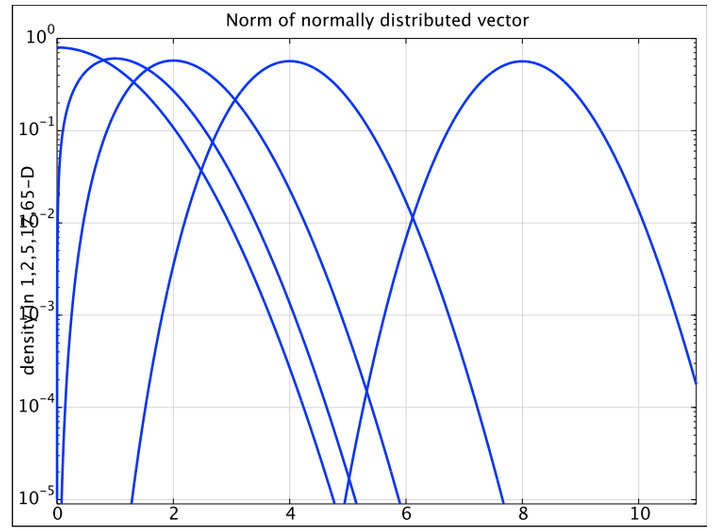
The covariance matrix $\mathbf{C}$

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

Effect of Dimensionality







$\|\mathcal{N}(\mathbf{0},\mathbf{I})\| \longrightarrow \mathcal{N}\left(\sqrt{n-1/2}, 1/2\right)$ with modal value $\sqrt{n-1}$

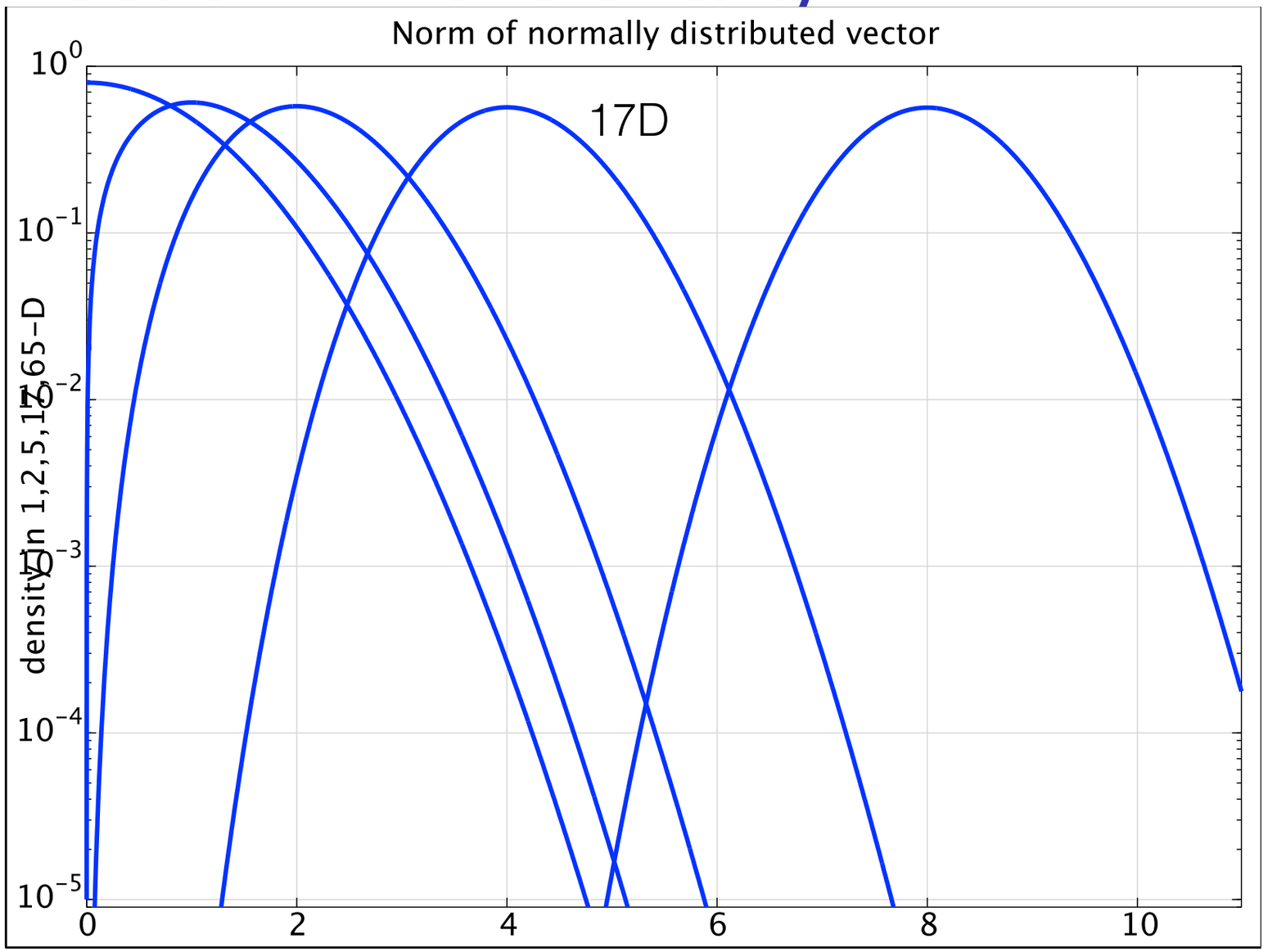
yet: maximum entropy distribution

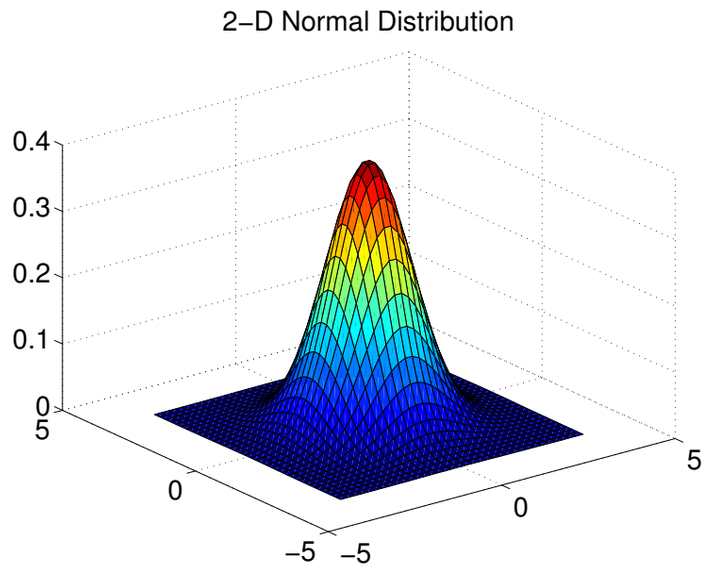
also consider a difference between two vectors:

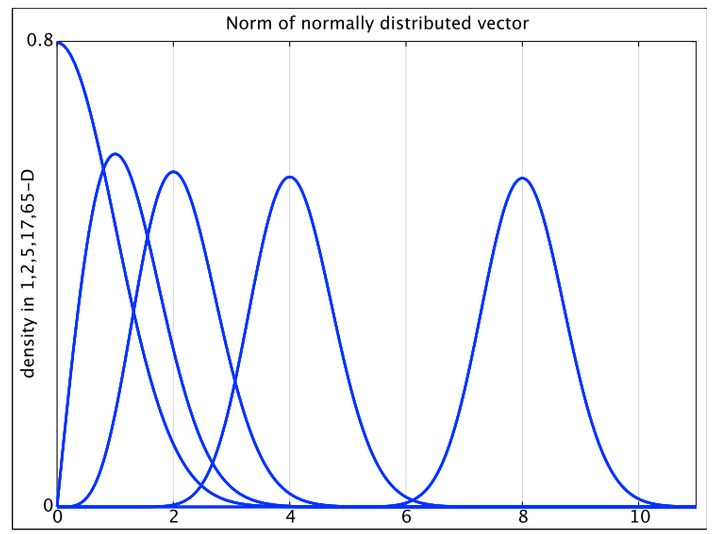
$\|\mathcal{N}(\mathbf{0},\mathbf{I}) - \mathcal{N}(\mathbf{0},\mathbf{I})\| \sim \|\mathcal{N}(\mathbf{0},\mathbf{I}) + \mathcal{N}(\mathbf{0},\mathbf{I})\| \sim \sqrt{2}\|\mathcal{N}(\mathbf{0},\mathbf{I})\|$

...

Effect of Dimensionality







$\|\mathcal{N}(\mathbf{0},\mathbf{I})\| \longrightarrow \mathcal{N}\left(\sqrt{n-1/2}, 1/2\right)$ with modal value $\sqrt{n-1}$

yet: maximum entropy distribution

also consider a difference between two vectors:

$\|\mathcal{N}(\mathbf{0},\mathbf{I}) - \mathcal{N}(\mathbf{0},\mathbf{I})\| \sim \|\mathcal{N}(\mathbf{0},\mathbf{I}) + \mathcal{N}(\mathbf{0},\mathbf{I})\| \sim \sqrt{2}\|\mathcal{N}(\mathbf{0},\mathbf{I})\|$

Stochastic Search

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

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

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

Everything depends on the definition of P and F_θ

deterministic algorithms are covered as well

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

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

The $(\mu/\mu, \lambda)$-ES, Update of the Distribution Mean

Non-elitist selection and intermediate (weighted) recombination

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

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

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

where

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

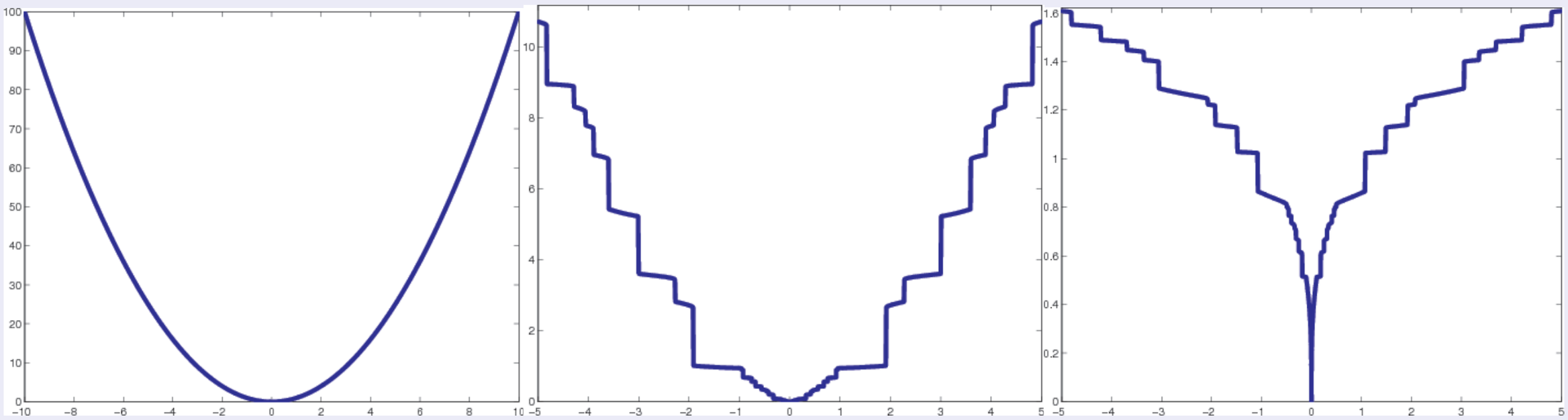
The best μ points are selected from the new solutions (non-elitistic) and weighted intermediate recombination is applied.

Invariance Under Monotonically Increasing Functions

Rank-based algorithms

Update of all parameters uses only the ranks

$$f(x_{1:\lambda}) \leq f(x_{2:\lambda}) \leq \ldots \leq f(x_{\lambda:\lambda})$$



$$g(f(x_{1:\lambda})) \leq g(f(x_{2:\lambda})) \leq \ldots \leq g(f(x_{\lambda:\lambda})) \quad \forall g$$

g is strictly monotonically increasing

g preserves ranks

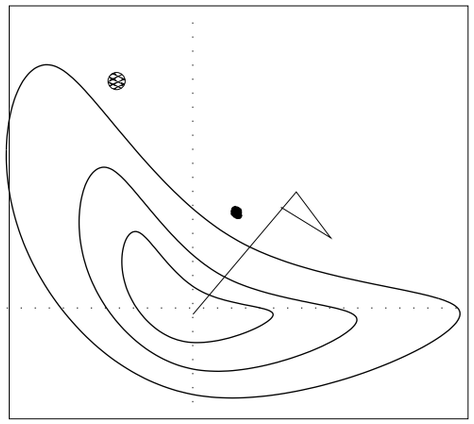
[3]

[3] Whitley 1989. The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best, ICGA

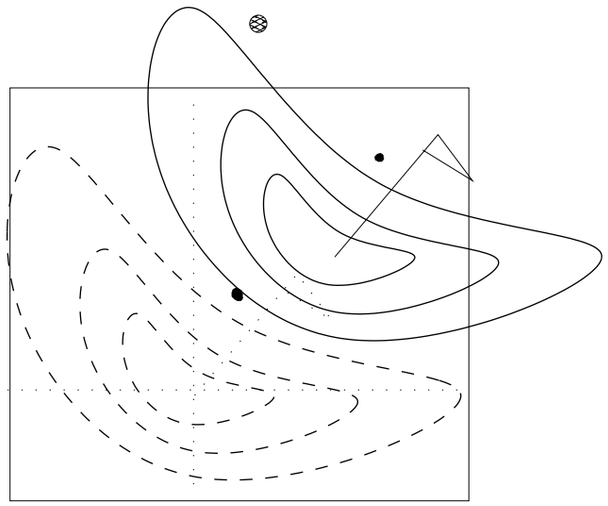
Basic Invariance in Search Space

- translation invariance

is true for most optimization algorithms



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



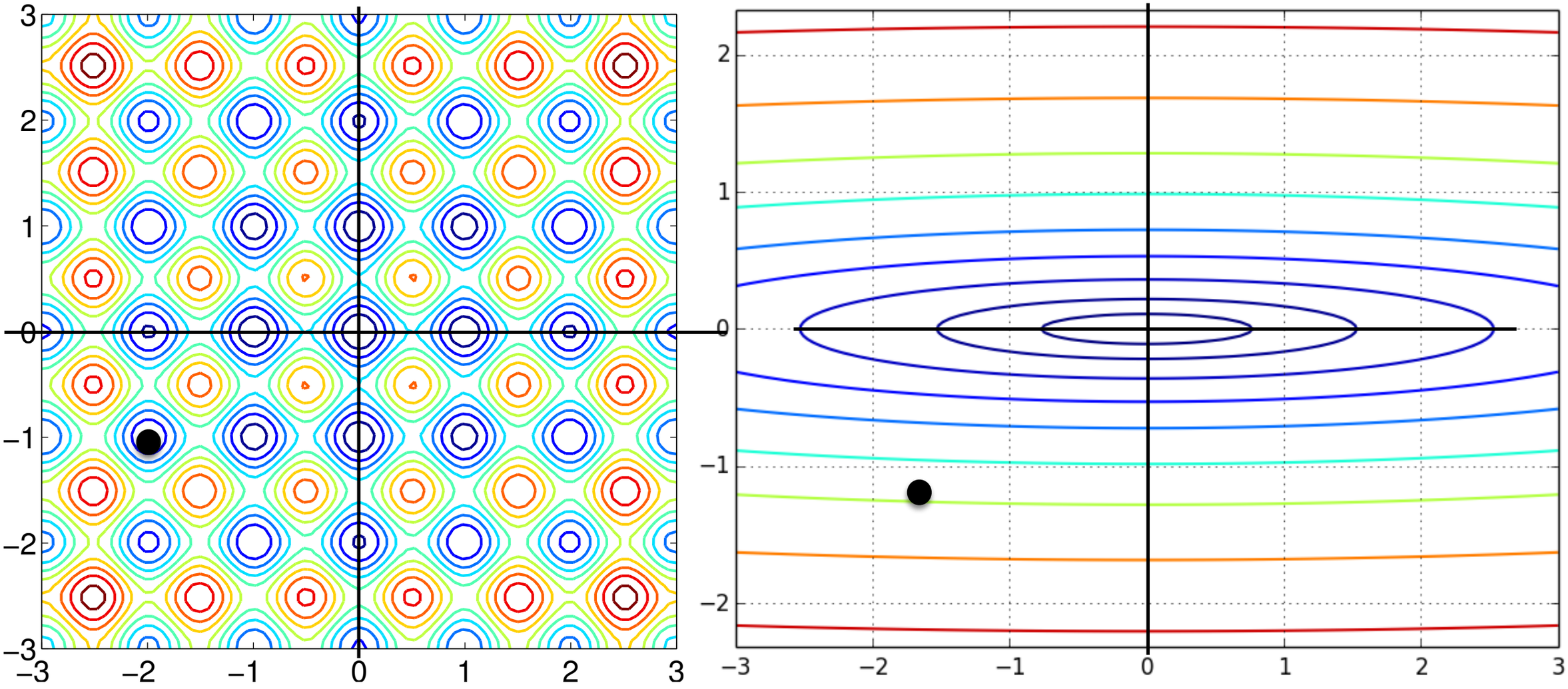
Identical behavior on f and f_a

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

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

Invariance Under Rigid Search Space Transformation

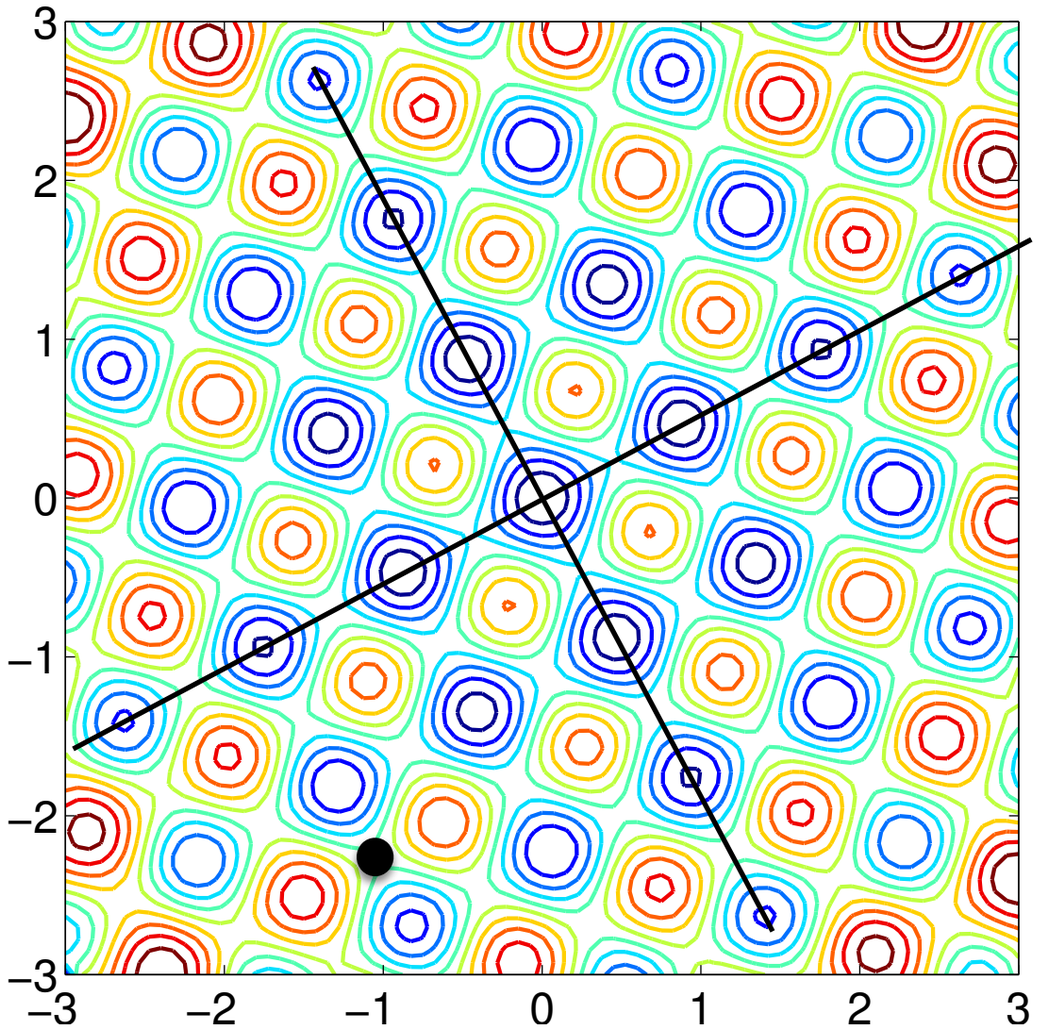
$f = h_{\mathrm{Rast}}$ f-level sets in dimension 2 $f = h$

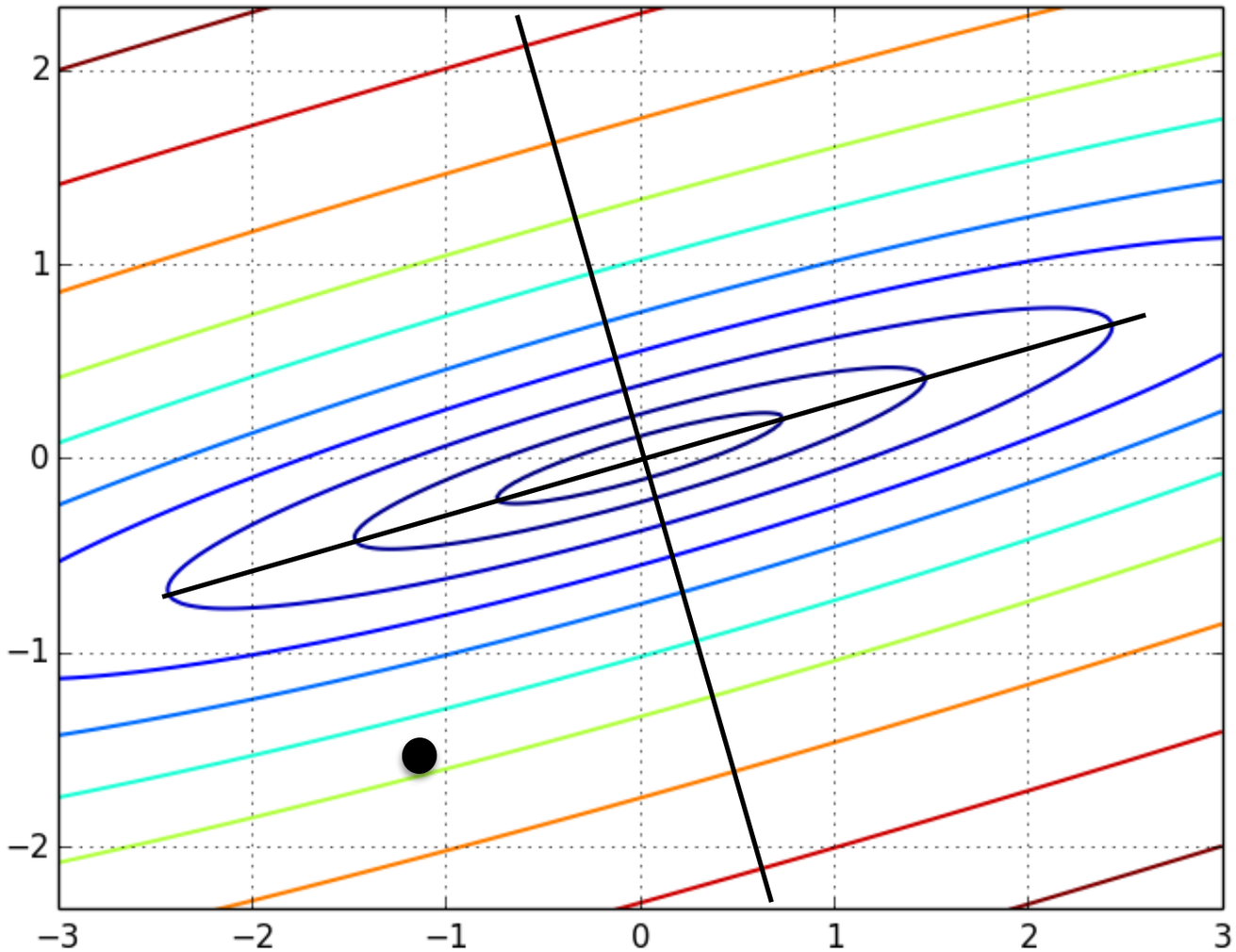


for example, invariance under search space rotation
(separable $\Leftrightarrow$ non-separable)

Invariance Under Rigid Search Space Transformation

$f = h_{\mathrm{Rast}} \circ R$ f-level sets in dimension 2 $f = h \circ R$





for example, invariance under search space rotation
(separable $\Leftrightarrow$ non-separable)

Invariance

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
  - from benchmark functions
  - from solved real world problems

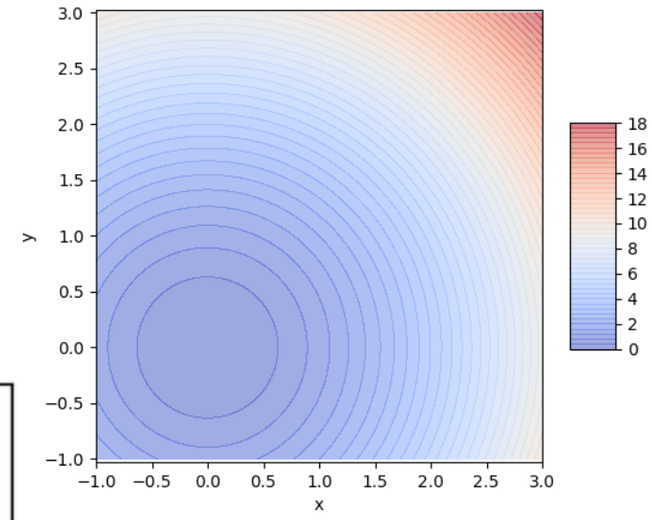
  are only useful if they do generalize to other problems

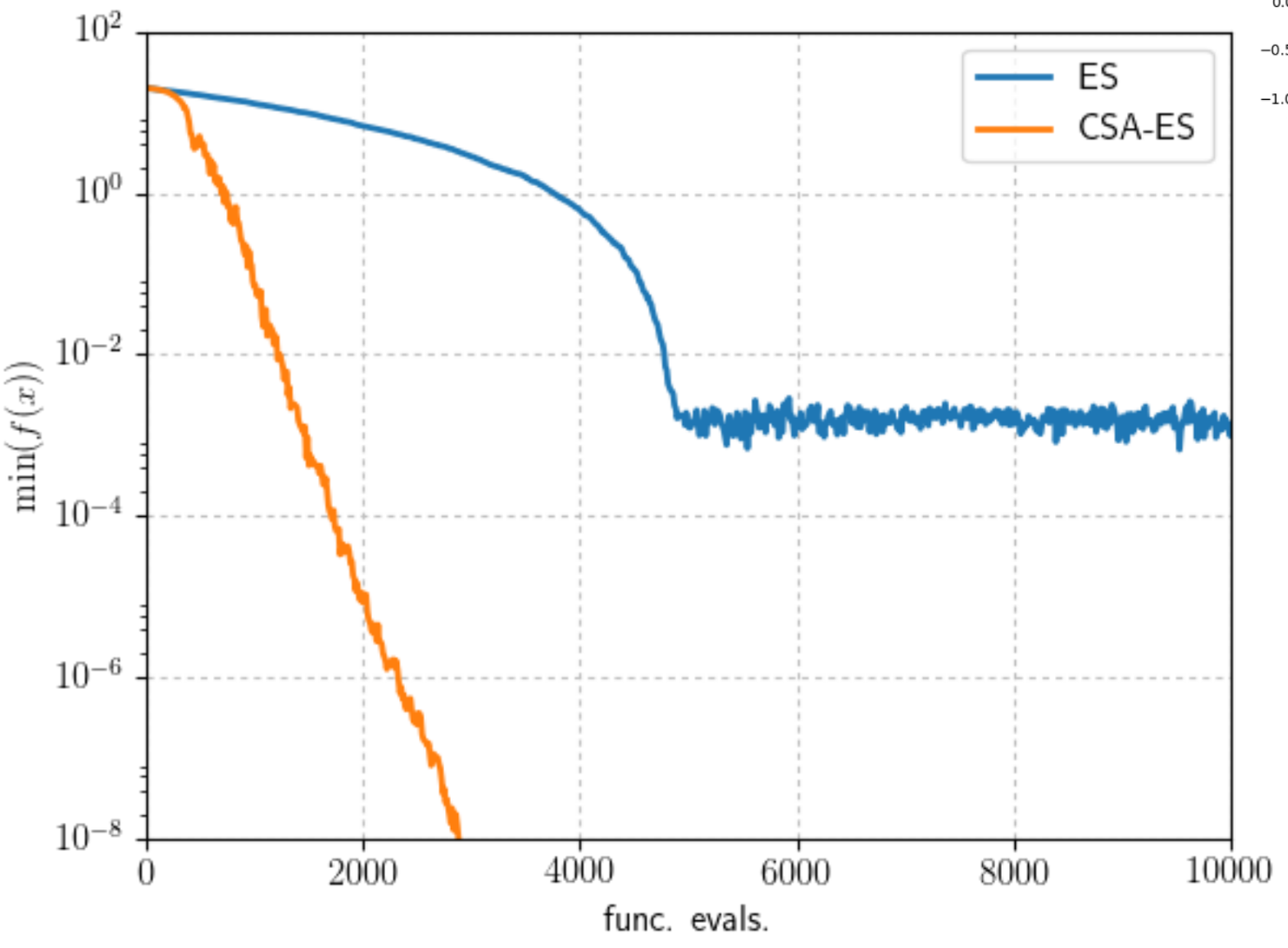
- Invariance is a strong non-empirical statement about generalization

  generalizing (identical) performance from a single function to a whole class of functions

consequently, invariance is important for the evaluation of search algorithms

Summary





On 20D Sphere Function: $f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 = \|\mathbf{x}\|^2$

- ES without adaptation can't approach the optimum $\Rightarrow$ adaptation required

Topics

0. Problem statement

1. What makes an optimization problem difficult to solve?

2. How does the CMA-ES work?

   - Normal Distribution, Rank-Based Recombination
   - Step-Size Adaptation
   - Covariance Matrix Adaptation

3. What can users do?

   - Choice of problem formulation and encoding (not covered)
   - Choice of initial solution and initial step-size
   - Restarts, Increasing Population Size
   - Restricted Covariance Matrix

Methods for Step-Size Control

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

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

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

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

  simplified "global" self-adaptation

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

  self-adaptation derandomized and non-localized

  [f]

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

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

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

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

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

[f] Hansen *et al* 2014, How to assess step-size adaptation mechanisms in randomised search, *PPSN IX*

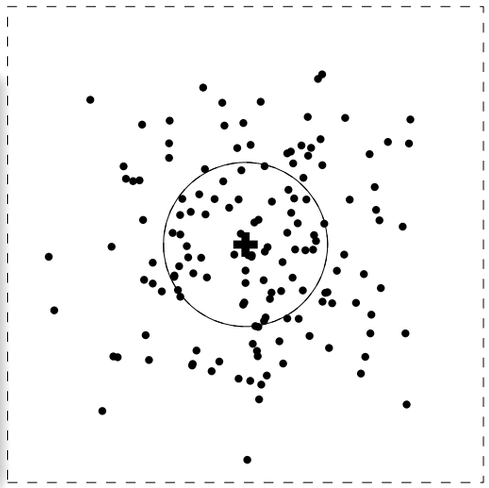
Evolution Strategies

Recalling

New search points are sampled normally distributed

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

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



where

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

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

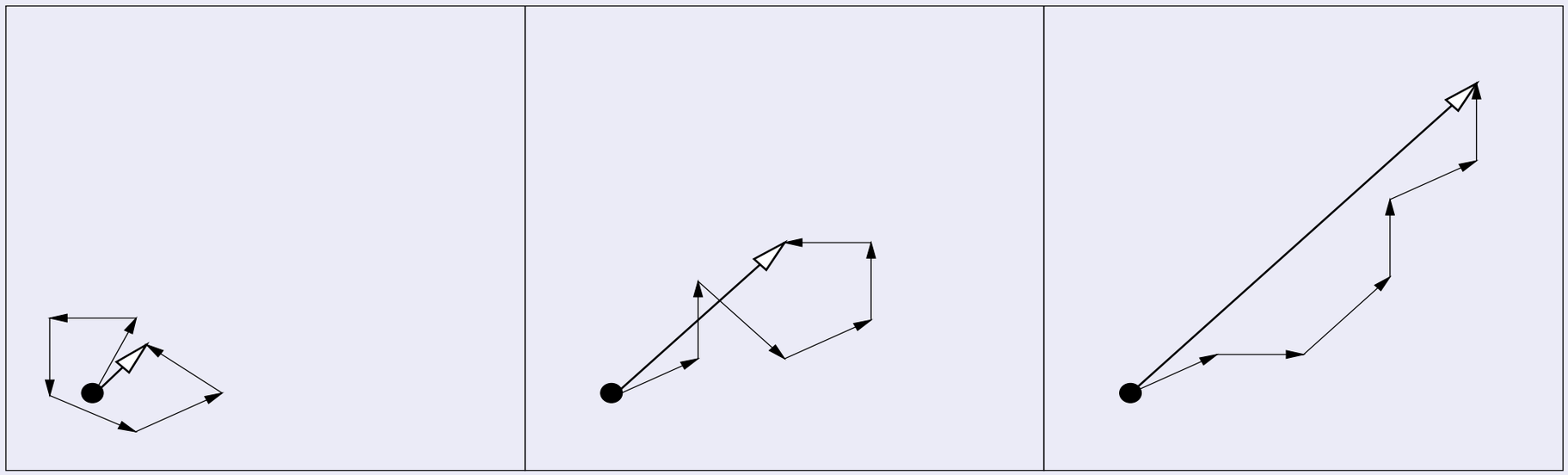
Path Length Control (CSA)

The Concept of Cumulative Step-Size Adaptation

$$\begin{aligned} \boldsymbol{x}_i &= \boldsymbol{m} + \sigma \boldsymbol{y}_i \\ \boldsymbol{m} &\leftarrow \boldsymbol{m} + \sigma \boldsymbol{y}_w \end{aligned}$$

Measure the length of the *evolution path*

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



$\Downarrow$ decrease σ

$\Downarrow$ increase σ

loosely speaking steps are

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

Path Length Control (CSA)

The Equations

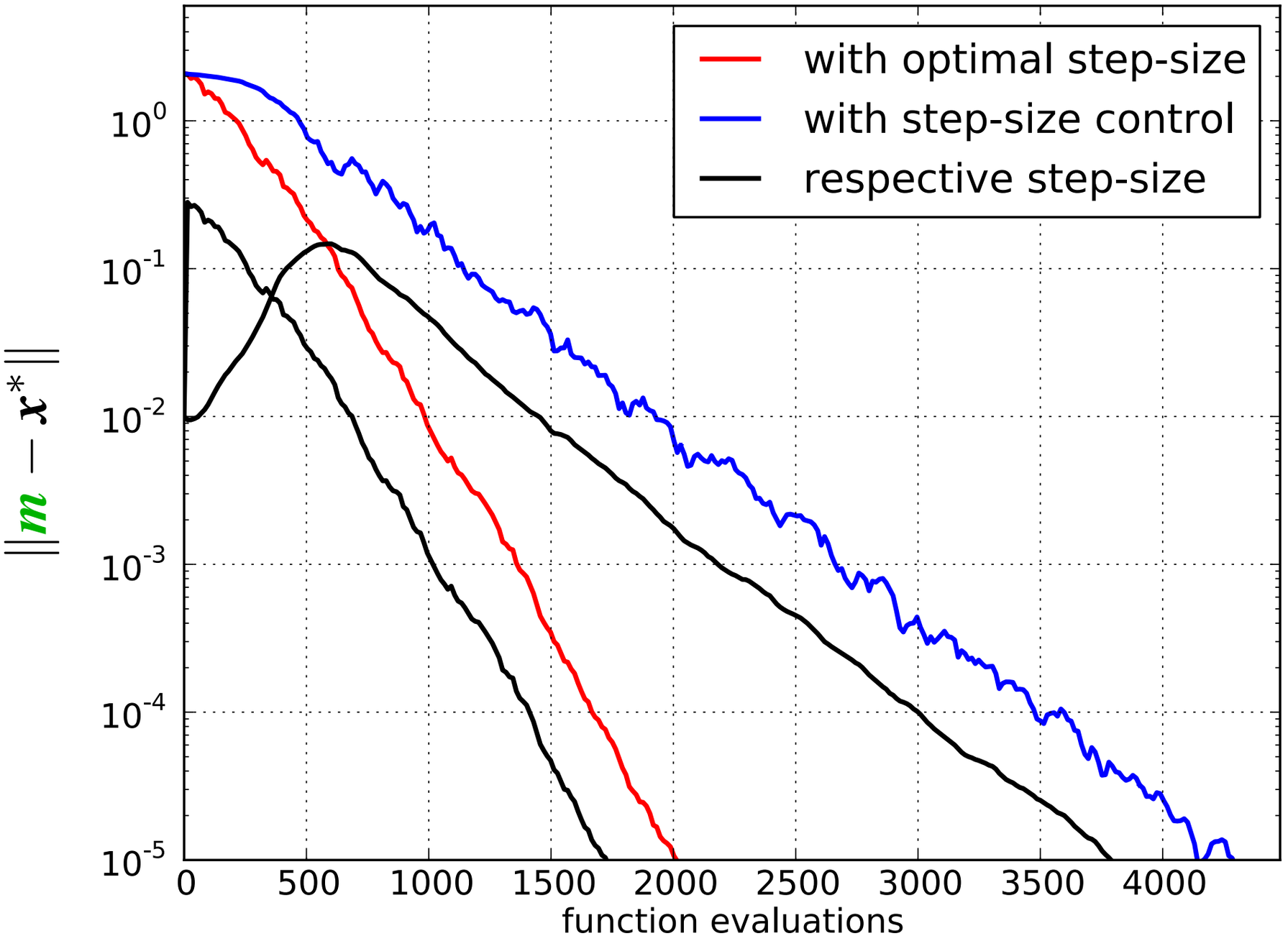
Initialize $\boldsymbol{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, evolution path $\boldsymbol{p}_\sigma = \mathbf{0}$,
set $c_\sigma \approx \mu_w/(n+\mu_w)$, $d_\sigma \approx 1$.

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

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

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

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



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

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

Two-Point Step-Size Adaptation (TPA)

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

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

- Compare the ranking of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ among λ current population

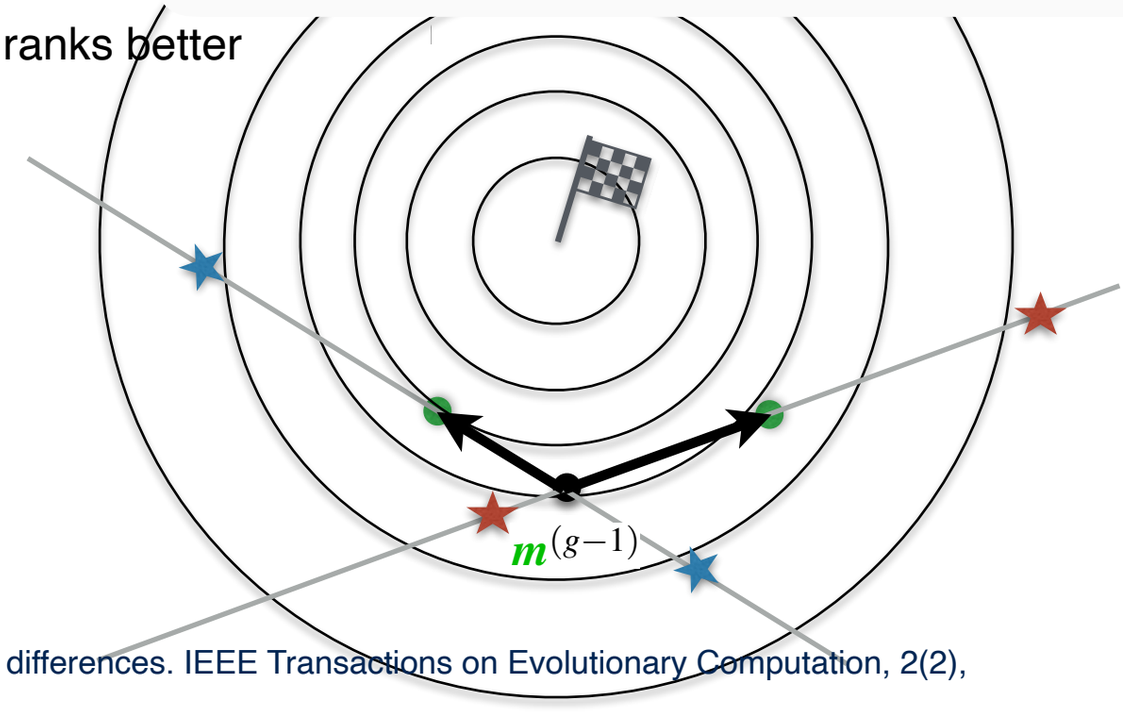
$$s^{(g+1)} = (1 - c_s) s^{(g)} + c_s \underbrace{\frac{\mathrm{rank}(\boldsymbol{x}_2) - \mathrm{rank}(\boldsymbol{x}_1)}{\lambda - 1}}$$

> 0 if forward direction ranks better

- Update the step-size

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

where $c_s \approx 0.3$ and $d_\sigma \approx 1 + 2\ln(n)$



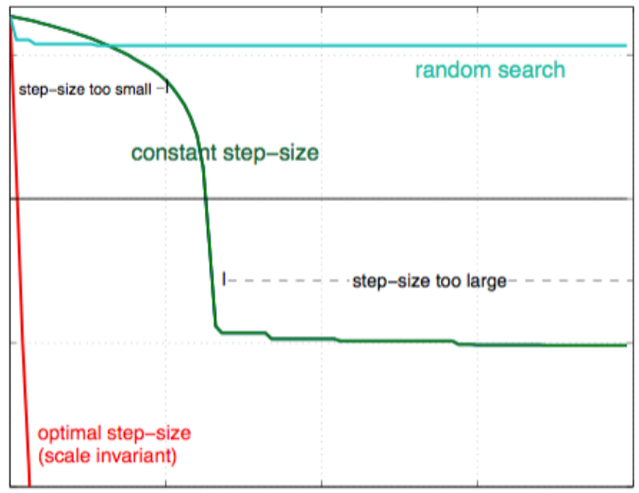
[Salomon, 1998] Salomon, R. (1998). Evolutionary algorithms and gradient search: similarities and differences. IEEE Transactions on Evolutionary Computation, 2(2), pages 45–55.

[Hansen, 2008] Hansen, N. (2008). CMA-ES with two-point step-size adaptation. [research report] rr-6527, 2008. Inria-00276854v5.

[Hansen et al, 2014] Hansen, N., Atamna, A., and Auger, A. (2014). How to assess step-size adaptation mechanisms in randomised search. In *Parallel Problem Solving from Nature*–PPSN XIII, pages 60–69. Springer.

[Akimoto and Hansen, 2020] Akimoto, Y. and Hansen, N. (2020). Diagonal acceleration for covariance matrix adaptation evolution strategies. *Evolutionary computation*, 28(3), pages 405–435.

Step-Size Control: Summary



Why Step-Size Control?

- to achieve linear convergence at near-optimal rate

Cumulative Step-Size Adaptation

- efficient and robust for $\lambda \leq N$
- inefficient on functions with (many) ineffective axes

Alternative Step-Size Adaptation Mechanisms

- Two-Point Step-Size Adaptation
- Median Success Rule[b], Population Success Rule[c]

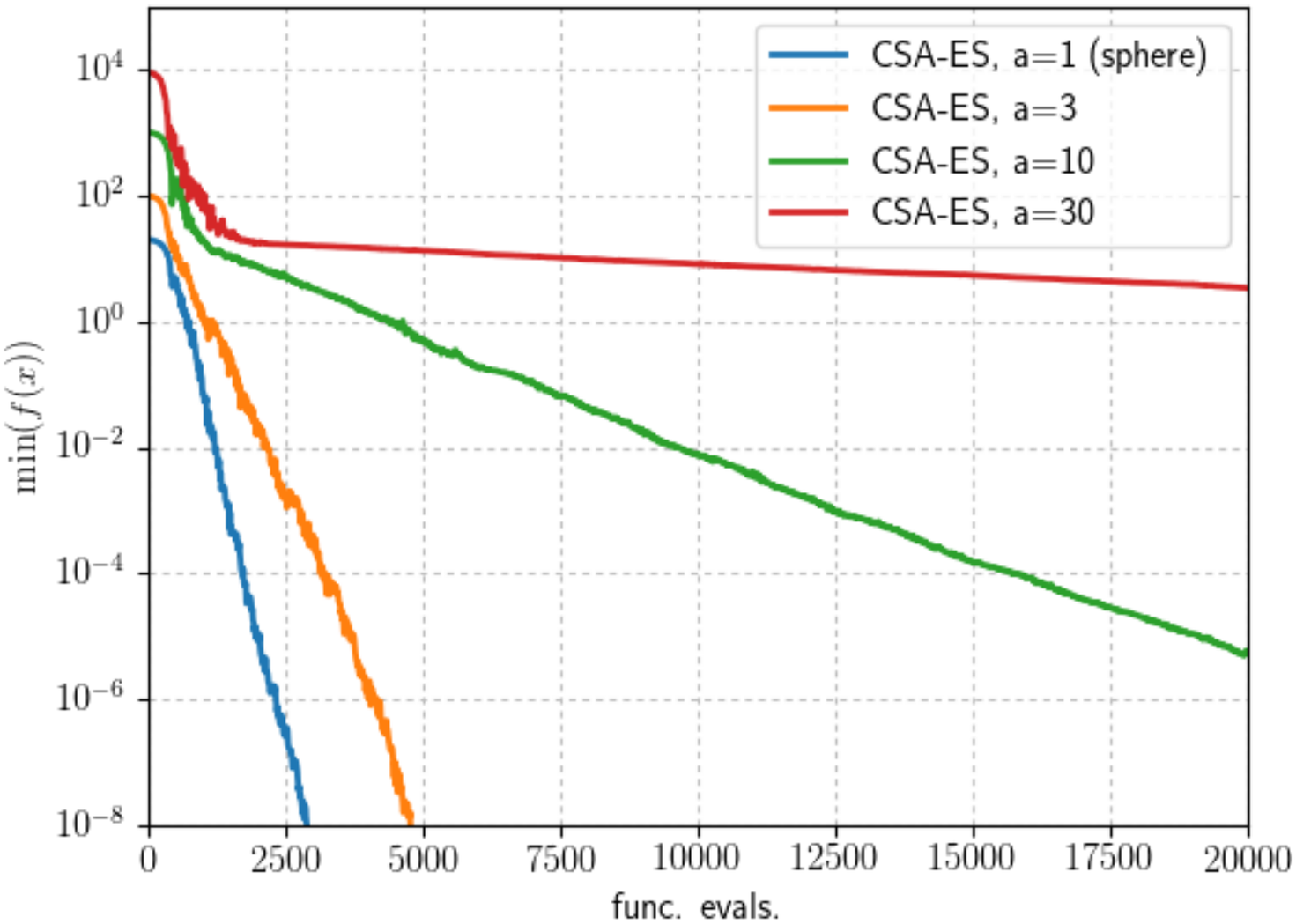
the effective adaptation of the overall population diversity seems yet to pose open questions, in particular with recombination or without entire control over the realized distribution.[a]

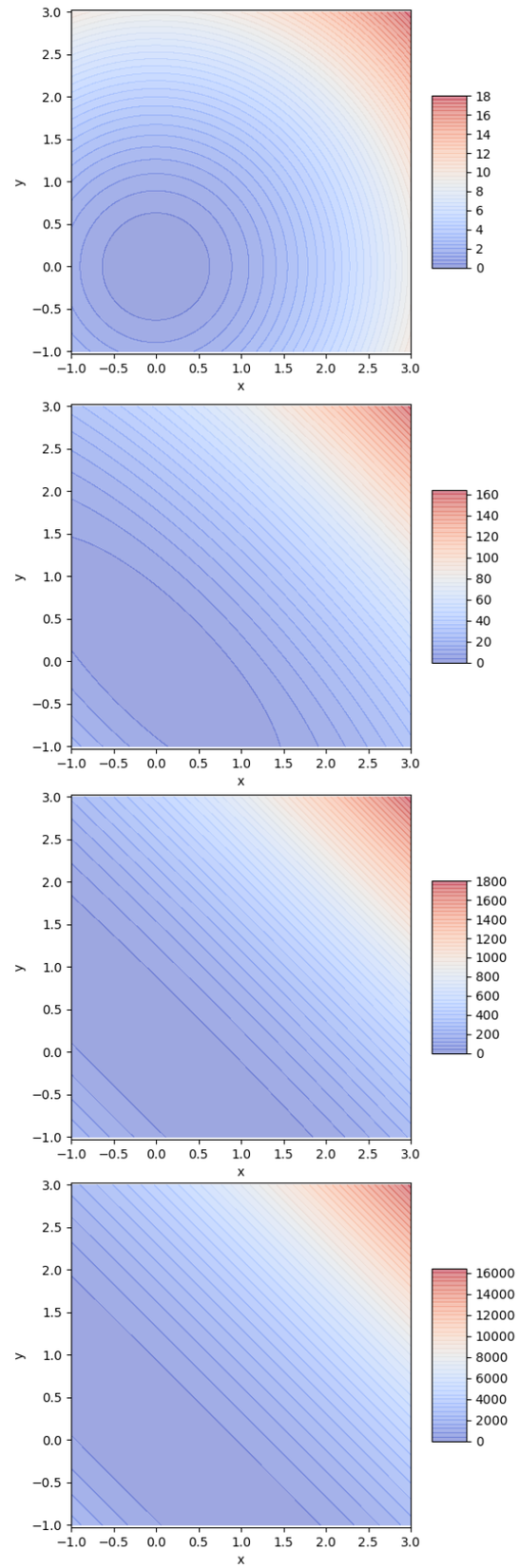
[a] Hansen et al. How to Assess Step-Size Adaptation Mechanisms in Randomised Search. PPSN 2014

[b] Ait Elhara, O., Auger, A., and Hansen, N. (2013). A median success rule for non-elitist evolution strategies: Study of feasibility. In Proc. of the GECCO, pages 415–422.

[c] Loshchilov, I. (2014). A computationally efficient limited memory CMA-ES for large scale optimization. In Proc. of the GECCO, pages 397–404.

Step-Size Control Is Not Enough

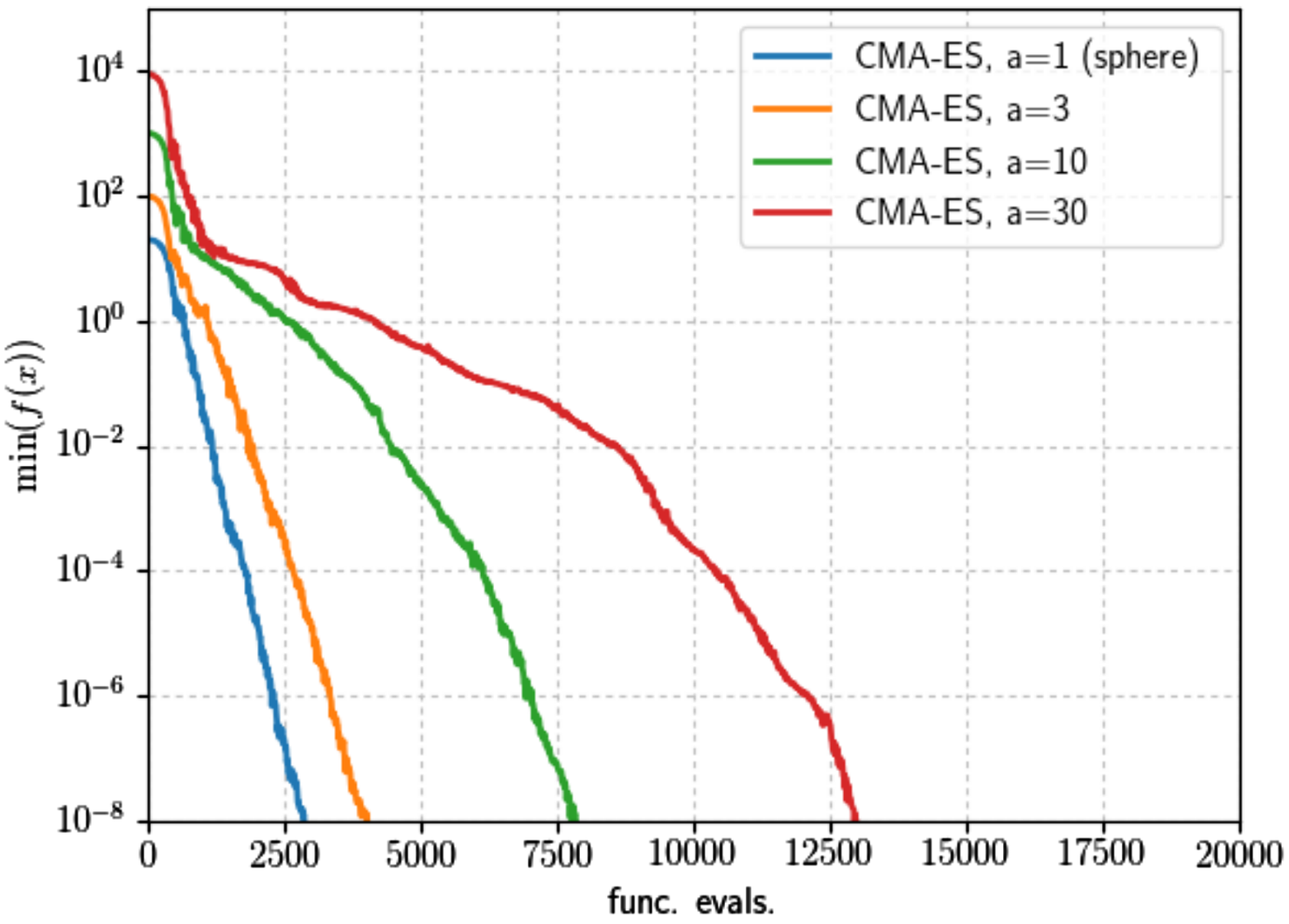


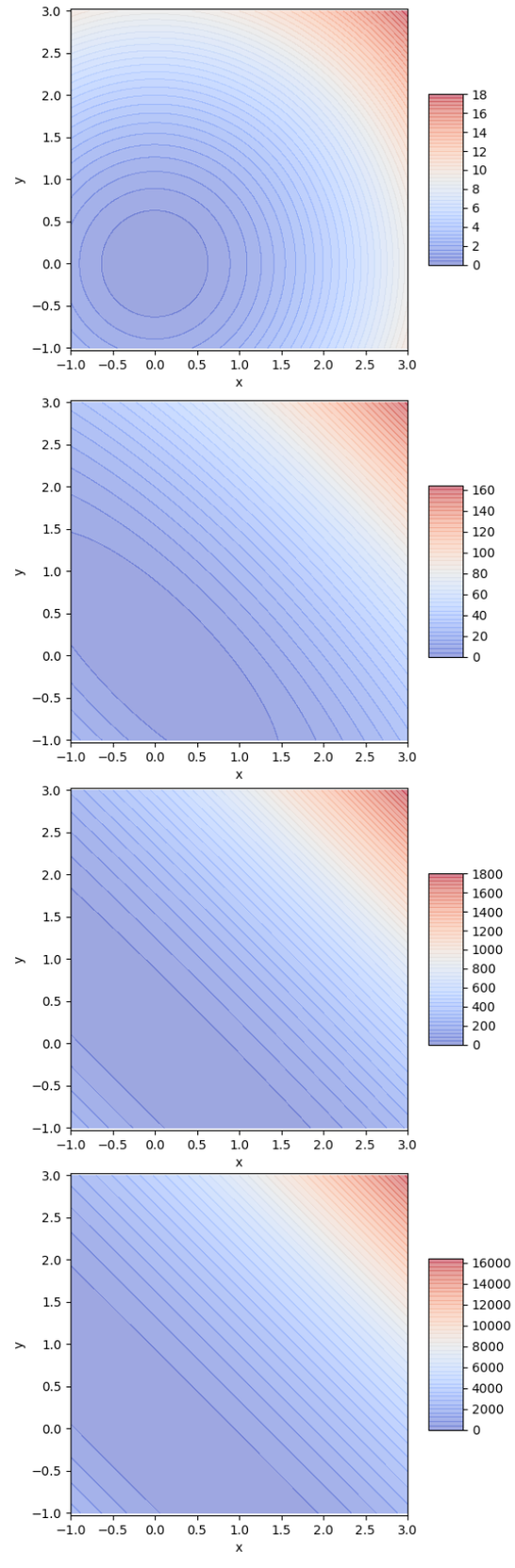


On 20D TwoAxes Function: $f(\mathbf{x}) = \sum_{i=1}^{\lfloor n/2 \rfloor} x_i^2 + a \sum_{i=\lfloor n/2 \rfloor + 1}^{n} x_i^2$

- convergence speed of CSA-ES becomes lower as the function becomes ill conditioned (a^2 becomes greater) $\Rightarrow$ covariance matrix adaptation required

Step-Size Control Is Not Enough





On 20D TwoAxes Function: $f(\mathbf{x}) = \sum_{i=1}^{\lfloor n/2 \rfloor} x_i^2 + a \sum_{i=\lfloor n/2 \rfloor + 1}^{n} x_i^2$

- convergence speed of CSA-ES becomes lower as the function becomes ill conditioned (a^2 becomes greater) $\Rightarrow$ covariance matrix adaptation required

Topics

0. Problem statement

1. What makes an optimization problem difficult to solve?

2. How does the CMA-ES work?

   - Normal Distribution, Rank-Based Recombination
   - Step-Size Adaptation
   - Covariance Matrix Adaptation

3. What can users do?

   - Choice of problem formulation and encoding (not covered)
   - Choice of initial solution and initial step-size
   - Restarts, Increasing Population Size
   - Restricted Covariance Matrix

Topics

0. Problem statement

1. What makes an optimization problem difficult to solve?

2. How does the CMA-ES work?

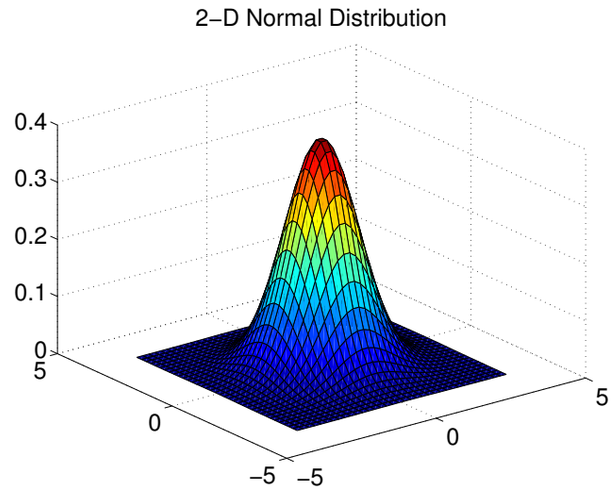
- Normal Distribution, Rank-Based Recombination
- Step-Size Adaptation
- Covariance Matrix Adaptation
  - Rank-One Update and Cumulation
  - Rank-μ Update
  - Active Covariance Update
- Choice of problem formulation and encoding (not covered)
- Choice of initial solution and initial step-size
- Restarts, Increasing Population Size
- Restricted Covariance Matrix

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

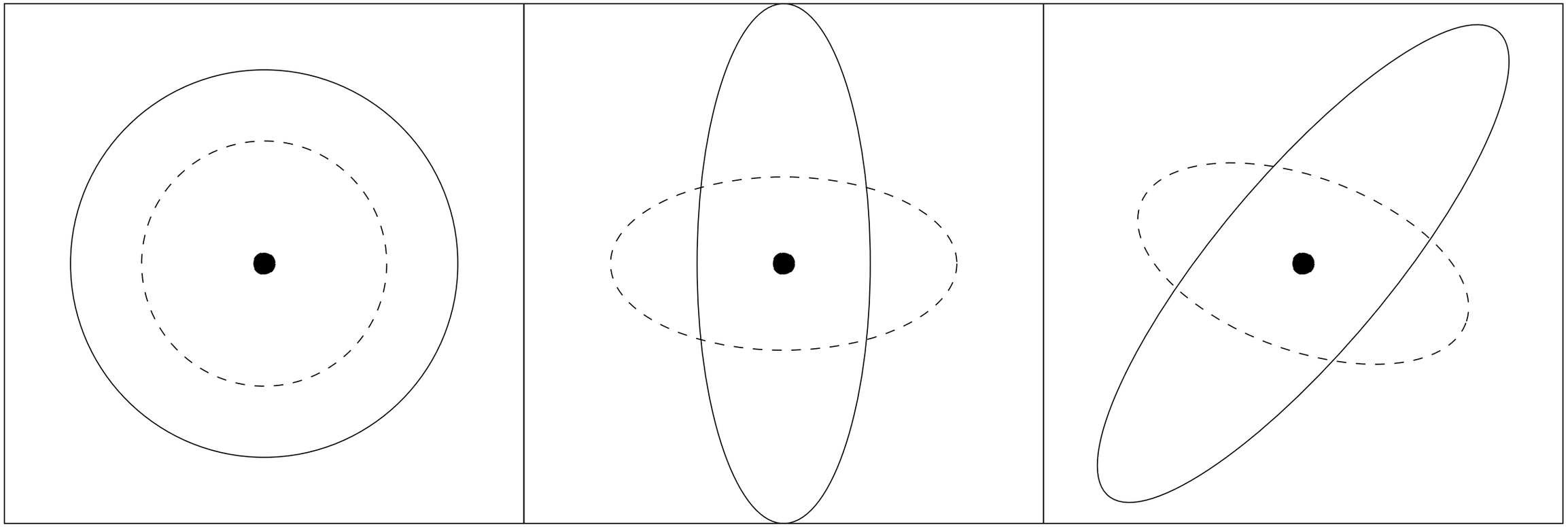


The covariance matrix $\mathbf{C}$

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

. . . any covariance matrix can be uniquely identified with the iso-density ellipsoid
$\{\boldsymbol{x} \in \mathbb{R}^n \mid (\boldsymbol{x} - \boldsymbol{m})^{\mathrm{T}} \mathbf{C}^{-1} (\boldsymbol{x} - \boldsymbol{m}) = n\}$

Lines of Equal Density



$\mathcal{N}\left(\boldsymbol{m}, \sigma^2 \mathbf{I}\right) \sim \boldsymbol{m} + \sigma \mathcal{N}(\mathbf{0}, \mathbf{I})$
one degree of freedom σ
components are
independent and identically
normally distributed

$\mathcal{N}\left(\boldsymbol{m}, \mathbf{D}^2\right) \sim \boldsymbol{m} + \mathbf{D}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$
n degrees of freedom
components are
independent, scaled

$\mathcal{N}(\boldsymbol{m}, \mathbf{C}) \sim \boldsymbol{m} + \mathbf{C}^{\frac{1}{2}} \mathcal{N}(\mathbf{0}, \mathbf{I})$
$(n^2 + n)/2$ degrees of freedom
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 $\mathbf{A}$.

Multivariate Normal Distribution and Eigenvalues

For any positive definite symmetric $\mathbf{C}$,

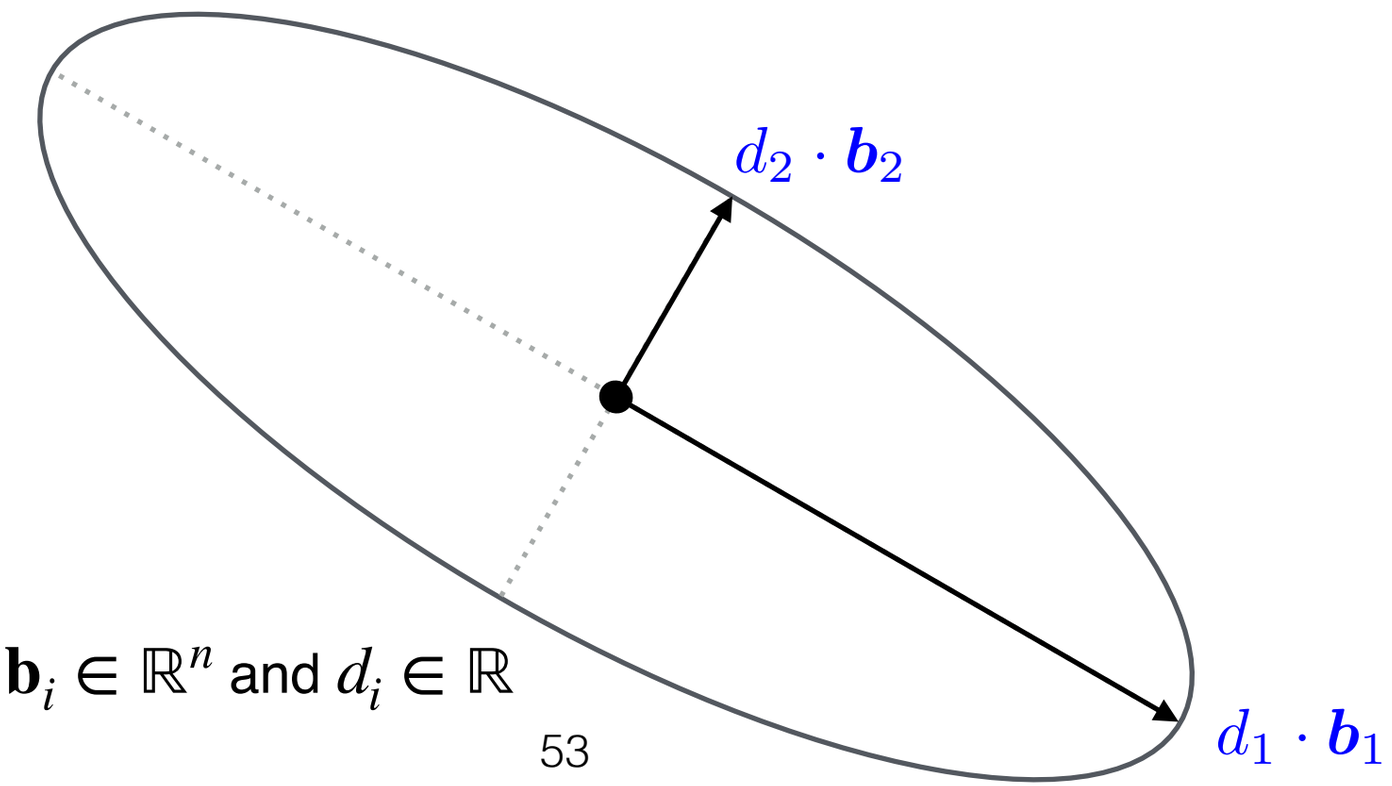
$$\mathbf{C} = d_1^2 \boldsymbol{b}_1 \boldsymbol{b}_1^{\mathrm{T}} + \cdots + d_N^2 \boldsymbol{b}_N \boldsymbol{b}_N^{\mathrm{T}} \qquad (1)$$

d_i: square root of the eigenvalue of $\mathbf{C}$

$\boldsymbol{b}_i$: eigenvector of $\mathbf{C}$, corresponding to d_i

The multivariate normal distribution $\mathcal{N}(\boldsymbol{m}, \mathbf{C})$

$$\mathcal{N}(\boldsymbol{m}, \mathbf{C}) \sim \boldsymbol{m} + \mathcal{N}(0, d_1^2)\boldsymbol{b}_1 + \cdots + \mathcal{N}(0, d_N^2)\boldsymbol{b}_N \qquad (2)$$



(1) $\iff$ (2) holds for any $\mathbf{b}_i \in \mathbb{R}^n$ and $d_i \in \mathbb{R}$

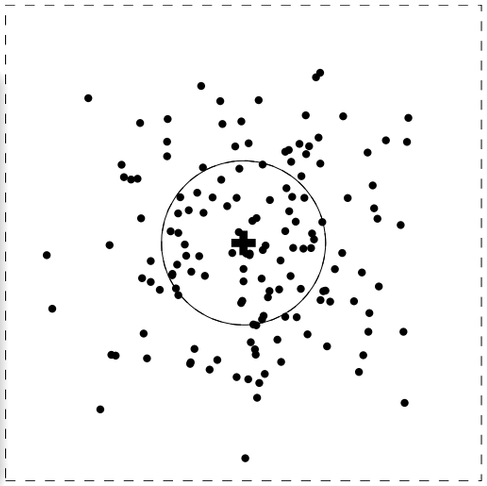
Evolution Strategies

Recalling

New search points are sampled normally distributed

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

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



where

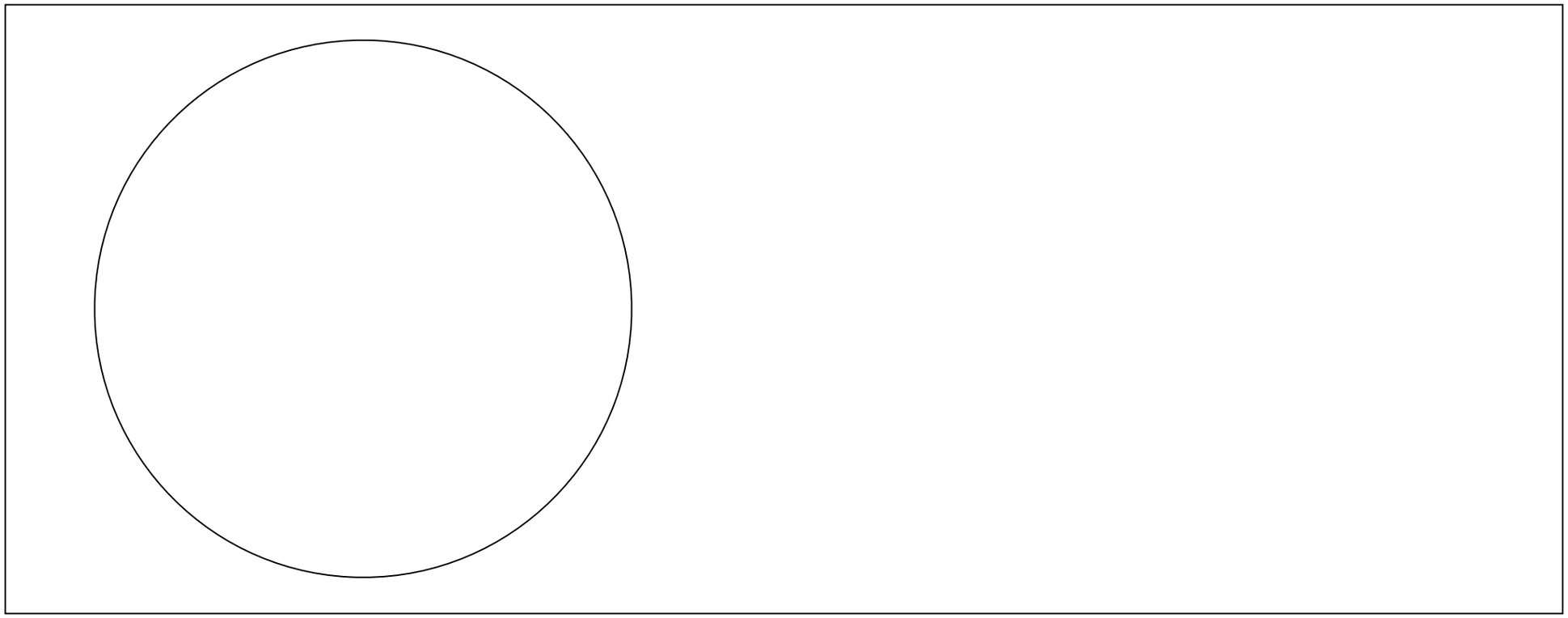
- the mean vector $\boldsymbol{m} \in \mathbb{R}^n$ represents the favorite solution
- the so-called step-size $\sigma \in \mathbb{R}_+$ controls the *step length*
- the covariance matrix $\mathbf{C} \in \mathbb{R}^{n \times n}$ determines the shape of the distribution ellipsoid

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

Covariance Matrix Adaptation

Rank-One Update

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



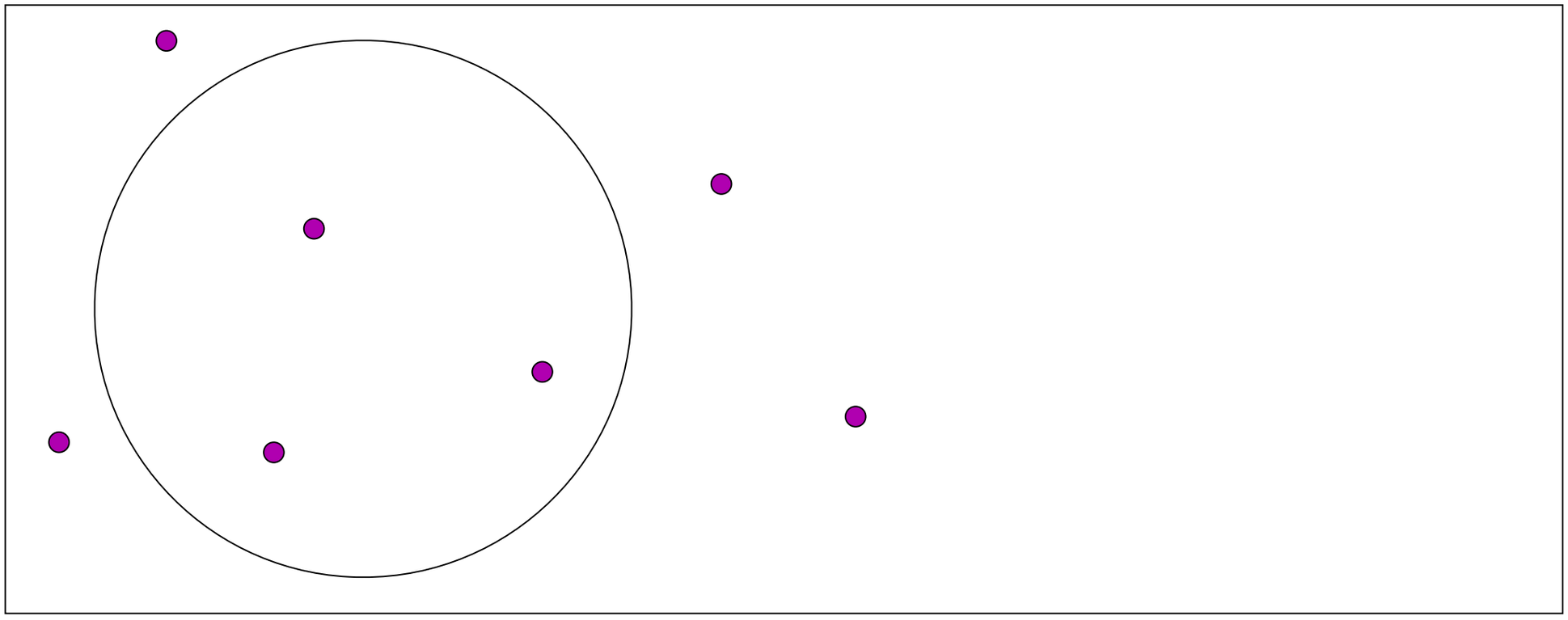
initial distribution, $\mathbf{C} = \mathbf{I}$

. . . equations

Covariance Matrix Adaptation

Rank-One Update

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



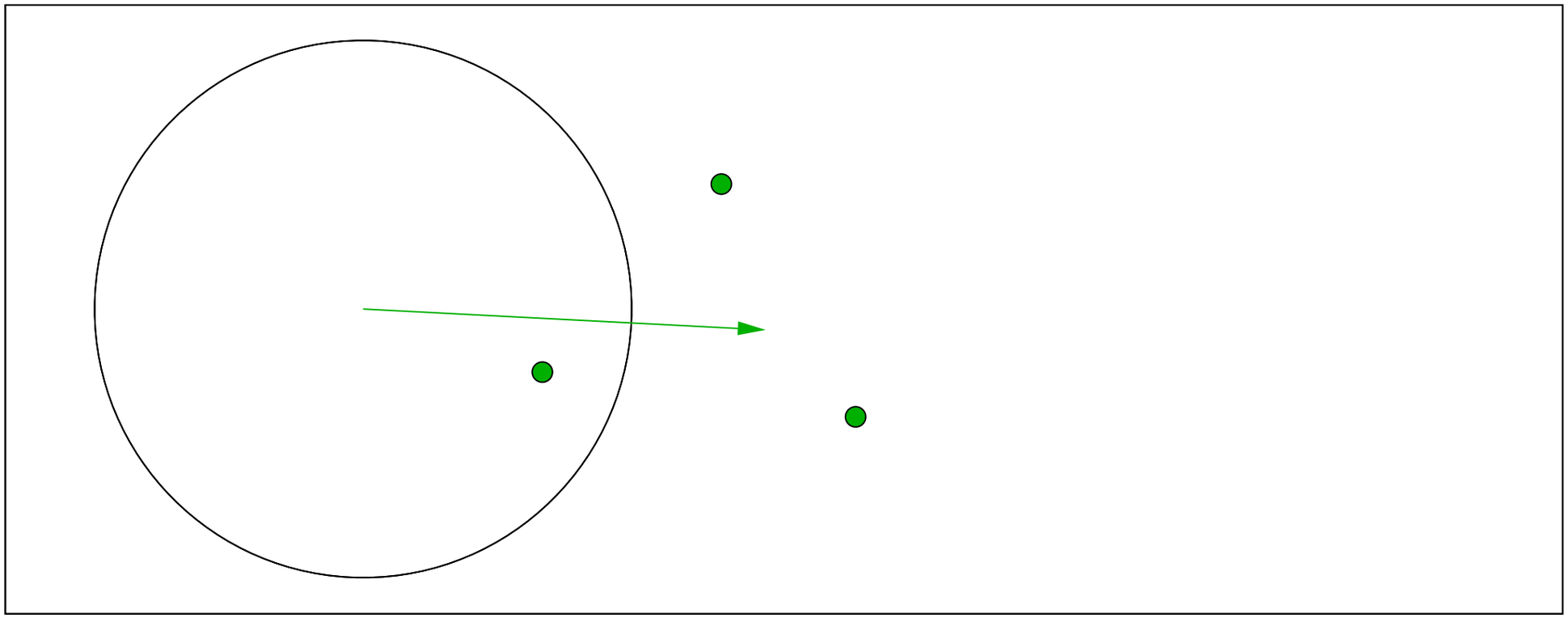
initial distribution, $\mathbf{C} = \mathbf{I}$

. . . equations

Covariance Matrix Adaptation

Rank-One Update

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



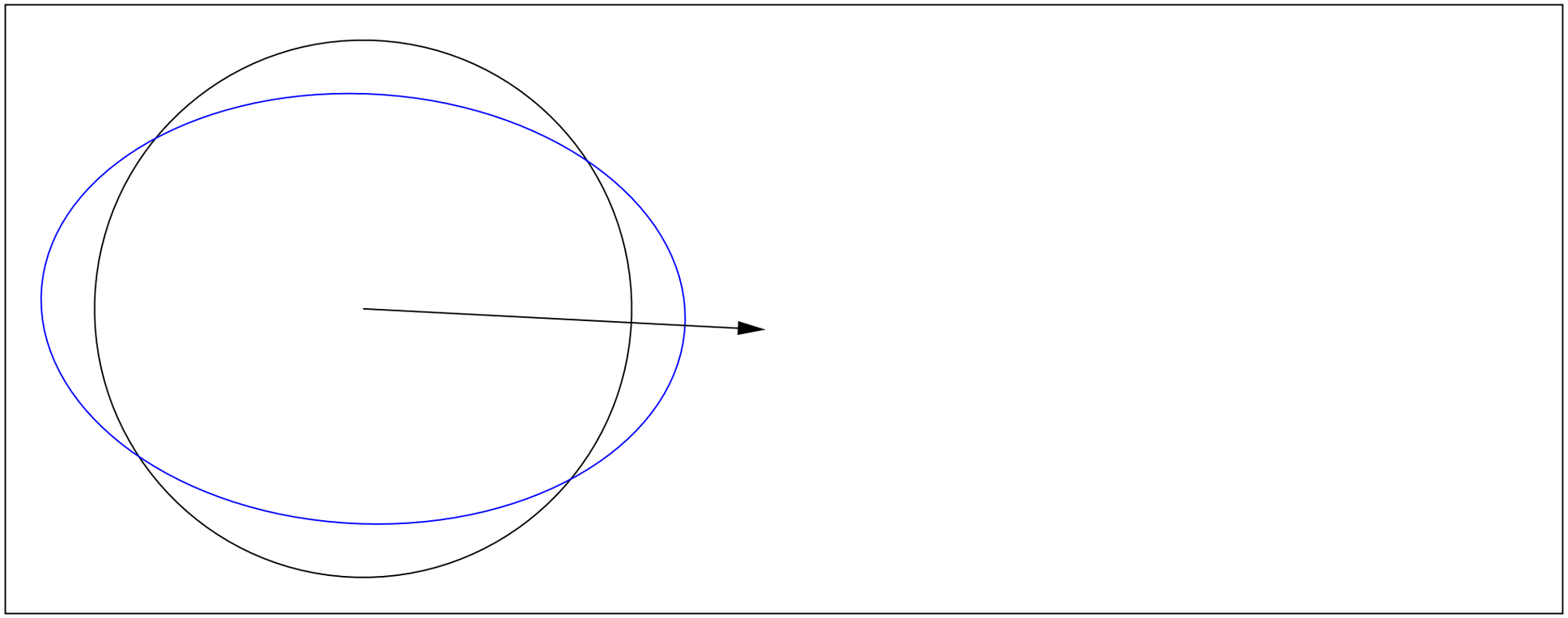
$\boldsymbol{y}_w$, movement of the population mean $\boldsymbol{m}$ (disregarding σ)

. . . equations

Covariance Matrix Adaptation

Rank-One Update

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



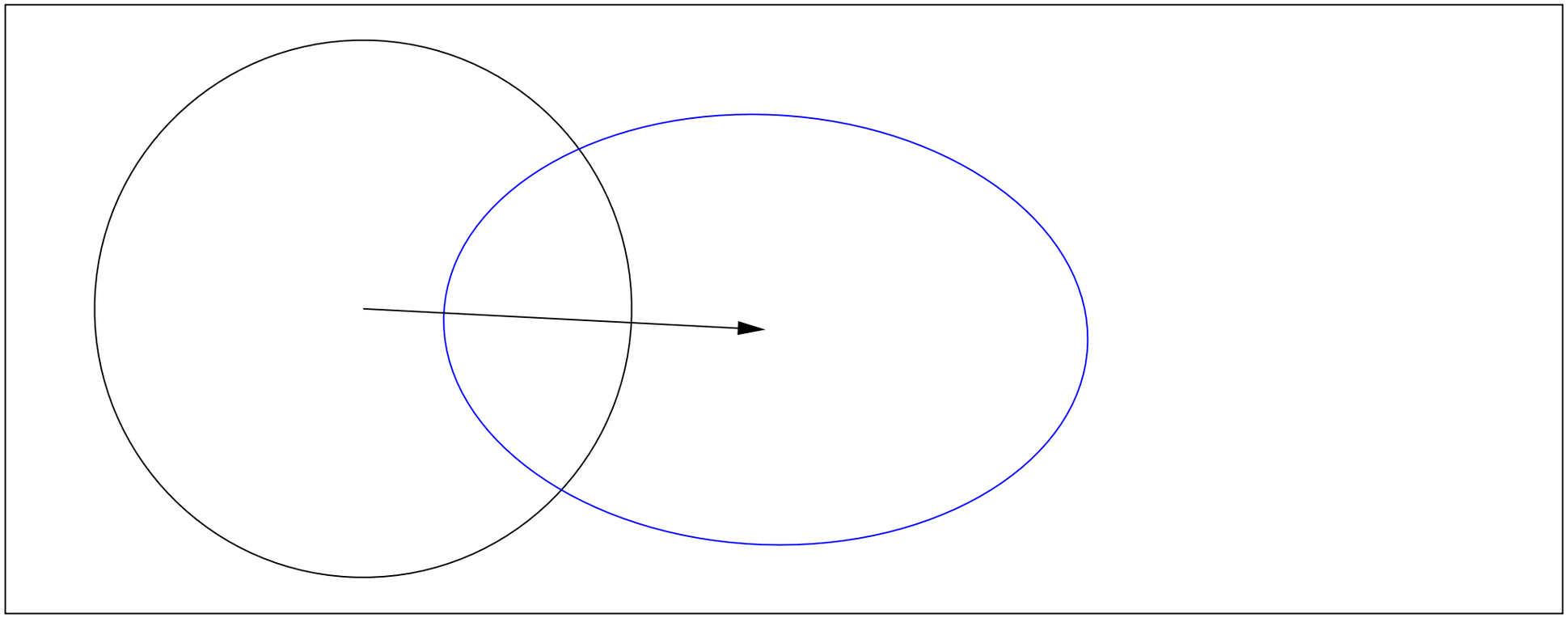
mixture of distribution $\mathbf{C}$ and step $\boldsymbol{y}_w$,
$\mathbf{C} \leftarrow 0.8 \times \mathbf{C} + 0.2 \times \boldsymbol{y}_w \boldsymbol{y}_w^{\mathrm{T}}$

. . . equations

Covariance Matrix Adaptation

Rank-One Update

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



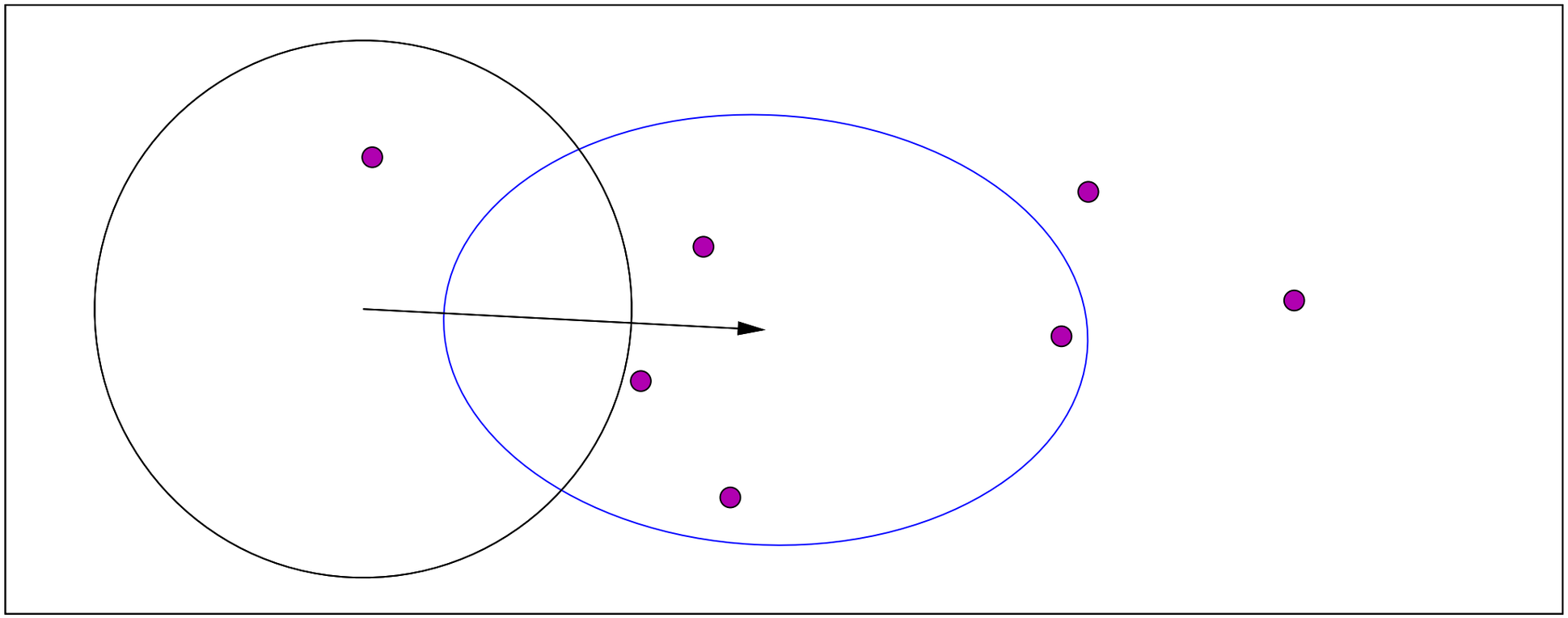
new distribution (disregarding σ)

. . . equations

Covariance Matrix Adaptation

Rank-One Update

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



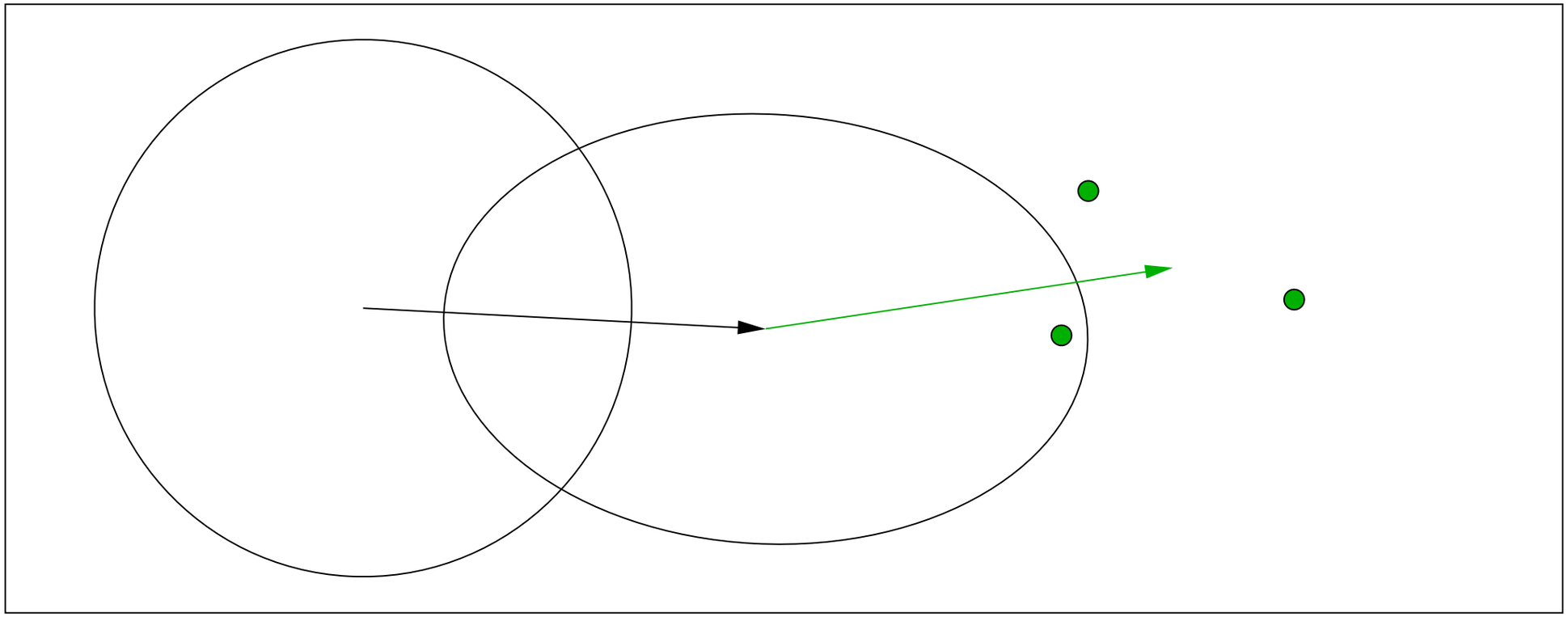
new distribution (disregarding σ)

. . . equations

Covariance Matrix Adaptation

Rank-One Update

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



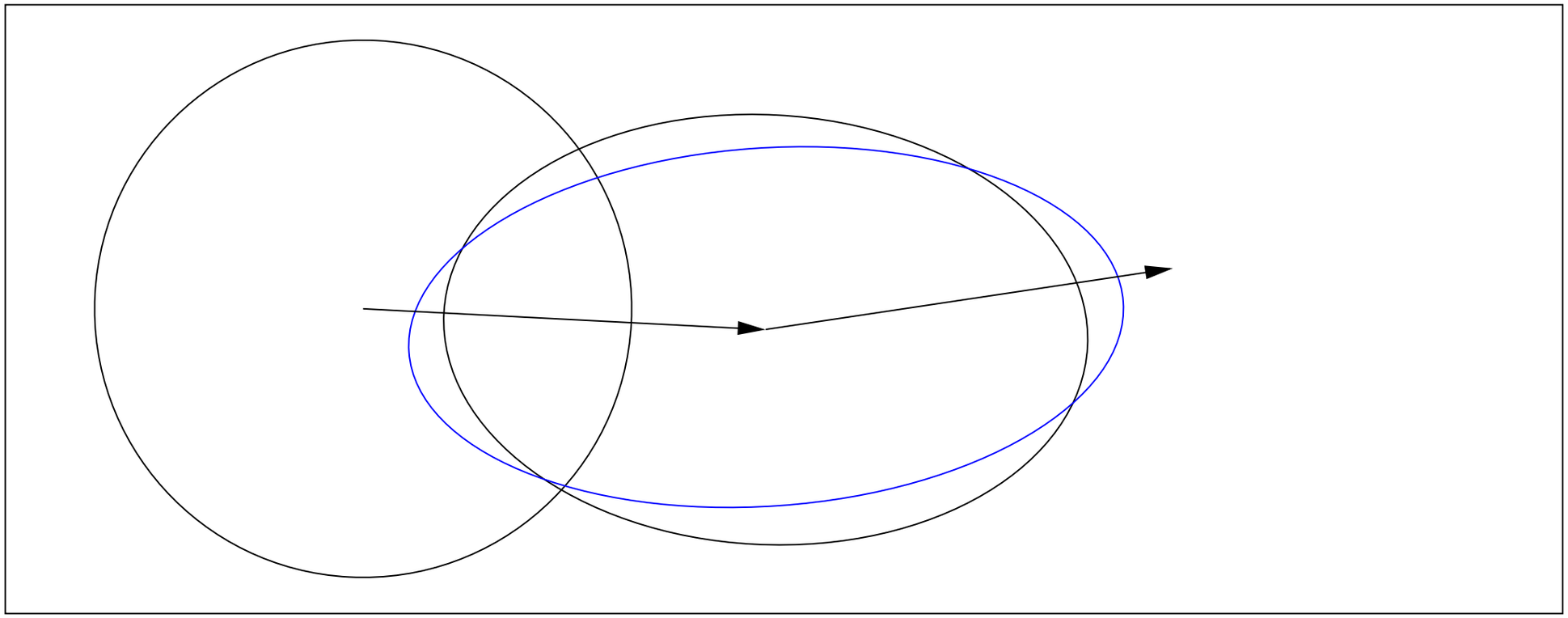
movement of the population mean $\boldsymbol{m}$

. . . equations

Covariance Matrix Adaptation

Rank-One Update

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



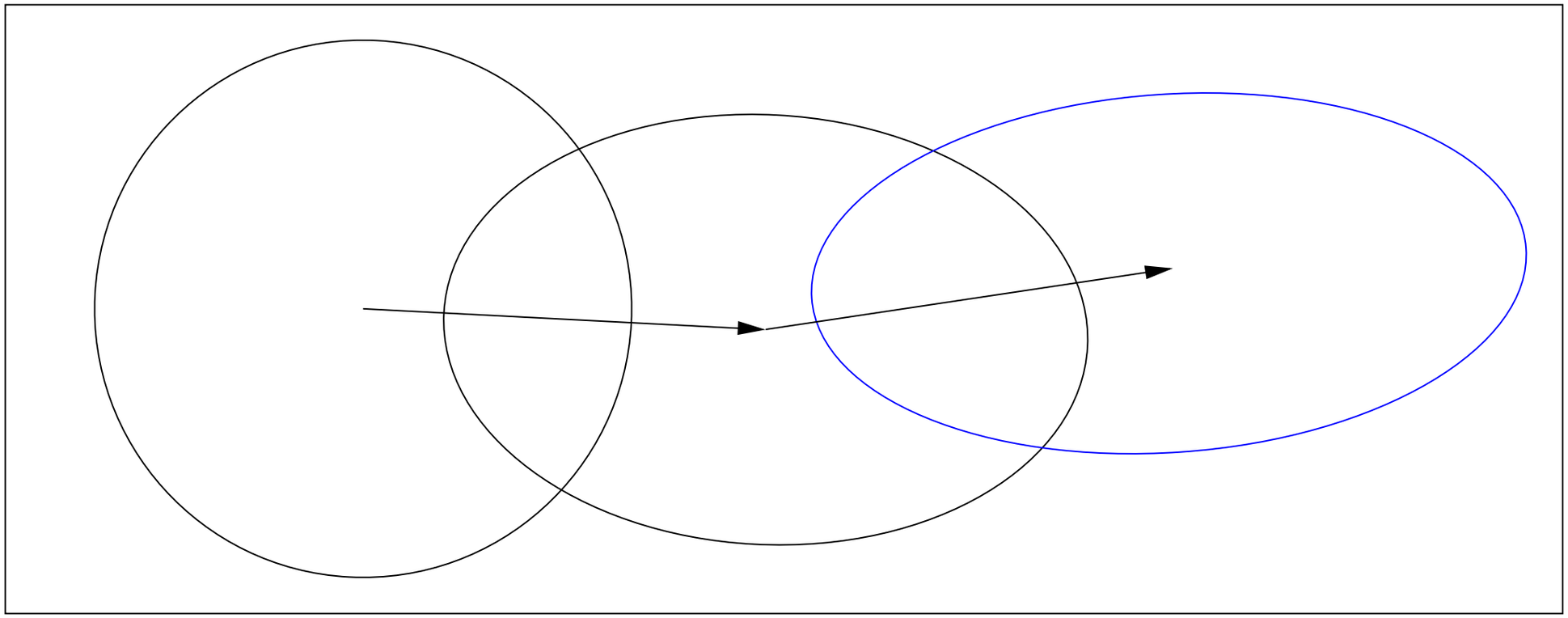
mixture of distribution $\mathbf{C}$ and step $\boldsymbol{y}_w$,
$\mathbf{C} \leftarrow 0.8 \times \mathbf{C} + 0.2 \times \boldsymbol{y}_w \boldsymbol{y}_w^{\mathrm{T}}$

. . . equations

Covariance Matrix Adaptation

Rank-One Update

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



new distribution,

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

the ruling principle: the adaptation increases the likelihood of successful steps, $\boldsymbol{y}_w$, to appear again

another viewpoint: the adaptation follows a natural gradient

approximation of the expected fitness

. . . equations

Covariance Matrix Adaptation

Rank-One Update

Initialize $\boldsymbol{m} \in \mathbb{R}^n$, and $\mathbf{C} = \mathbf{I}$, set $\sigma = 1$, learning rate $c_{\text{cov}} \approx 2/n^2$

While not terminate

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

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

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

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

---

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

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

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

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

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

covariance matrix adaptation

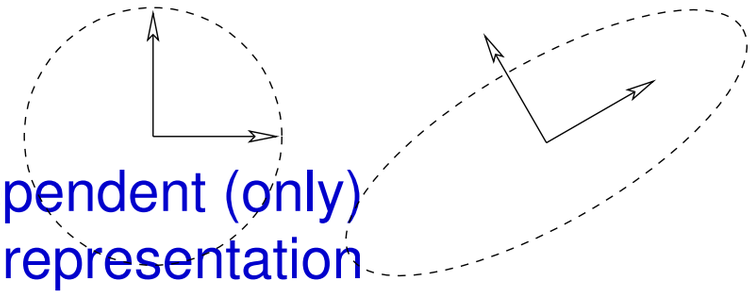
- learns all pairwise dependencies between variables

  off-diagonal entries in the covariance matrix reflect the dependencies

- conducts a principle component analysis (PCA) of steps $\boldsymbol{y}_w$, sequentially in time and space

  eigenvectors of the covariance matrix $\mathbf{C}$ are the principal components / the principal axes of the mutation ellipsoid

- learns a new rotated problem representation[a]



components are independent (only) in the new representation

- learns a new (Mahalanobis) metric

  variable metric method

- approximates the inverse Hessian on quadratic functions

  transformation into the sphere function

- for $\mu = 1$: conducts a natural gradient ascent on the distribution $\mathcal{N}$

  entirely independent of the given coordinate system

[a] invariant under (independently of) search space rotations

Topics

0. Problem statement

1. What makes an optimization problem difficult to solve?

2. How does the CMA-ES work?

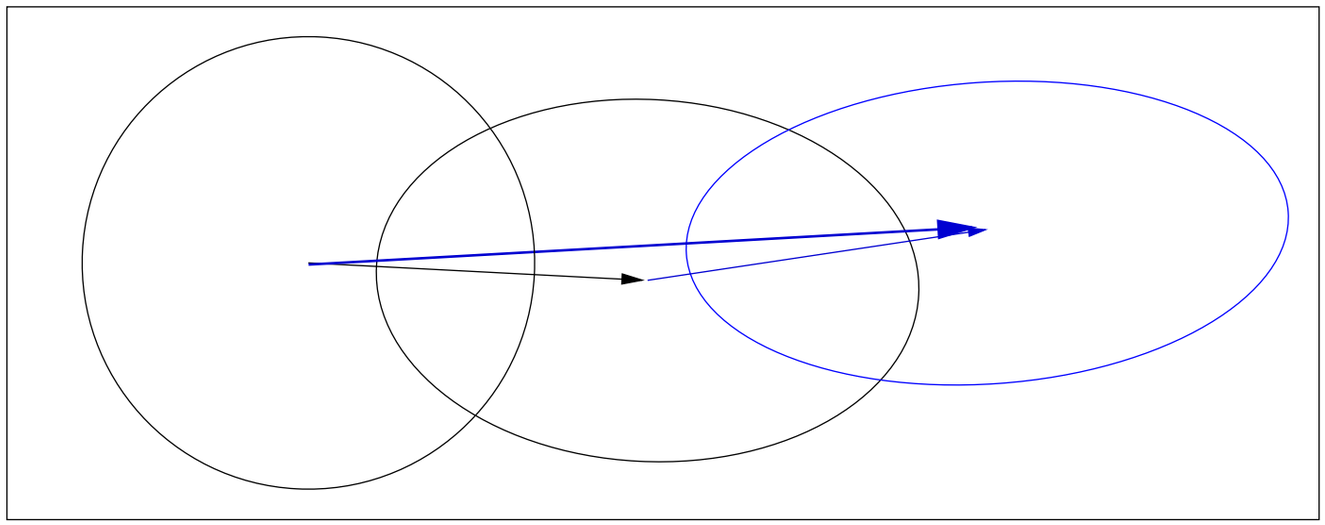
- Normal Distribution, Rank-Based Recombination
- Step-Size Adaptation
- Covariance Matrix Adaptation
  - Rank-One Update and Cumulation
  - Rank-μ Update
  - Active Covariance Update
- Choice of problem formulation and encoding (not covered)
- Choice of initial solution and initial step-size
- Restarts, Increasing Population Size
- Restricted Covariance Matrix

Cumulation

The Evolution Path

Evolution Path

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



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

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

The recursive construction of the evolution path (cumulation):

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

where $\mu_w = \frac{1}{\sum {w_i}^2}$, $c_\mathbf{c} \ll 1$. 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 *moving average*
- *iterate averaging* in stochastic approximation
- *momentum* in the back-propagation algorithm for ANNs
- ...

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

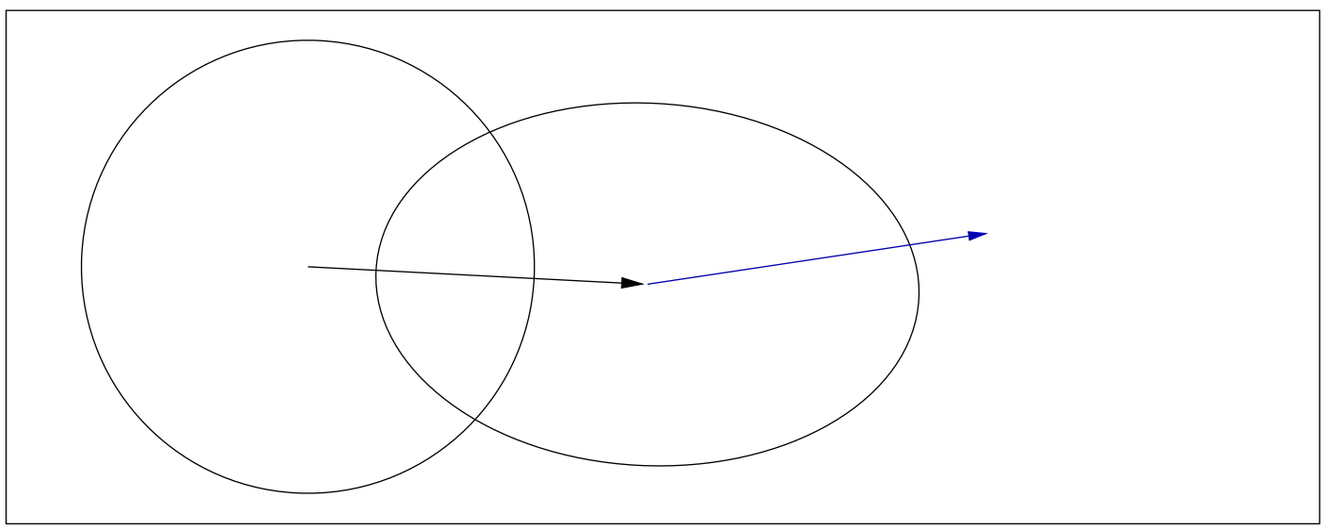
...why?

Cumulation

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

Utilizing the Evolution Path

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



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

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

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

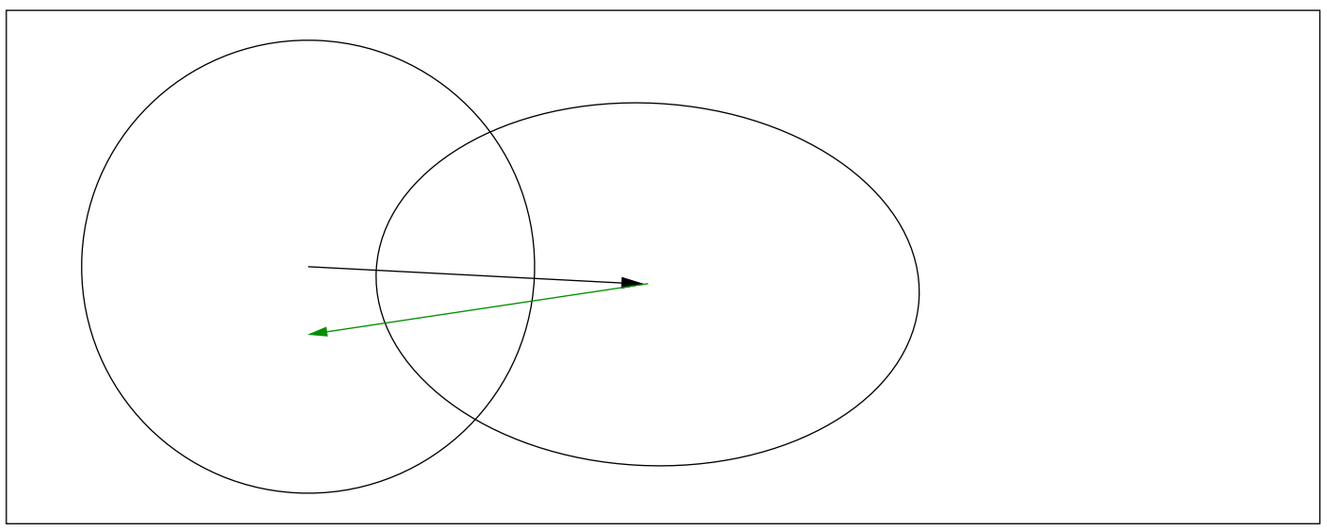
where $\mu_w = \frac{1}{\sum {w_i}^2}$, $c_{\text{cov}} \ll c_{\mathbf{c}} \ll 1$ such that $1/c_{\mathbf{c}}$ is the "backward time horizon".

Cumulation

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

Utilizing the Evolution Path

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



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

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

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

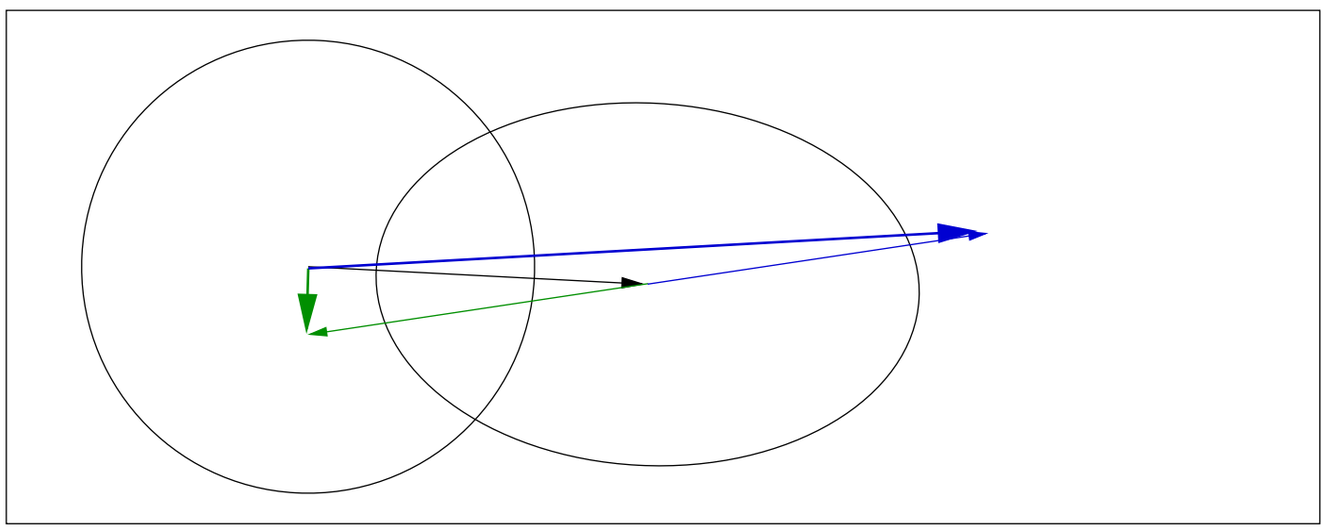
where $\mu_w = \frac{1}{\sum {w_i}^2}$, $c_{\text{cov}} \ll c_{\mathbf{c}} \ll 1$ such that $1/c_{\mathbf{c}}$ is the "backward time horizon".

Cumulation

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

Utilizing the Evolution Path

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



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

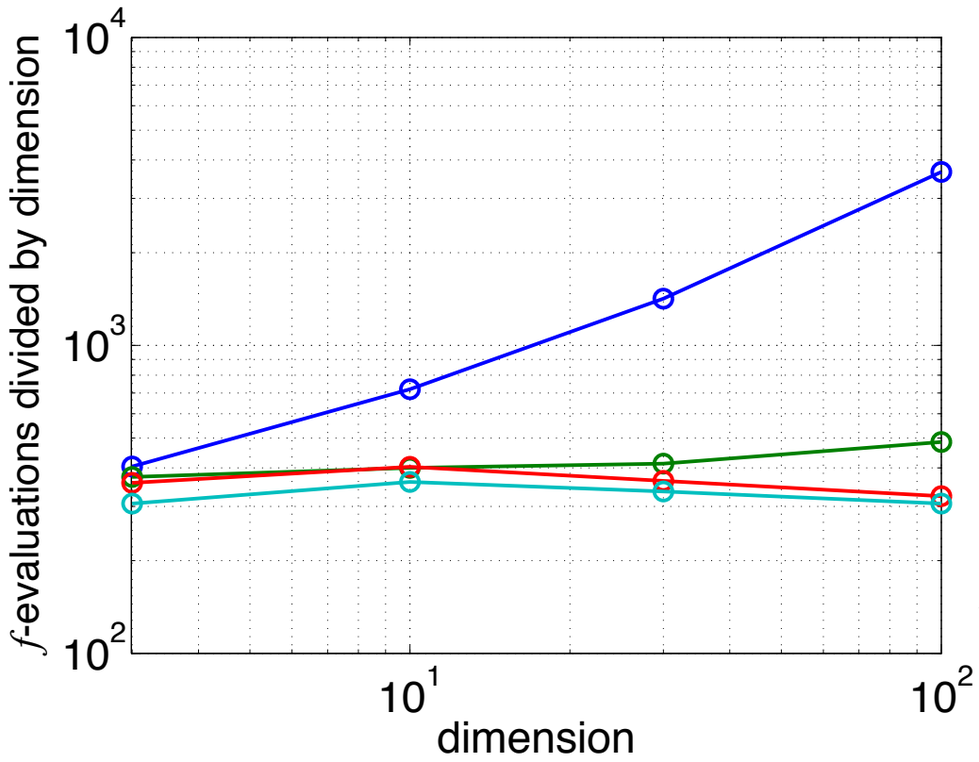
$$\begin{aligned} \boldsymbol{p}_{\mathbf{c}} &\leftarrow \underbrace{(1 - c_{\mathbf{c}})}_{\text{decay factor}} \boldsymbol{p}_{\mathbf{c}} + \underbrace{\sqrt{1 - (1 - c_{\mathbf{c}})^2}\sqrt{\mu_w}}_{\text{normalization factor}} \boldsymbol{y}_w \\ \mathbf{C} &\leftarrow (1 - c_{\text{cov}})\mathbf{C} + c_{\text{cov}} \underbrace{\boldsymbol{p}_{\mathbf{c}} \boldsymbol{p}_{\mathbf{c}}^{\mathrm{T}}}_{\text{rank-one}} \end{aligned}$$

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

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

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

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



$c_c = 1$ (no cumulation)

$c_c = 1/\sqrt{n}$

$c_c = 1/n,\ 3/(n+3)$

with CSA-ES (without CMA): $\approx 3 \times 10^7 n$ evaluations

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

Topics

0. Problem statement

1. What makes an optimization problem difficult to solve?

2. How does the CMA-ES work?

- Normal Distribution, Rank-Based Recombination
- Step-Size Adaptation
- Covariance Matrix Adaptation
  - Rank-One Update and Cumulation
  - Rank-μ Update
  - Active Covariance Update
- Choice of problem formulation and encoding (not covered)
- Choice of initial solution and initial step-size
- Restarts, Increasing Population Size
- Restricted Covariance Matrix

Rank-μ Update

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

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 weighted empirical covariance matrix

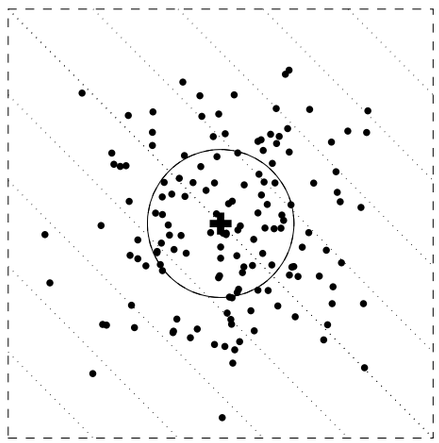
$$\mathbf{C}_\mu = \sum_{i=1}^{\mu} w_i\, \boldsymbol{y}_{i:\lambda} \boldsymbol{y}_{i:\lambda}^{\mathrm{T}}$$

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

The rank-μ update then reads

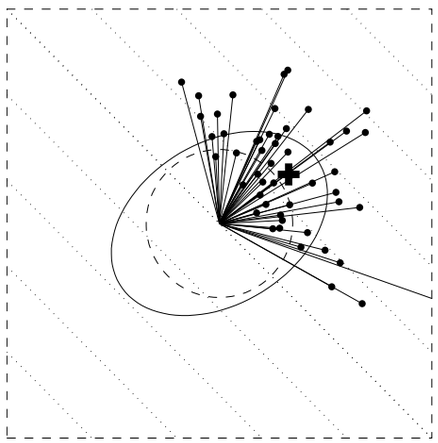
$$\mathbf{C} \leftarrow (1 - c_{\mathrm{cov}})\, \mathbf{C} + c_{\mathrm{cov}}\, \mathbf{C}_\mu$$

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



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

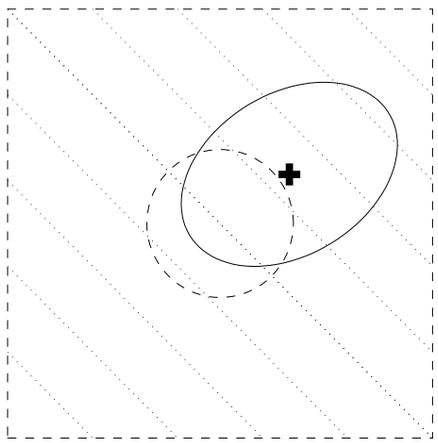
sampling of $\lambda = 150$ solutions where $\mathbf{C} = \mathbf{I}$ and $\sigma = 1$



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

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

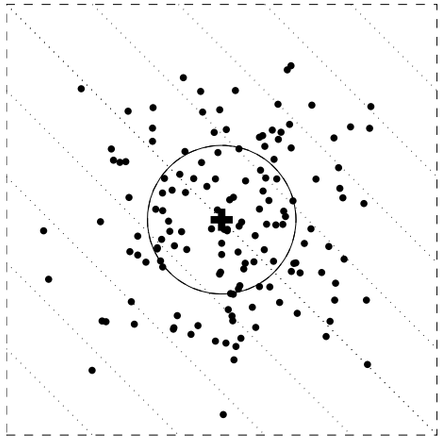
calculating $\mathbf{C}$ where $\mu = 50$, $w_1 = \cdots = w_\mu = \frac{1}{\mu}$, and $c_{\mathrm{cov}} = 1$



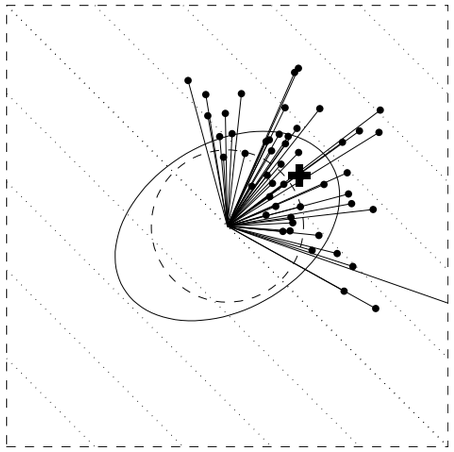
$$\boldsymbol{m}_{\mathrm{new}} \leftarrow \boldsymbol{m} + \frac{1}{\mu} \sum \boldsymbol{y}_{i:\lambda}$$

new distribution

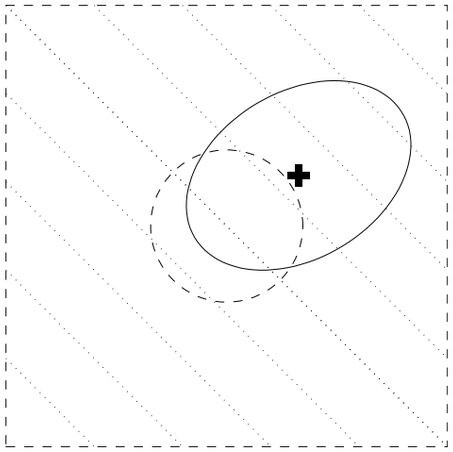
Rank-μ CMA versus Estimation of Multivariate Normal Algorithm $EMNA_{global}$[11]



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

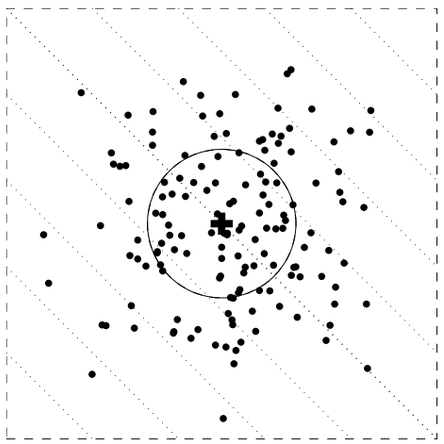


$\mathbf{C} \leftarrow \frac{1}{\mu} \sum (x_{i:\lambda} - m_{\text{old}})(x_{i:\lambda} - m_{\text{old}})^{\mathrm{T}}$

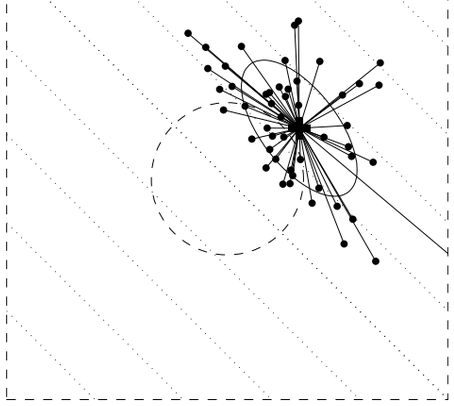


$m_{\text{new}} = m_{\text{old}} + \frac{1}{\mu} \sum y_{i:\lambda}$

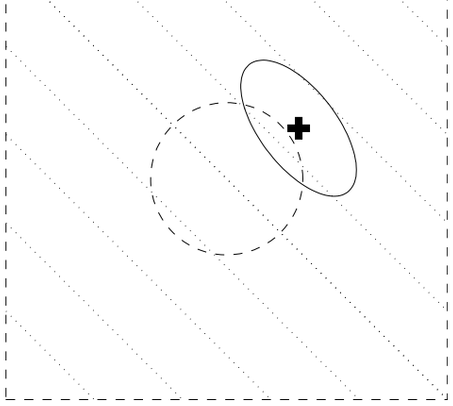
rank-μ CMA conducts a PCA of steps



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



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



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

$EMNA_{global}$ conducts a PCA of points

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

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

new distribution

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

---

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

The rank-μ update

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

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

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 $\mathcal{O}(n)$.

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

. . . all equations

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

The rank-μ update

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

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

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 $\mathcal{O}(n)$.

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

. . . all equations

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

The rank-μ update

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

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

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 $\mathcal{O}(n)$.

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

. . . all equations

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

Active Update

utilize negative weights [Jastrebski and Arnold, 2006]

$$\boldsymbol{C} \leftarrow (1 - c_1 - c_\mu + c_\mu^-)\boldsymbol{C} + c_1 \boldsymbol{p_c}\boldsymbol{p_c}^T + c_\mu \sum_{i=1}^{\lfloor \lambda/2 \rfloor} w_i \boldsymbol{y}_{i:\lambda}\boldsymbol{y}_{i:\lambda}^{\mathrm{T}} \overbrace{- c_\mu^- \sum_{i=\lambda-\mu}^{\lambda} |w_i| \boldsymbol{y}_{i:\lambda}\boldsymbol{y}_{i:\lambda}^{\mathrm{T}}}^{\text{decreasing the variances in unpromising directions}}$$

- $-|w_i| < 0$ (for $i \geq \lambda - \mu$): negative weight assigned to $\boldsymbol{y}_{i:\lambda}$, $\sum_{i=\lambda-\mu}^{\lambda} |w_i| \approx 1$
- $c_\mu^- > 0$: learning rate for the active update

- increases the variance in the directions of $\boldsymbol{p_c}$ and promising steps $\boldsymbol{y}_{i:\lambda}$ $(i \leq \lfloor \lambda/2 \rfloor)$
- decrease the variance in the directions of unpromising steps $\boldsymbol{y}_{i:\lambda}$ $(i \geq \lambda - \lfloor \lambda/2 \rfloor + 1)$
- keep the variance in the subspace orthogonal to the above

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

Input: $\boldsymbol{m} \in \mathbb{R}^n$; $\sigma \in \mathbb{R}_+$; $\lambda \in \mathbb{N}_{\geq 2}$, usually $\lambda \geq 5$, default $4 + \lfloor 3 \log n \rfloor$

Set $c_m = 1$; $c_1 \approx 2/n^2$; $c_\mu \approx \mu_w/n^2$; $c_\mathrm{c} \approx 4/n$; $c_\sigma \approx 1/\sqrt{n}$; $d_\sigma \approx 1$; $w_{i=1\ldots\lambda}$ decreasing in i and $\sum_i^\mu w_i = 1$, $w_\mu > 0 \geq w_{\mu+1}$, $\mu_w^{-1} := \sum_{i=1}^\mu w_i^2 \approx 3/\lambda$

Initialize $\mathbf{C} = \mathbf{I}$, and $\boldsymbol{p}_\mathrm{c} = \mathbf{0}$, $\boldsymbol{p}_\sigma = \mathbf{0}$

While not *terminate*

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

$$\boldsymbol{m} \leftarrow \boldsymbol{m} + c_m \sigma \boldsymbol{y}_w, \quad \text{where } \boldsymbol{y}_w = \sum_{i=1}^\mu w_i \, \boldsymbol{y}_{\mathrm{rk}^{-1}(i)} \qquad \text{update mean}$$

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

$$\boldsymbol{p}_\mathrm{c} \leftarrow (1 - c_\mathrm{c})\, \boldsymbol{p}_\mathrm{c} + \mathbb{1}_{[0,2n]}\left\{\|\boldsymbol{p}_\sigma\|^2\right\} \sqrt{1 - (1 - c_\mathrm{c})^2}\, \sqrt{\mu_w}\, \boldsymbol{y}_w \qquad \text{path for } \mathbf{C}$$

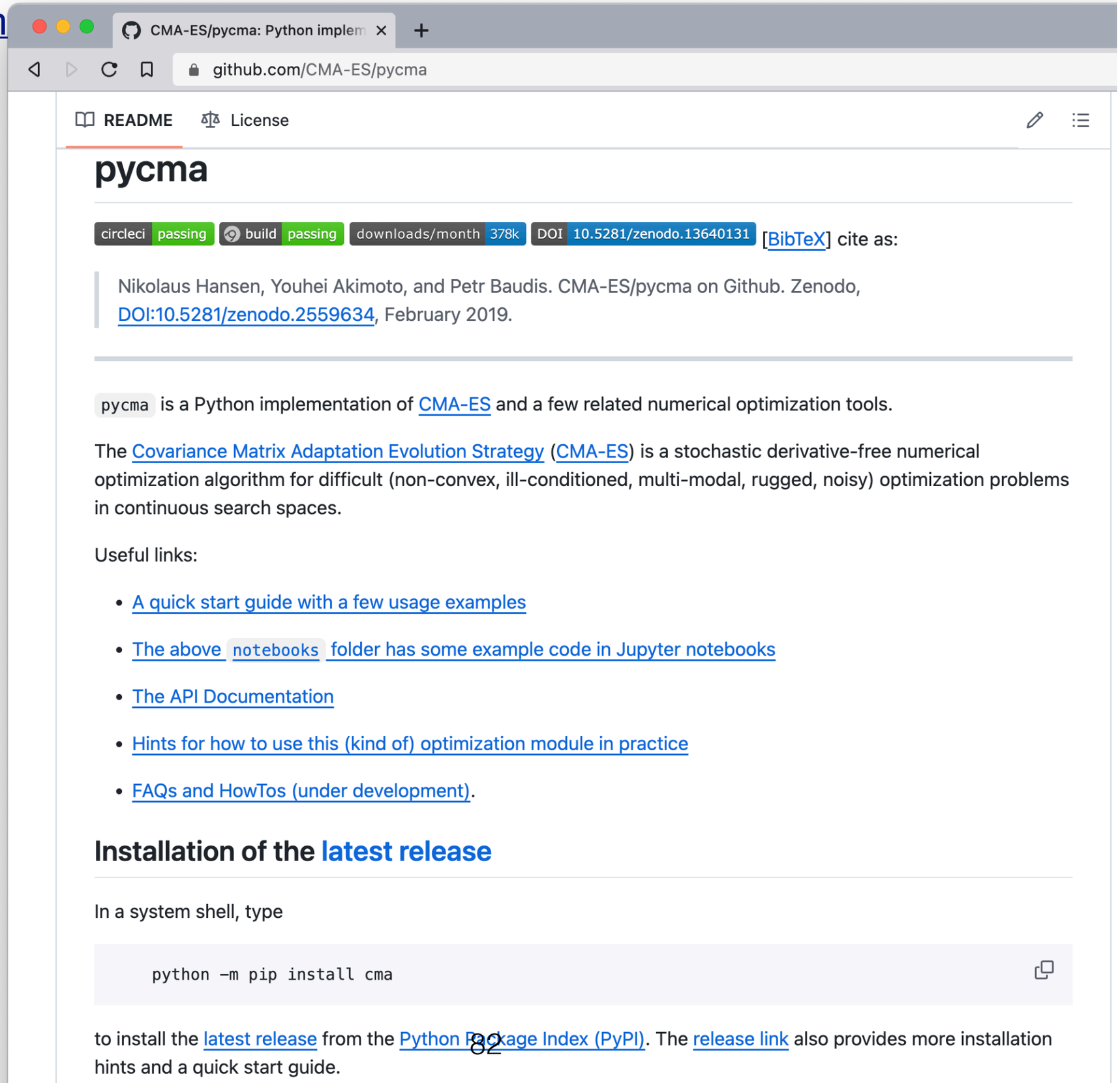
$$\sigma \leftarrow \sigma \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_\sigma\|}{\mathsf{E}\|\mathcal{N}(\mathbf{0},\mathbf{I})\|} - 1\right)\right) \qquad \text{update of } \sigma$$

$$\mathbf{C} \leftarrow \mathbf{C} + c_\mu \sum_{i=1}^\lambda w_{\mathrm{rk}(i)} \left(\boldsymbol{y}_i \boldsymbol{y}_i^\mathsf{T} - \mathbf{C}\right) + c_1 \left(\boldsymbol{p}_\mathrm{c}\, \boldsymbol{p}_\mathrm{c}^\mathsf{T} - \mathbf{C}\right) \qquad \text{update } \mathbf{C}$$

Not covered: termination, restarts, useful output, search boundaries and encoding, corrections for: positive definiteness guaranty, $\boldsymbol{p}_\mathrm{c}$ variance loss, c_σ and d_σ for large λ

Python CMA-ES Implementation

https://github.com



Python CMA-ES Demo

https://github.com/CMA-ES/pycma

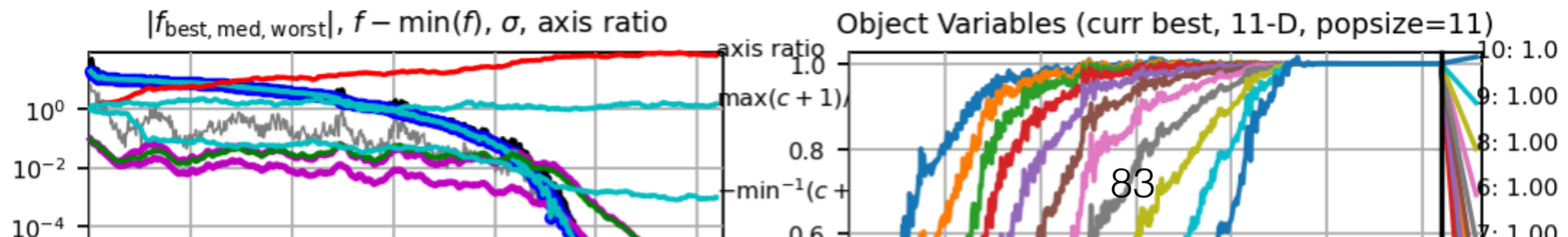
Optimizing the 11D Rosenbrock Function

```
1  import cma
2
3  x, es = cma.fmin2(cma.ff.rosen, 11 * [0.1], 0.1, {'verb_disp_overwrite': 200})
4  es.plot();  # cma.plot()  # works too
```

✓ 1.2s Python

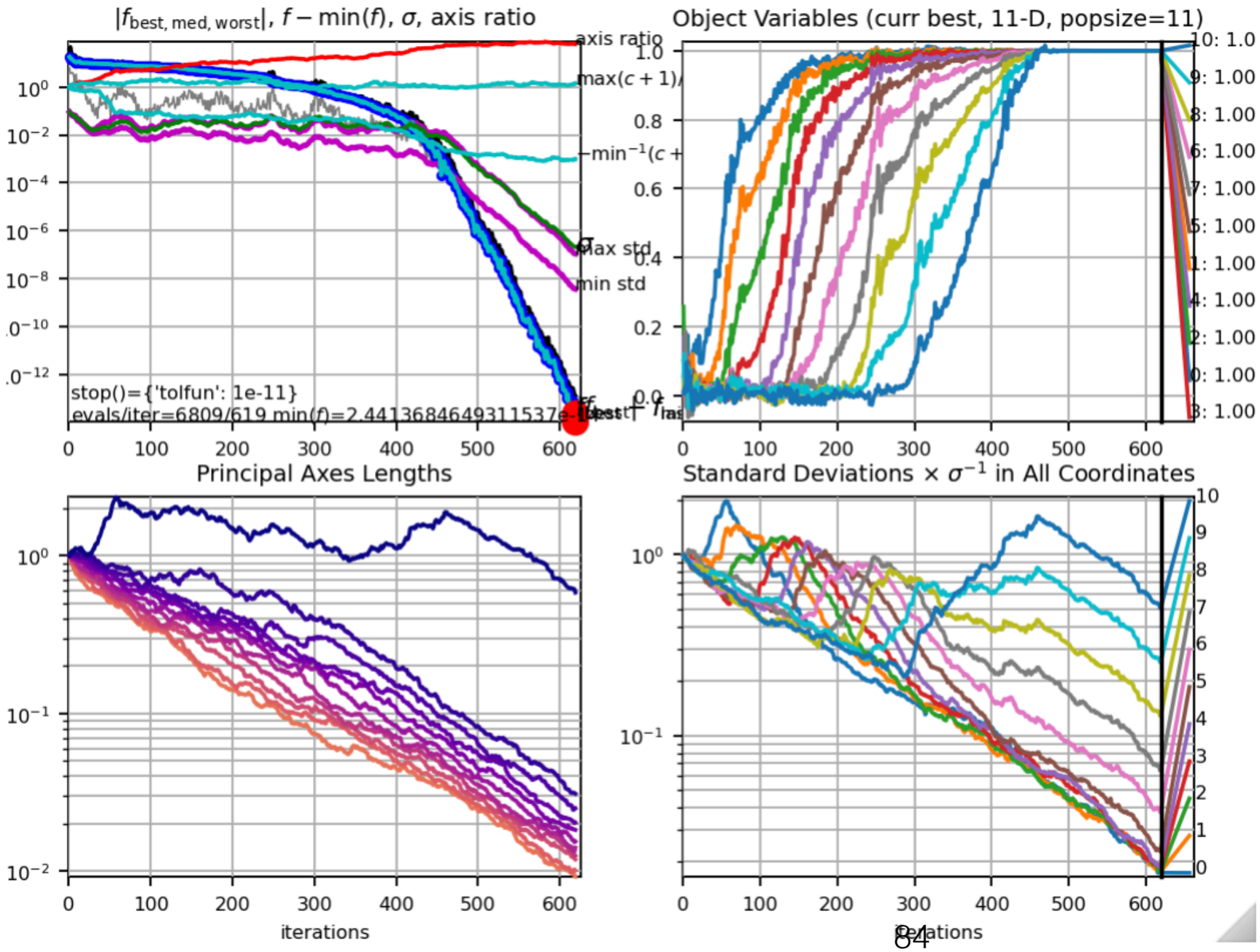
```
(5_w,11)-aCMA-ES (mu_w=3.4,w_1=42%) in dimension 11 (seed=329070, Fri Sep 13 13:55:03 2024)
Iterat #Fevals   function value  axis ratio  sigma  min&max std  t[m:s]
    1     11 1.807461200564364e+01 1.0e+00 9.01e-02  9e-02  9e-02 0:00.0
    2     22 1.609456060870233e+01 1.1e+00 8.18e-02  8e-02  8e-02 0:00.0
    3     33 1.501834332056772e+01 1.2e+00 7.39e-02  7e-02  7e-02 0:00.0
  100   1100 7.225343889168434e+00 5.6e+00 1.84e-02  8e-03  2e-02 0:00.1
  200   2200 3.475523910468565e+00 1.1e+01 2.94e-02  8e-03  3e-02 0:00.1
  619   6809 2.441368464931154e-14 6.4e+01 2.14e-07  4e-09  1e-07 0:00.3
termination on tolfun=1e-11 (Fri Sep 13 13:55:04 2024)
final/bestever f-value = 1.235402e-14 1.235402e-14 after 6810/6810 evaluations
incumbent solution: [1. 1. 1. 1. 1. 1. 1. 1. ...]
std deviations: [3.73610667e-09 3.79778919e-09 3.80249505e-09 3.97976791e-09
 4.15284285e-09 5.01249494e-09 8.22295518e-09 1.37738208e-08 ...]
```

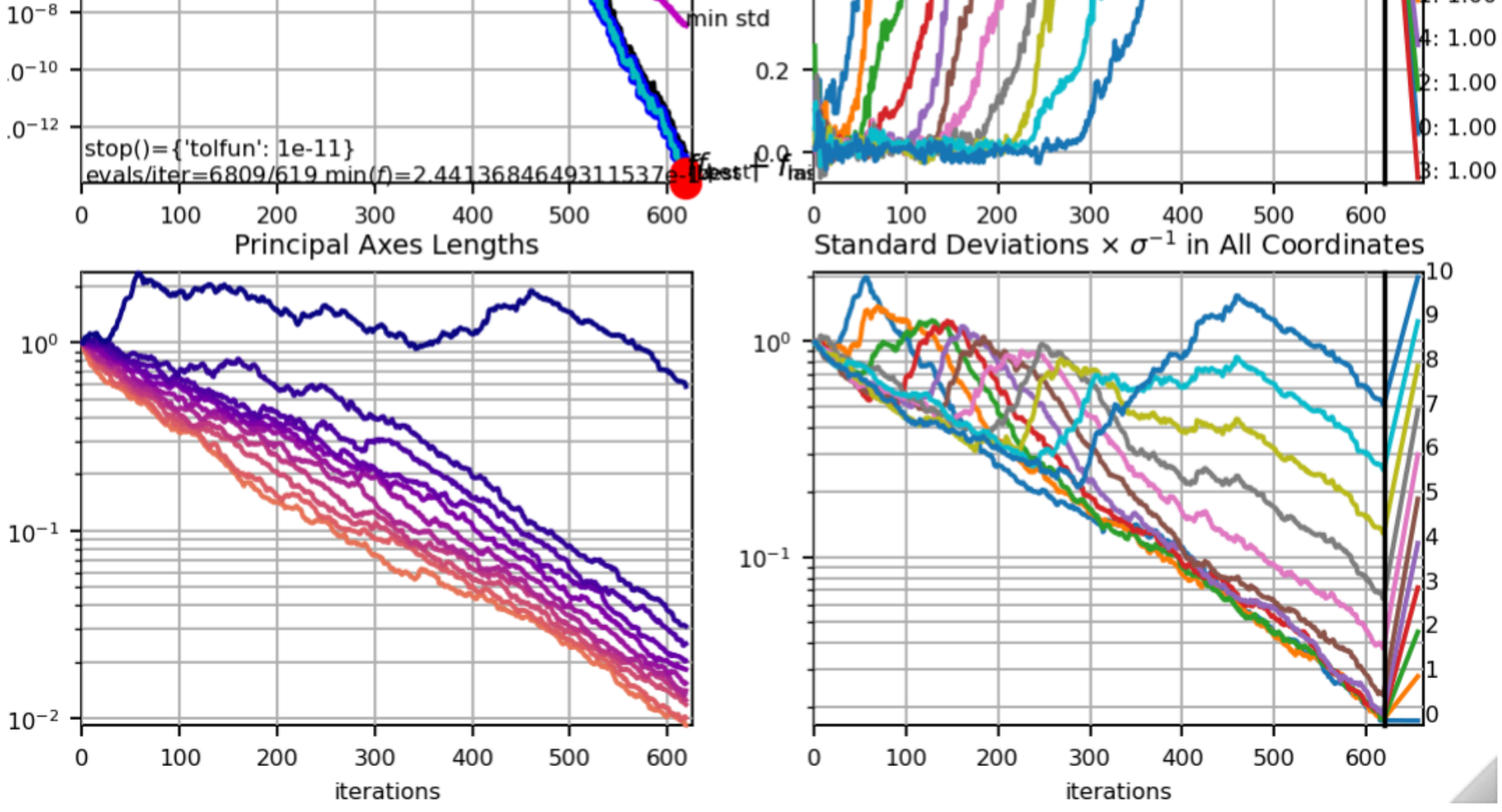
Figure 374



```
  100    1100 7.2253438891684 34e+00 5.6e+00 1.84e-02  8e-03  2e-02 0:00.1
  200    2200 3.4755239104685 65e+00 1.1e+01 2.94e-02  8e-03  3e-02 0:00.1
  619    6809 2.4413684649311 54e-14 6.4e+01 2.14e-07  4e-09  1e-07 0:00.3
termination on tolfun=1e-11 (Fri Sep 13 13:55:04 2024)
final/bestever f-value = 1.235402e-14 1.235402e-14 after 6810/6810 evaluations
incumbent solution: [1. 1. 1. 1. 1. 1. 1. 1. ...]
std deviations: [3.73610667e-09 3.79778919e-09 3.80249505e-09 3.97976791e-09
 4.15284285e-09 5.01249494e-09 8.22295518e-09 1.37738208e-08 ...]
```

Figure 374






```
1  cma.CMAOptions('tol')
```

✓ 0.0s Python

```
{'mindx': '0  #v minimal std in any arbitrary direction, cave interference with tol*',
 'minstd': '0  #v minimal std (scalar or vector) in any coordinate direction, cave interference with tol*',
 'tolconditioncov': '1e14  #v stop if the condition of the covariance matrix is above `tolconditioncov`',
 'tolfacupx': '1e3  #v termination when step-size increases by tolfacupx (diverges). That is, the initial step-size
 'tolflatfitness': '1  #v iterations tolerated with flat fitness before termination',
 'tolfun': '1e-11  #v termination criterion: tolerance in function value, quite useful',
 'tolfunhist': '1e-12  #v termination criterion: tolerance in function value history',
 'tolfunrel': '0  #v termination criterion: relative tolerance in function value: Delta f current < tolfunrel * (me
 'tolstagnation': 'int(100 + 100 * N**1.5 / popsize)  #v termination if no improvement over tolstagnation iteration
 'tolupsigma': '1e20  #v sigma/sigma0 > tolupsigma * max(eivenvals(C)**0.5) indicates "creeping behavior" with usua
 'tolx': '1e-11  #v termination criterion: tolerance in x-changes',
 'tolxstagnation': '[1e-9, 20, 0.1]  #v termination thresholds for Delta of [mean, iterations, iterations fraction]
```

Topics

0. Problem statement

1. What makes an optimization problem difficult to solve?

2. How does the CMA-ES work?
   - Normal Distribution, Rank-Based Recombination
   - Step-Size Adaptation
   - Covariance Matrix Adaptation

3. What can users do?

   - Choice of problem formulation and encoding (not covered)
   - Choice of initial solution and initial step-size
   - Restarts, Increasing Population Size
   - Restricted Covariance Matrix

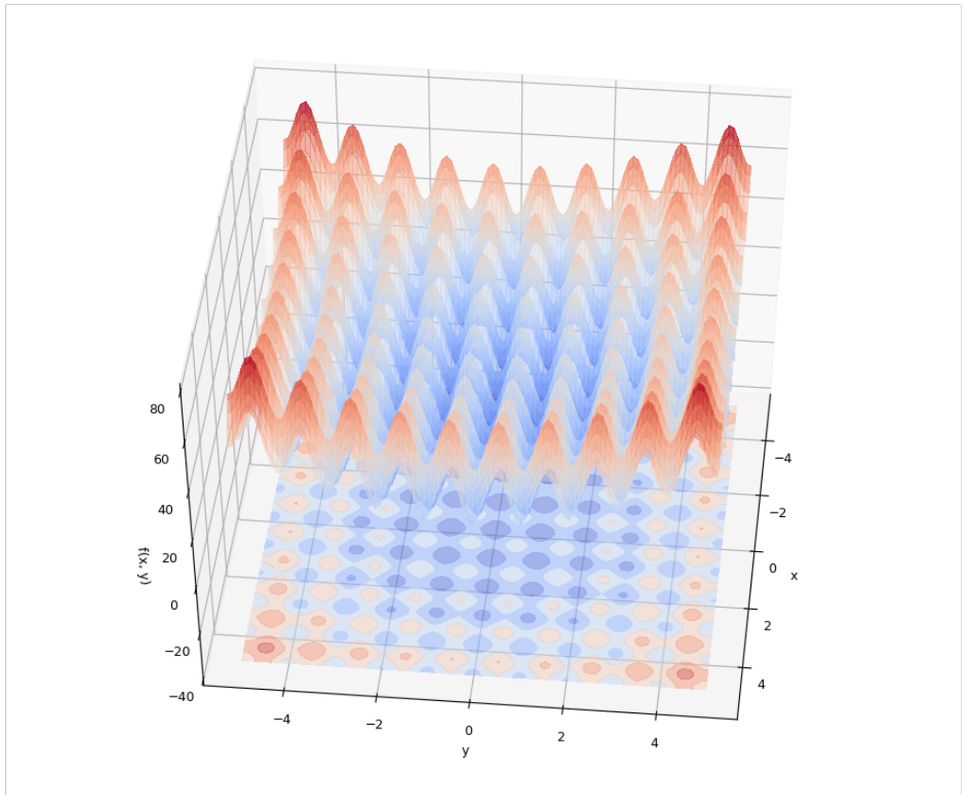
Multimodality

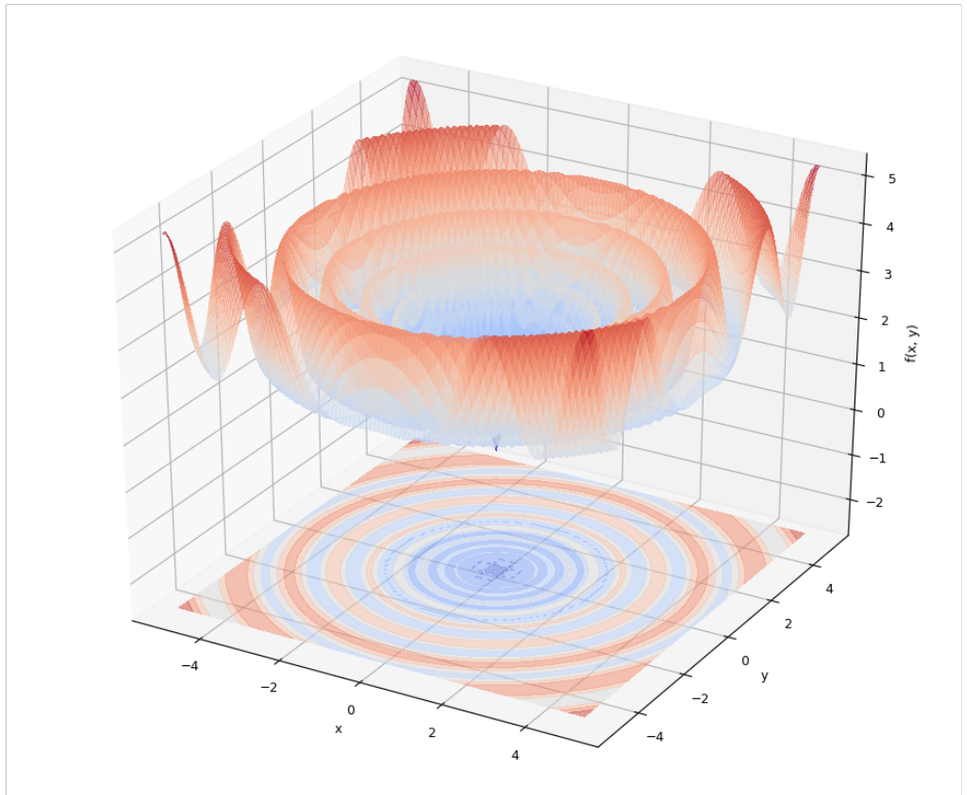
Approaches for multimodal functions: Try again with

- the final solution as initial solution (non-elitist) and small step-size
- a larger population size
- a different initial mean vector (and a smaller initial step-size)

A restart with a **large population size** helps if the objective function has a well global structure

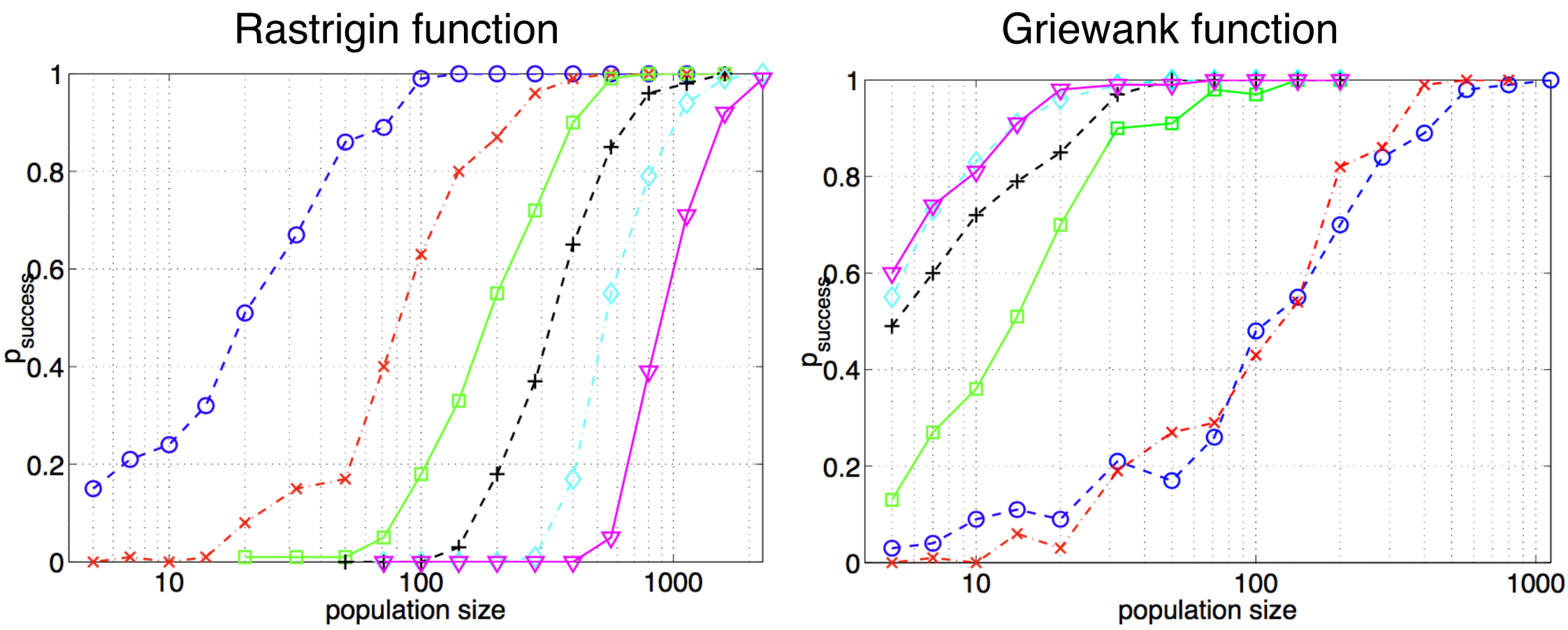
- functions such as Schaffer, Rastrigin, BBOB function 15~19
- loosely, unimodal global structure + deterministic noise





Multimodality

Hansen and Kern. Evaluating the CMA Evolution Strategy on Multimodal Test Functions, PPSN 2004.



(a) (b)

Fig. 1. Success rate to reach $f_{\text{stop}} = 10^{-10}$ versus population size for (a) Rastrigin function (b) Griewank function for dimensions $n = 2$ ('−−◯−−'), $n = 5$ ('−·−×−·−'), $n = 10$ ('—□—'), $n = 20$ ('− − + − −'), $n = 40$ ('− · −◇− ·−'), and $n = 80$ ('—▽—').

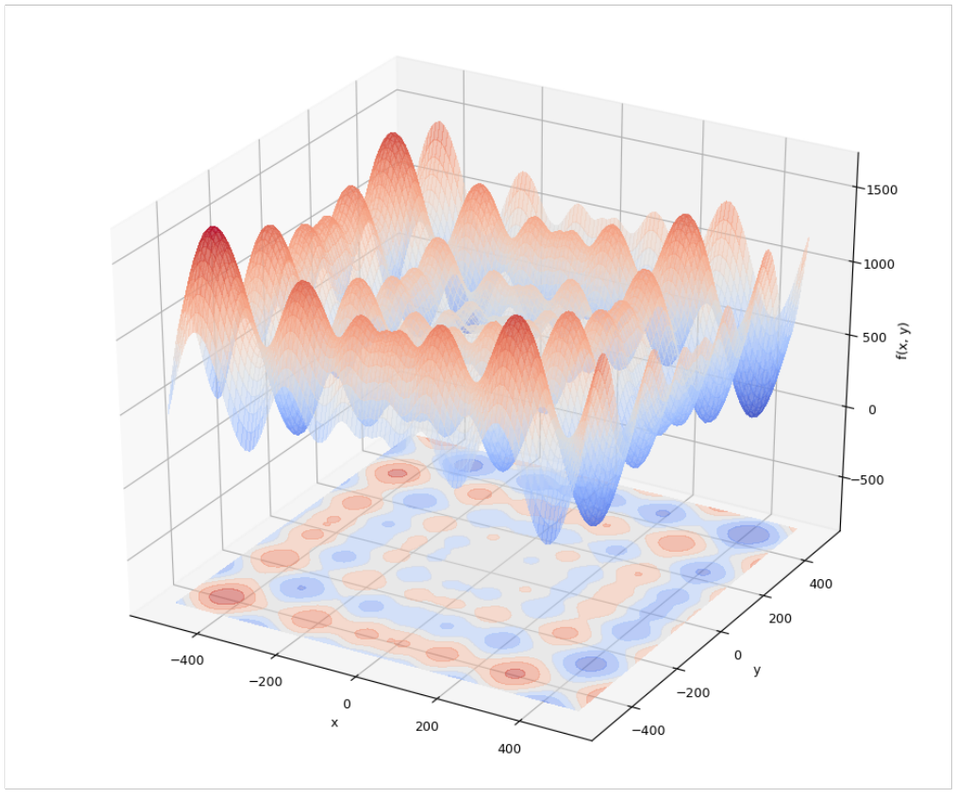
Multimodality

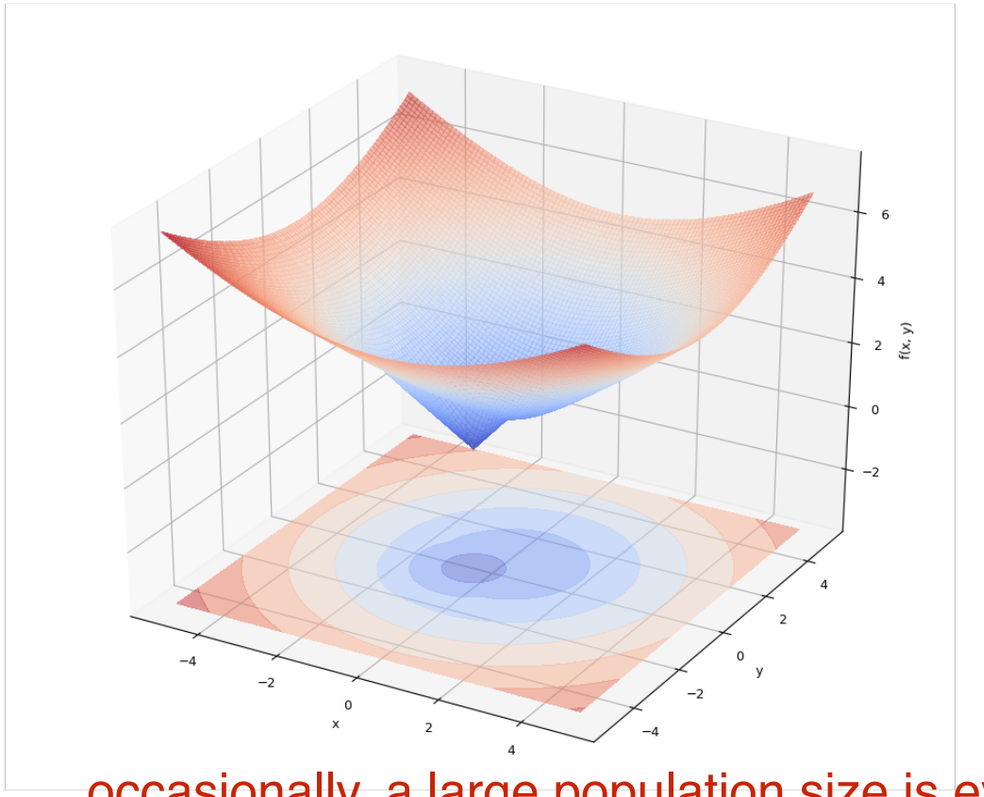
Approaches for multimodal functions: Try again with

- the final solution as initial solution (non-elitist) and small step-size
- a larger population size
- a different initial mean vector (and a smaller initial step-size)

A restart with a **small initial step-size** helps if the objective function has a weak global structure

- functions such as Schwefel, Bi-Sphere, BBOB function 20~24





occasionally, a large population size is even counterproductive

Restart Strategy

It makes the CMA-ES parameter free

IPOP: Restart with increasing the population size

- start with the default population size
- double the population size after each trial (parameter sweep)
- may be considered as gold standard for automated restarts

BIPOP: IPOP regime + Local search regime

- IPOP regime: restart with increasing population size
- Local search regime: restart with a smaller step-size and a smaller population size than the IPOP regime

Summary and Final Remarks

The Continuous Search Problem

Difficulties of a non-linear optimization problem are

- dimensionality and non-separabitity

  demands to exploit problem structure, e.g. neighborhood

  caveat: design of benchmark functions

- ill-conditioning

  demands to acquire a second order model

- ruggedness

  demands a non-local (stochastic? population based?) approach

Main Characteristics of (CMA) Evolution Strategies

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

Limitations

of CMA Evolution Strategies

- internal CPU-time: $10^{-8} n^2$ seconds per function evaluation on a 2GHz PC, tweaks are available

  1 000 000 f-evaluations in 100-D take 100 seconds *internal* CPU-time

  variants with restricted covariance matrix such as Sep-CMA

- better methods are presumably available in case of
  - partly separable problems
  - specific problems, for example with cheap gradients

    specific methods
  - small dimension ($n \ll 10$)

    for example Nelder-Mead
  - small running times (number of f-evaluations $< 100n$)

    model-based methods

Thank you

Source code for CMA-ES in C, C++, Java, Matlab, Octave, Python, R, Scilab and

Practical hints for problem formulation, variable encoding, parameter setting are available (or linked to) at

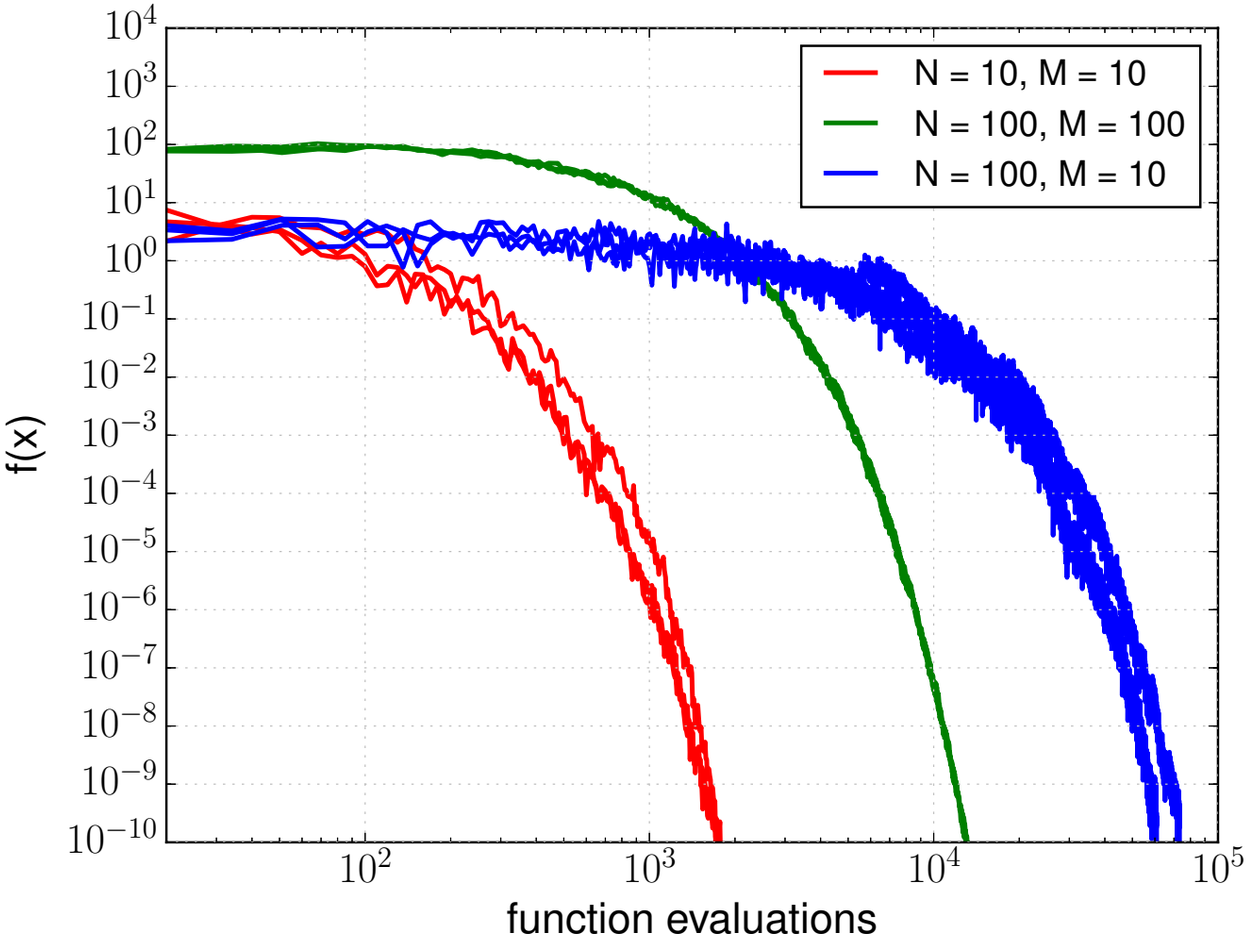
https://cma-es.github.io/

On Sphere with Low Effective Dimension

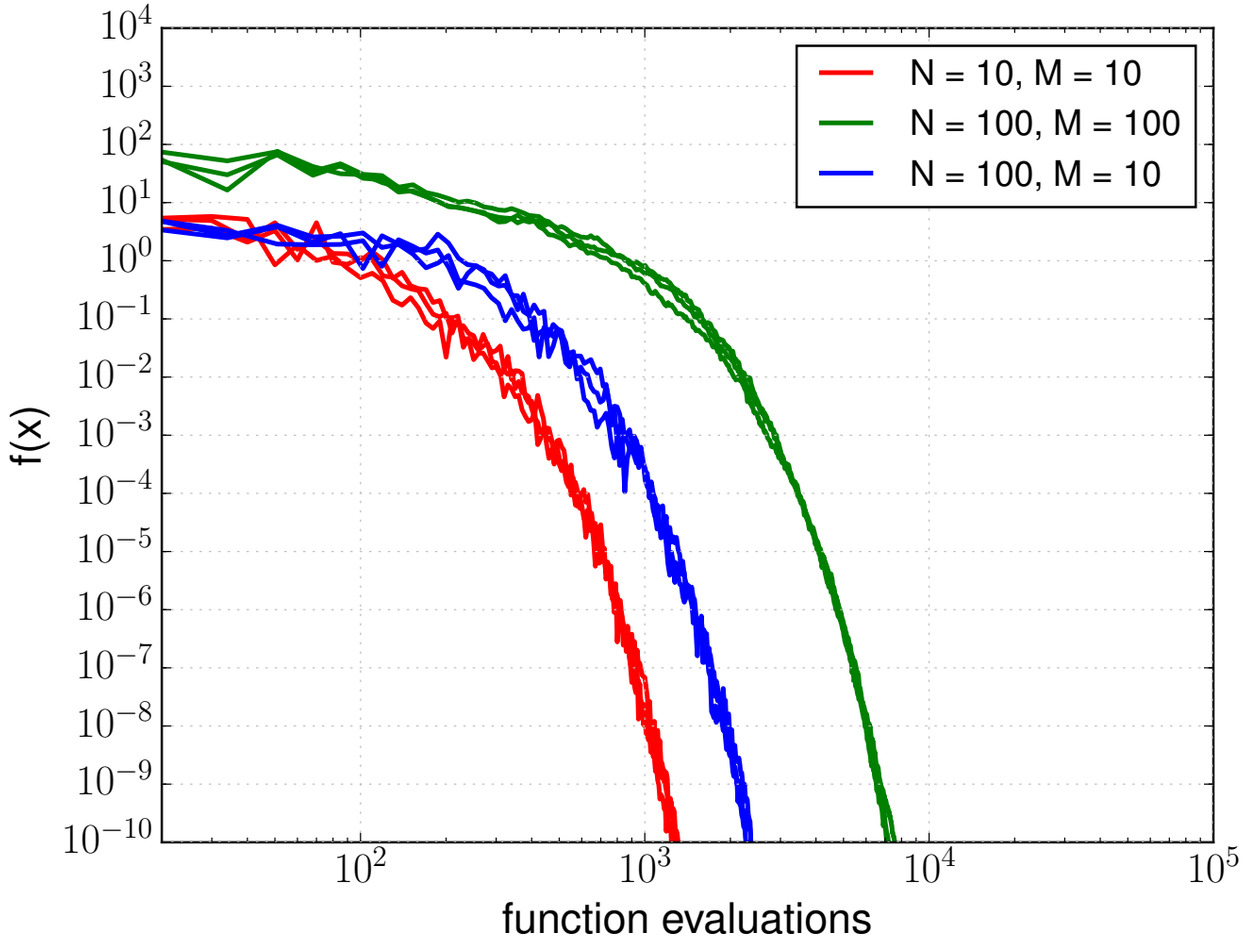
On a function with low effective dimension

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

CSA



TPA



Maximum Likelihood Update

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

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

independently of the given covariance matrix

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

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

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

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
  - $w_{i=1,\ldots,\mu}$, recombination weights
- related to $\mathbf{C}$-update
  - $c_{\mathbf{c}}$, decay rate for the evolution path
  - c_1, learning rate for rank-one update of $\mathbf{C}$
  - c_μ, learning rate for rank-μ update of $\mathbf{C}$
- related to σ-update
  - c_σ, decay 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 (and the initial σ) might be reasonably varied in a wide range, *depending on the objective function*
Useful: restarts with increasing population size (IPOP)