CMA-ES and Advanced Adaptation Mechanisms

Youhei Akimoto¹ & Nikolaus Hansen²
1. University of Tsukuba, Japan
2. Inria, Research Centre Saclay, France

akimoto@cs.tsukuba.ac.jp
nikolaus.hansen@inria.fr

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.

GECCO '18 Companion, July 15–19, 2018, Kyoto, Japan
© 2018 Copyright is held by the owner/author(s).
https://doi.org/10.1145/3205651.3207854
We are happy to answer questions at any time.
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
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
Problem Statement

Continuous Domain Search/Optimization

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

  \[ f : \mathcal{X} \subseteq \mathbb{R}^n \rightarrow \mathbb{R}, \quad x \mapsto f(x) \]

- **Black Box scenario** *(direct search scenario)*
  - gradients are not available or not useful
  - problem domain specific knowledge is used only within the black box, e.g. within an appropriate encoding

- **Search costs**: number of function evaluations
Problem Statement

Continuous Domain Search/Optimization

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

Typical Examples
- shape optimization (e.g. using CFD)
- model calibration
- parameter calibration

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

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

- \(f : x \mapsto f(x)\) separable
- \(f : x \mapsto f(Rx)\) non-separable

\[ R \text{ rotation matrix} \]

---


Ill-Conditioned Problems

Curvature of level sets

Consider the convex-quadratic function
\[ f(x) = \frac{1}{2} (x - x^*)^T H (x - x^*) = \frac{1}{2} \sum_i h_{i,i} (x_i - x_i^*)^2 + \frac{1}{2} \sum_{i \neq j} h_{i,j} (x_i - x_i^*)(x_j - x_j^*) \]

\( H \) is Hessian matrix of \( f \) and symmetric positive definite

Gradient direction \(-f'(x)^T\)

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

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 \( H \approx I \) (small condition number of \( H \)) first order information (e.g. the gradient) is sufficient. Otherwise second order information (estimation of \( H^{-1} \)) is necessary.
Non-smooth level sets (sharp ridges)

Similar difficulty but worse than ill-conditioning

1-norm  scaled 1-norm  1/2-norm
What Makes a Function Difficult to Solve?

...and what can be done

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

13
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 $f : \mathbb{R}^n \rightarrow \mathbb{R}$

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

While not terminate

1. Sample distribution $P(x|\theta) \rightarrow x_1, \ldots, x_\lambda \in \mathbb{R}^n$
2. Evaluate $x_1, \ldots, x_\lambda$ on $f$
3. Update parameters $\theta \leftarrow F_\theta(\theta, x_1, \ldots, x_\lambda, f(x_1), \ldots, f(x_\lambda))$

Everything depends on the definition of $P$ and $F_\theta$
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 $f : \mathbb{R}^n \rightarrow \mathbb{R}$

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

While not terminate

1. Sample distribution $P(x|\theta) \rightarrow x_1, \ldots, x_\lambda \in \mathbb{R}^n$
2. Evaluate $x_1, \ldots, x_\lambda$ on $f$
3. Update parameters $\theta \leftarrow F_\theta(\theta, x_1, \ldots, x_\lambda, f(x_1), \ldots, f(x_\lambda))$

Everything depends on the definition of $P$ and $F_\theta$

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 \( f : \mathbb{R}^n \rightarrow \mathbb{R} \)

Initialize distribution parameters \( \theta \), set population size \( \lambda \in \mathbb{N} \)

While not terminate

1. Sample distribution \( P(x|\theta) \rightarrow x_1, \ldots, x_\lambda \in \mathbb{R}^n \)
2. Evaluate \( x_1, \ldots, x_\lambda \) on \( f \)
3. Update parameters \( \theta \leftarrow F_\theta(\theta, x_1, \ldots, x_\lambda, f(x_1), \ldots, f(x_\lambda)) \)

Everything depends on the definition of \( P \) and \( F_\theta \)

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 $f : \mathbb{R}^n \to \mathbb{R}$

1. Initialize distribution parameters $\theta$, set population size $\lambda \in \mathbb{N}$
2. While not terminate
   1. Sample distribution $P(x|\theta) \to x_1, \ldots, x_{\lambda} \in \mathbb{R}^n$
   2. Evaluate $x_1, \ldots, x_{\lambda}$ on $f$
   3. Update parameters $\theta \leftarrow F_{\theta}(\theta, x_1, \ldots, x_{\lambda}, f(x_1), \ldots, f(x_{\lambda}))$

Everything depends on the definition of $P$ and $F_\theta$

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 $f : \mathbb{R}^n \rightarrow \mathbb{R}$

1. Initialize distribution parameters $\theta$, set population size $\lambda \in \mathbb{N}$
2. While not terminate
   1. Sample distribution $P(x|\theta) \rightarrow x_1, \ldots, x_\lambda \in \mathbb{R}^n$
   2. Evaluate $x_1, \ldots, x_\lambda$ on $f$
   3. Update parameters $\theta \leftarrow F_\theta(\theta, x_1, \ldots, x_\lambda, f(x_1), \ldots, f(x_\lambda))$

Everything depends on the definition of $P$ and $F_\theta$

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 \( f : \mathbb{R}^n \rightarrow \mathbb{R} \)

Initialize distribution parameters \( \theta \), set population size \( \lambda \in \mathbb{N} \)

While not terminate

1. Sample distribution \( P(x|\theta) \rightarrow x_1, \ldots, x_\lambda \in \mathbb{R}^n \)
2. Evaluate \( x_1, \ldots, x_\lambda \) on \( f \)
3. Update parameters \( \theta \leftarrow F_\theta(\theta, x_1, \ldots, x_\lambda, f(x_1), \ldots, f(x_\lambda)) \)

Everything depends on the definition of \( P \) and \( F_\theta \)
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 $f : \mathbb{R}^n \rightarrow \mathbb{R}$

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

1. Sample distribution $P(x|\theta) \rightarrow x_1, \ldots, x_\lambda \in \mathbb{R}^n$
2. Evaluate $x_1, \ldots, x_\lambda$ on $f$
3. Update parameters $\theta \leftarrow F_\theta(\theta, x_1, \ldots, x_\lambda, f(x_1), \ldots, f(x_\lambda))$

Everything depends on the definition of $P$ and $F_\theta$

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 $f : \mathbb{R}^n \rightarrow \mathbb{R}$

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

While not terminate

1. Sample distribution $P (x | \theta) \rightarrow x_1, \ldots, x_\lambda \in \mathbb{R}^n$
2. Evaluate $x_1, \ldots, x_\lambda$ on $f$
3. Update parameters $\theta \leftarrow F_\theta(\theta, x_1, \ldots, x_\lambda, f(x_1), \ldots, f(x_\lambda))$

Everything depends on the definition of $P$ and $F_\theta$
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: \( m \in \mathbb{R}^n; \sigma \in \mathbb{R}^+; \lambda \in \mathbb{N}_{\geq 2}, \) usually \( \lambda \geq 5, \) default \( 4 + \lfloor 3 \log n \rfloor \)

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

Initialize \( C = I, \) and \( p_c = 0, p_{\sigma} = 0 \)

While not terminate
\[
x_i = m + \sigma y_i, \quad \text{where } y_i \sim \mathcal{N}_i(0, C) \quad \text{for } i = 1, \ldots, \lambda \quad \text{sampling}
\]
\[
m \leftarrow m + c_m \sigma y_w, \quad \text{where } y_w = \sum_{i=1}^{\mu} w_{rk(i)} y_i \quad \text{update mean}
\]
\[
p_{\sigma} \leftarrow (1 - c_{\sigma}) p_{\sigma} + \sqrt{1 - (1 - c_{\sigma})^2} \sqrt{\mu_w} C^{-1/2} y_w \quad \text{path for } \sigma
\]
\[
p_c \leftarrow (1 - c_c) p_c + 1_{[0,2n]} \| p_{\sigma} \| \sqrt{1 - (1 - c_c)^2} \sqrt{\mu_w} y_w \quad \text{path for } C
\]
\[
\sigma \leftarrow \sigma \times \exp \left( \frac{c_{\sigma}}{d_{\sigma}} \left( \frac{\| p_{\sigma} \|}{E\| \mathcal{N}(0, I) \|} - 1 \right) \right) \quad \text{update of } \sigma
\]
\[
C \leftarrow C + c_{\mu} \sum_{i=1}^{\lambda} w_{rk(i)} (y_i y_i^T - C) + c_1 (p_c p_c^T - C) \quad \text{update } C
\]

Not covered: termination, restarts, useful output, search boundaries and encoding, corrections for: positive definiteness guaranty, \( p_c \) variance loss, \( c_{\sigma} \) and \( d_{\sigma} \) for large \( \lambda \)
Evolution Strategies

New search points are sampled normally distributed

\[ x_i \sim m + \sigma \mathcal{N}_i(0, C) \quad \text{for } i = 1, \ldots, \lambda \]

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

where

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

Here, all new points are sampled with the same parameters.

The question remains how to update \( m, C, \) and \( \sigma \).
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:
   \[ \mathcal{N}(x, A) + \mathcal{N}(y, B) \sim \mathcal{N}(x + y, A + B) \]
   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 \((n\text{-Dimensional})\) Normal Distribution

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

The mean value \(\mathbf{m}\)

- 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 \(\{\mathbf{x} \in \mathbb{R}^n \mid (\mathbf{x} - \mathbf{m})^T \mathbf{C}^{-1} (\mathbf{x} - \mathbf{m}) = n\}\)
Evolution Strategies (ES)  The Normal Distribution

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

Lines of Equal Density

\[
\begin{align*}
\mathcal{N}(m, \sigma^2 I) & \sim m + \sigma \mathcal{N}(0, I) \\
\text{one degree of freedom} & \quad \sigma \\
\text{components are} & \quad \text{independent standard} \\
\text{normally distributed} & \\
\mathcal{N}(m, D^2) & \sim m + D \mathcal{N}(0, I) \\
n \text{degrees of freedom} & \quad n \\
\text{components are} & \quad \text{independent, scaled} \\
\mathcal{N}(m, C) & \sim m + C^{1/2} \mathcal{N}(0, I) \\
(n^2 + n) / 2 \text{ degrees of freedom} & \quad (n^2 + n) / 2 \\
\text{components are} & \quad \text{correlated} \\
\end{align*}
\]

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

For any positive definite symmetric $C$,

$$C = d_1^2 b_1 b_1^T + \cdots + d_N^2 b_N b_N^T$$

$d_i$: square root of the eigenvalue of $C$

$b_i$: eigenvector of $C$, corresponding to $d_i$

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

$$\mathcal{N}(m, C) \sim m + \mathcal{N}(0, d_1^2) b_1 + \cdots + \mathcal{N}(0, d_N^2) b_N$$
The \((\mu/\mu, \lambda)\)-ES

Non-elitist selection and intermediate (weighted) recombination

Given the \(i\)-th solution point \(x_i = m + \sigma \mathcal{N}_i(0, C) = m + \sigma y_i \)

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

\[
m \leftarrow \sum_{i=1}^{\mu} w_i x_{i:}\lambda = m + \sigma \sum_{i=1}^{\mu} w_i y_{i:}\lambda =: y_w
\]

where

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

The best \(\mu\) 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

---

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

\[ f(x) \leftrightarrow f(x - a) \]

![Diagram showing invariance](image)

Identical behavior on \( f \) and \( f_a \)

\[
\begin{align*}
  f & : \quad x \mapsto f(x), & x^{(t=0)} &= x_0 \\
  f_a & : \quad x \mapsto f(x - a), & x^{(t=0)} &= x_0 + a
\end{align*}
\]

No difference can be observed w.r.t. the argument of \( f \)
On 20D Sphere Function: \( f(x) = \sum_{i=1}^{N} [x_i]^2 \)

- ES without adaptation can’t approach the optimum \( \Rightarrow \) adaptation required
Evolution Strategies

Recalling

New search points are sampled normally distributed

\[ x_i \sim m + \sigma N_i(0, C) \quad \text{for } i = 1, \ldots, \lambda \]

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

where

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

The remaining question is how to update \( \sigma \) and \( C \).
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

- **$\sigma$-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

---


$^b$Schumer and Steiglitz 1968. Adaptive step size random search. *IEEE TAC*


$^e$Ostermeier et al 1994, Step-size adaptation based on non-local use of selection information, *PPSN IV*
Path Length Control (CSA)
The Concept of Cumulative Step-Size Adaptation

Measure the length of the evolution path
the pathway of the mean vector $m$ in the generation sequence

\[ x_i = m + \sigma y_i \]
\[ m \leftarrow m + \sigma y_w \]

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

\[ m \leftarrow m + \sigma y_w \quad \text{where} \quad y_w = \sum_{i=1}^{\mu} w_i y_i : \lambda \quad \text{update mean} \]

\[ p_\sigma \leftarrow (1 - c_\sigma) p_\sigma + \sqrt{1 - (1 - c_\sigma)^2} \frac{\sqrt{\mu_w}}{\sum_{i=1}^{\mu} w_i} y_w \quad \text{accounts for } 1 - c_\sigma \text{ accounts for } w_i \]

\[ \sigma \leftarrow \sigma \times \exp \left( \frac{c_\sigma}{d_\sigma} \left( \frac{\|p_\sigma\|}{\mathbb{E}\|\mathcal{N}(0, I)\|} - 1 \right) \right) \quad \text{update step-size} \]

\[ > 1 \iff \|p_\sigma\| \text{ is greater than its expectation} \]
(5/5, 10)-CSA-ES, default parameters

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

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

$$x_{1/2} = m^{(g)} \pm \sigma^{(g)} \frac{\|N(0, I)\|}{\|m^{(g)} - m^{(g-1)}\|} (m^{(g)} - m^{(g-1)})$$

$\|x\|_C := x^TC^{-1}x$

Compare the ranking of $x_1$ and $x_2$ among $\lambda$ current populations

$$s^{(g+1)} = (1 - c_s)s^{(g)} + c_s \frac{\text{rank}(x_2) - \text{rank}(x_1)}{\lambda - 1}$$

$>0$ if the previous step still produces a promising solution

Update the step-size

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


On Sphere with Low Effective Dimension

On a function with low effective dimension

\[ f(x) = \sum_{i=1}^{M} [x]_i^2, \quad x \in \mathbb{R}^N, \quad M \leq N. \]

- \( N - M \) variables do not affect the function value

![Graphs of CSA and TPA](image)
Alternatives: Success-Based Step-Size Control

comparing the fitness distributions of current and previous iterations

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

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

controls a *success probability*

An advantage over CSA and TPA: Cheap Computation

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


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, Population Success Rule

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

---

*Hansen et al. How to Assess Step-Size Adaptation Mechanisms in Randomised Search. PPSN 2014*
Step-Size Control: Summary

On 20D TwoAxes Function: $f(x) = \sum_{i=1}^{N/2} [Rx]_i^2 + a^2 \sum_{i=N/2+1}^{N} [Rx]_i^2$, $R$: orthogonal

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

Recalling

New search points are sampled normally distributed

\[ x_i \sim m + \sigma N_i(0, C) \quad \text{for } i = 1, \ldots, \lambda \]

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

where

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

The remaining question is how to update \( C \).
Covariance Matrix Adaptation

Rank-One Update

\[ m \leftarrow m + \sigma y_w, \quad y_w = \sum_{i=1}^{\mu} w_i \lambda_i, \quad y_i \sim \mathcal{N}_i(0, C) \]

Initial distribution, \( C = I \)
Covariance Matrix Adaptation

Rank-One Update

\[ m \leftarrow m + \sigma y_w, \quad y_w = \sum_{i=1}^{\mu} w_i y_i, \quad y_i \sim \mathcal{N}_i(0, C) \]

initial distribution, \( C = I \)
Covariance Matrix Adaptation

Rank-One Update

\[ m \leftarrow m + \sigma y_w, \quad y_w = \sum_{i=1}^{\mu} w_i y_i, \quad y_i \sim \mathcal{N}_i(0, C) \]

\( y_w \), movement of the population mean \( m \) (disregarding \( \sigma \))
Covariance Matrix Adaptation

Rank-One Update

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

mixture of distribution \( C \) and step \( y_w \),

\[ C \leftarrow 0.8 \times C + 0.2 \times y_w y_w^T \]
Covariance Matrix Adaptation

Rank-One Update

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

new distribution (disregarding \(\sigma\))
Covariance Matrix Adaptation

Rank-One Update

\[ m \leftarrow m + \sigma y_w, \quad y_w = \sum_{i=1}^{\mu} w_i y_i, \quad y_i \sim \mathcal{N}_i(0, C) \]

new distribution (disregarding \( \sigma \))
Covariance Matrix Adaptation

Rank-One Update

\[ m \leftarrow m + \sigma y_w, \quad y_w = \sum_{i=1}^{\mu} w_i y_i, \quad y_i \sim \mathcal{N}_i(0, C) \]

movement of the population mean \( m \)
Covariance Matrix Adaptation

Rank-One Update

\[ m \leftarrow m + \sigma y_w, \quad y_w = \sum_{i=1}^{\mu} w_i y_i, \quad y_i \sim \mathcal{N}_i(0, C) \]

mixture of distribution C and step \( y_w \),

\[ C \leftarrow 0.8 \times C + 0.2 \times y_w y_w^T \]
Covariance Matrix Adaptation

Rank-One Update

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

new distribution,
\[ \mathbf{C} \leftarrow 0.8 \times \mathbf{C} + 0.2 \times y_w y_w^T \]

the ruling principle: the adaptation increases the likelihood of successful steps, \( y_w \), to appear again

another viewpoint: the adaptation follows a natural gradient approximation of the expected fitness

... equations
Covariance Matrix Adaptation

Rank-One Update

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

While not terminate

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

$$m \leftarrow m + \sigma y_w \quad \text{where } y_w = \sum_{i=1}^{\mu} w_i y_i : \lambda$$

$$C \leftarrow (1 - c_{cov}) C + c_{cov} \mu_w y_w y_w^T \quad \text{where } \mu_w = \frac{1}{\sum_{i=1}^{\mu} w_i^2} \geq 1$$

The rank-one update has been found independently in several domains\textsuperscript{6 7 8 9}

\textsuperscript{6} Kjellström&Taxon 1981. Stochastic Optimization in System Design, IEEE TCS
\textsuperscript{7} Hansen&Ostermeier 1996. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation, ICEC
\textsuperscript{8} Ljung 1999. System Identification: Theory for the User
\textsuperscript{9} Haario et al 2001. An adaptive Metropolis algorithm, JSTOR
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 \( y_w \), sequentially in time and space
  - eigenvectors of the covariance matrix \( C \) are the principle components / the principle axes of the mutation ellipsoid
- learns a new **rotated problem representation**
  - components are independent (only) in the new representation
- learns a new **(Mahalanobis) metric**
- 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

\[
C \leftarrow (1 - c_{\text{cov}})C + c_{\text{cov}} \mu_w y_w y_w^T
\]
Invariance Under Rigid Search Space Transformation

for example, invariance under search space rotation

(separable ⇔ non-separable)
Invariance Under Rigid Search Space Transformation

\[ f = h_{\text{Rast}} \circ R \]

\[ f \text{-level sets in dimension 2} \]

\[ f = h \circ R \]

for example, invariance under search space rotation

(separable \( \Leftrightarrow \) non-separable)
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 $m$.

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

$$p_c \propto \sum_{i=0}^{g} (1 - c_c)^{g-i} \mathbf{y}_w^{(i)}$$

exponentially fading weights

The recursive construction of the evolution path (cumulation):

$$p_c \leftarrow \frac{(1 - c_c) p_c + \sqrt{1 - (1 - c_c)^2} \sqrt{\mu_w} \mathbf{y}_w}{\text{decay factor}}$$

$$\text{normalization factor}$$

Input $= \frac{m - m_{\text{old}}}{\sigma}$

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

Evolution Path

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

An exponentially weighted sum of steps $y_w$ is used

$$p_c \propto \sum_{i=0}^{g} (1 - c_c)^{g-i} y_w^{(i)}$$

The recursive construction of the evolution path (cumulation):

$$p_c \leftarrow (1 - c_c) \cdot p_c + \sqrt{1 - (1 - c_c)^2} \sqrt{\mu_w} \cdot y_w$$

where $\mu_w = \frac{1}{\sum w_i^2}$, $c_c \ll 1$. History information is accumulated in the evolution path.
“Cumulation” is a widely used technique and also known 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. . .
Cumulation

Utilizing the Evolution Path

We used $y_w y_w^T$ for updating $C$. Because $y_w y_w^T = -y_w (-y_w)^T$ the sign of $y_w$ is lost.

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

\[
C \leftarrow (1 - c_{\text{cov}})C + c_{\text{cov}} \mu_w y_w y_w^T
\]

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

\[
p_c \leftarrow (1 - c_c) p_c + \sqrt{1 - (1 - c_c)^2} \sqrt{\mu_w} y_w
\]

\[
p_c \leftarrow (1 - c_{\text{cov}}) p_c + c_{\text{cov}} p_c p_c^T
\]

\text{decay factor} \quad \text{normalization factor} \quad \text{rank-one}
Cumulation—Utilizing the Evolution Path

We used $y_w y_w^T$ for updating $C$. Because $y_w y_w^T = -y_w (-y_w)^T$ the sign of $y_w$ is lost.

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

\[
C \leftarrow (1 - c_{\text{cov}})C + c_{\text{cov}} \mu_w y_w y_w^T
\]

\[
p_c \leftarrow \left(1 - c_c\right) p_c + \sqrt{1 - (1 - c_c)^2} \sqrt{\mu_w} y_w
\]

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

\[
p_c \leftarrow \left(1 - c_c\right) p_c + c_{\text{cov}} p_c p_c^T
\]
Cumulation

Utilizing the Evolution Path

We used $y_w y_w^T$ for updating $C$. Because $y_w y_w^T = -y_w (-y_w)^T$ the sign of $y_w$ is lost.

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

$$P_c \leftarrow (1 - c_c) P_c + \frac{\sqrt{1 - (1 - c_c)^2}}{\mu_w} y_w$$

$$C \leftarrow (1 - c_{cov}) C + c_{cov} \frac{p_c p_c^T}{\mu_w}$$

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

$$C \leftarrow (1 - c_{cov}) C + c_{cov} \mu_w y_w y_w^T$$
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 $O(n^2)$ to $O(n)$.\(^{(a)}\)

\(^{(a)}\)Hansen & Auger 2013. Principled design of continuous stochastic search: From theory to practice.

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

The overall model complexity is $n^2$ but important parts of the model can be learned in time of order $n$
**Rank-$\mu$ Update**

\[
\begin{align*}
    x_i &= m + \sigma y_i, \\
    y_i &\sim \mathcal{N}_i(0, C), \\
    m &\leftarrow m + \sigma y_w \\
    y_w &= \sum_{i=1}^{\mu} w_i y_i: \lambda
\end{align*}
\]

The rank-$\mu$ update extends the update rule for large population sizes $\lambda$ using $\mu > 1$ vectors to update $C$ at each generation step.

The weighted empirical covariance matrix

\[
C_\mu = \sum_{i=1}^{\mu} w_i y_i: \lambda y_i^T: \lambda
\]

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

with $\mu = \lambda$ weights can be negative

The rank-$\mu$ update then reads

\[
C \leftarrow (1 - c_{\text{cov}}) C + c_{\text{cov}} C_\mu
\]

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

---

Jastrebski and Arnold (2006). Improving evolution strategies through active covariance matrix adaptation. CEC.
\[ x_i = m + \sigma y_i, \quad y_i \sim \mathcal{N}(0, C) \]

\[ C_\mu = \frac{1}{\mu} \sum_{i=1}^{\mu} y_i y_i^T \]

\[ C \leftarrow (1 - 1) \times C + 1 \times C_\mu \]

\[ m_{\text{new}} \leftarrow m + \frac{1}{\mu} \sum_{i=1}^{\mu} y_i : \lambda \]

Sampling of \( \lambda = 150 \) solutions where \( C = I \) and \( \sigma = 1 \)

Calculating \( C \) where \( \mu = 50 \), \( w_1 = \cdots = w_\mu = \frac{1}{\mu} \), and \( c_{\text{cov}} = 1 \)
Rank-$\mu$ CMA versus Estimation of Multivariate Normal Algorithm EMNA_{global}\(^{11}\)

\[ x_i = m_{old} + y_i, \quad y_i \sim \mathcal{N}(0, C) \]

\[ C \leftarrow \frac{1}{\mu} \sum (x_i: \lambda - m_{old})(x_i: \lambda - m_{old})^T \]

\[ m_{new} = m_{old} + \frac{1}{\mu} \sum y_i: \lambda \]

rank-$\mu$ CMA conducts a PCA of steps

EMNA_{global} conducts a PCA of points

sampling of $\lambda = 150$ solutions (dots) calculating $C$ from $\mu = 50$ solutions

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

---

The rank-$\mu$ update

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

Therefore the rank-$\mu$ update is the primary mechanism whenever a large population size is used 

say $\lambda \geq 3n + 10$

The rank-one update

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

Rank-one update and rank-$\mu$ update can be combined

The **rank-μ update**

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

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

The **rank-one update**

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

Rank-one update and rank-μ update can be combined …all equations

The **rank-\(\mu\) update**

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

Therefore the rank-\(\mu\) update is the primary mechanism whenever a large population size is used

\[\text{say } \lambda \geq 3n + 10\]

The **rank-one update**

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

Rank-one update and rank-\(\mu\) update can be combined

---

The rank-µ update increases the possible learning rate in large populations roughly from $\frac{2}{n^2}$ to $\frac{\mu w}{n^2}$. This can reduce the number of necessary generations roughly from $O(n^2)$ to $O(n)$. 

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

The rank-one update uses the evolution path and reduces the number of necessary function evaluations to learn straight ridges from $O(n^2)$ to $O(n)$.

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

...all equations

Anne Auger & Nikolaus Hansen  CMA-ES July, 2014 56 / 81
The rank-\(\mu\) update increases the possible learning rate in large populations roughly from \(\frac{2}{n}\) to \(\mu w / n\). This can reduce the number of necessary generations roughly from \(O(n^2)\) to \(O(n)\).

Therefore the rank-\(\mu\) update is the primary mechanism whenever a large population size is used, say \(3n + 10\).

The rank-one update uses the evolution path and reduces the number of necessary function evaluations to learn straight ridges from \(O(n^2)\) to \(O(n)\).

Rank-one update and rank-\(\mu\) update can be combined...

---

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

\(\lambda = 50\)
Different Types of Ill-Conditioning

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

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

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

\[ f(x) = \alpha \cdot x_1^2 + \sum_{i=1}^{n} x_i^2 \]
Active Update
utilize negative weights [Jastrebski and Arnold, 2006]

Active Update (rewriting)

\[ C \leftarrow C + c_1 p_c p_c^T + c_\mu \sum_{i=1}^{\lfloor \lambda/2 \rfloor} w_i y_{i: \lambda} y_{i: \lambda}^T - c_\mu \sum_{i=\lambda - \lfloor \lambda/2 \rfloor + 1}^{\lambda} |w_i| y_{i: \lambda} y_{i: \lambda}^T \]

- increasing the variances in promising directions
- decreasing the variances in unpromising directions

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

On 10D Discus Function

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

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

- Positive: wait for the smallest $\text{eig}(C)$ decreasing
- Active: decrease the smallest $\text{eig}(C)$ actively
Summary

Active Covariance Matrix Adaptation + Cumulation

\[
C \leftarrow (1 - c_1 - c_\mu + c_\mu^-) C + c_1 p_c p_c^T + c_\mu \sum_{i=1}^{\lfloor \lambda/2 \rfloor} w_i y_{i: \lambda} y_{i: \lambda}^T - c_\mu^- \sum_{i=\lambda - \lfloor \lambda/2 \rfloor + 1}^{\lambda} |w_i| y_{i: \lambda} y_{i: \lambda}^T
\]

- \(|w_i| < 0\) (for \(i \geq \lambda - \lfloor \lambda/2 \rfloor + 1\)): negative weight assigned to \(y_{i: \lambda}\), \(\sum_{i=\lambda - \mu}^{\lambda} |w_i| = 1\).
- \(c_\mu^- > 0\): learning rate for the active update

These components complement each other
- cumulation: excels to learn a long axis, but inefficient for a large \(\lambda\)
- rank-\(\mu\) update: efficient for a large \(\lambda\)
- active update: effective to learn short axes

An important yet solvable issue of active update
- The positive definiteness of \(C\) will be violated if \(c_\mu^-\) is not small enough
- The positive definiteness can be guaranteed w.p.1 by controlling \(c_\mu^- w_i\)
Input: $m \in \mathbb{R}^n; \sigma \in \mathbb{R}^+; \lambda \in \mathbb{N}_{\geq 2}$, usually $\lambda \geq 5$, default $4 + \lfloor 3 \log n \rfloor$

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

Initialize $C = I$, and $p_c = 0, p_\sigma = 0$

While not terminate

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

$$m \leftarrow m + c_m \sigma y_w, \quad \text{where } y_w = \sum_i w_{rk(i)} y_i \quad \text{update mean}$$

$$p_\sigma \leftarrow (1 - c_\sigma) p_\sigma + \sqrt{1 - (1 - c_\sigma)^2} \sqrt{\mu_w} C^{-\frac{1}{2}} y_w \quad \text{path for } \sigma$$

$$p_c \leftarrow (1 - c_c) p_c + 1_{[0,2n]} \{ \|p_\sigma\|^2 \} \sqrt{1 - (1 - c_c)^2} \sqrt{\mu_w} y_w \quad \text{path for } C$$

$$\sigma \leftarrow \sigma \times \exp \left( \frac{c_\sigma}{d_\sigma} \left( \frac{\|p_\sigma\|}{\mathbb{E}\|\mathcal{N}(0, I)\|} - 1 \right) \right) \quad \text{update of } \sigma$$

$$C \leftarrow C + c_\mu \sum_{i=1}^\lambda w_{rk(i)} (y_i y_i^T - C) + c_1 (p_c p_c^T - C) \quad \text{update } C$$

*Not covered*: termination, restarts, useful output, search boundaries and encoding, corrections for: positive definiteness guaranty, $p_c$ variance loss, $c_\sigma$ and $d_\sigma$ for large $\lambda$
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
Default Parameter Values
CMA-ES + (B)IPOP Restart Strategy = Quasi-Parameter Free Optimizer

The following parameters were identified in carefully chosen experimental set ups.

- related to selection and recombination
  - $\lambda$: offspring number, new solutions sampled, population size
  - $\mu$: parent number, solutions involved in mean update
  - $w_i$: recombination weights

- related to $C$-update
  - $1 - c_c$: decay rate for the evolution path, cumulation factor
  - $c_1$: learning rate for rank-one update of $C$
  - $c_\mu$: learning rate for rank-$\mu$ update of $C$

- related to $\sigma$-update
  - $1 - c_\sigma$: decay rate of the evolution path
  - $d_\sigma$: damping for $\sigma$-change

The default values depends only on the dimension. They do in the first place not depend on the objective function.
Parameters to be set depending on the problem

Initialization and termination conditions

The following should be set or implemented depending on the problem.

- related to the initial search distribution
  - $m^{(0)}$: initial mean vector
  - $\sigma^{(0)}$ (or $\sqrt{C_{i,i}^{(0)}}$): initial (coordinate-wise) standard deviation

- related to stopping conditions
  - max. func. evals.
  - max. iterations
  - function value tolerance
  - min. axis length
  - stagnation

Practical Hints:

- start with an initial guess $m^{(0)}$ with a relatively small step-size $\sigma^{(0)}$ to *locally* improve the current guess;

- then increase the step-size, e.g., by factor of 10, to *globally* search for a better solution.
Python CMA-ES Implementation

[https://github.com/CMA-ES/pycma](https://github.com/CMA-ES/pycma)

pycma

A Python implementation of CMA-ES and a few related numerical optimization tools.

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is a stochastic derivative-free numerical optimization algorithm for difficult (non-convex, ill-conditioned, multi-modal, rugged, noisy) optimization problems in continuous search spaces.

Useful links:

- A quick start guide with a few usage examples
- The API Documentation
- Hints for how to use this (kind of) optimization module in practice

**Installation of the latest release**

Type

```
   python -m pip install cma
```

in a system shell to install the latest release from the Python Package Index (PyPI). The release link also provides more installation hints and a quick start guide.
What can/should the users do?

**Strategy Parameters and Initialization**

**Python CMA-ES Demo**

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

**Optimizing 10D Rosenbrock Function**

```
In [1]:
    import cma
    # import
    opts = cma.CMAOptions()  # CMA Options
    opts['ftarget'] = 1e-4  # - function value target
    opts['maxfevals'] = 1e6  # - max. function evaluations
    cma.fmin(cma.ff.rosen,  # Minimize Rosenbrock function
             x0=[0.0] * 10,  # - x0 = [0,..., 0]
             sigma0=0.1,  # - sigma0 = 0.1
             options=opts)  # - other options
```

(5_w,10)-aCMA-ES (mu_w=3.2, w_1=45%) in dimension 10 (seed=909490, Mon Apr 16 13:39:57 2018)

| Iterat | #Fevals | function value | axis ratio | sigma | min|max | std | t|m|s |
|--------|---------|----------------|------------|-------|----|----|-----|---|---|
| 1      | 10      | 1.169928472214858e+01 | 1.0e+00 | 9.12e-02 | 9e-02 | 9e-02 | 0:00.0 |
| 2      | 20      | 1.363303277917634e+01 | 1.1e+00 | 8.33e-02 | 8e-02 | 8e-02 | 0:00.0 |
| 3      | 30      | 1.232089008099892e+01 | 1.2e+00 | 7.55e-02 | 7e-02 | 8e-02 | 0:00.0 |
| 100    | 1000    | 5.724977739870999e+00 | 9.1e+00 | 1.65e-02 | 7e-03 | 2e-02 | 0:00.1 |
| 200    | 2000    | 2.550841127554589e+00 | 1.5e+01 | 3.97e-02 | 1e-02 | 4e-02 | 0:00.2 |
| 300    | 3000    | 3.674986141687587e-01 | 1.5e+01 | 2.76e-02 | 3e-03 | 2e-02 | 0:00.4 |
| 400    | 4000    | 1.26634564781239e-03 | 5.0e+01 | 1.18e-02 | 8e-04 | 2e-02 | 0:00.5 |
| 420    | 4200    | 7.039461687999381e-05 | 5.5e+01 | 4.04e-03 | 2e-04 | 5e-03 | 0:00.5 |

 termination on ftarget=0.0001 (Mon Apr 16 13:39:58 2018)

final/bestever f-value = 2.804423e-05 2.804423e-05

incumbent solution: [ 0.9998542 0.99996219 0.9999681 1.00000445 0.9998977 0.99968537
                      0.99954974 0.99918266 ...]

std deviations: [ 0.00023937 0.00022203 0.00024836 0.00024782 0.0003
                 1258 0.00043481
                 0.00078261 0.0014964 ...]

82
Python CMA-ES Demo
https://github.com/CMA-ES/pycma

Optimizing 10D Rosenbrock Function
Multimodality

Two approaches for multimodal functions: Try again with

- a larger population size
- a smaller initial step-size (and random initial mean vector)
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
What can/should the users do? Multimodality

Multimodality

![Graphs showing success rate vs population size for Rastrigin and Griewank functions](image)

**Fig. 1.** Success rate to reach $f_{\text{stop}} = 10^{-10}$ versus population size for (a) Rastrigin function (b) Griewank function for dimensions $n = 2$ (‘−−〇−−’), $n = 5$ (‘⋯×⋯’), $n = 10$ (‘□’), $n = 20$ (‘−−−−’), $n = 40$ (‘−⋅−◇−⋅’), and $n = 80$ (‘−▽’).
What can/should the users do?

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

A large population size has a negative effect
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
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
Motivation of the Restricted Covariance Matrix

Bottlenecks of the CMA-ES on high dimensional problems

1. \( \mathcal{O}(N^2) \) Time and Space Complexities
   - to store and update \( C \in \mathbb{R}^{N \times N} \)
   - to compute the eigen decomposition of \( C \)

2. \( \mathcal{O}(1/N^2) \) Learning Rates for \( C \)-Update
   - \( c_\mu \approx \mu_w/N^2 \)
   - \( c_1 \approx 2/N^2 \)

Exploit prior knowledge on the problem structure such as separability

⇒ decrease the degrees of freedom of the covariance matrix for
   • less time and space complexities
   • a higher learning rates that potentially accelerate the adaptation
What can/should the users do? Restricted Covariance Matrix

Variants with Restricted Covariance Matrix

CMA-ES Variants with Restricted Covariance Matrices

- **Sep-CMA** [Ros and Hansen, 2008]
  - $C = D$. $D$: Diagonal

- **VD-CMA** [Akimoto et al., 2014]
  - $C = D(I + vv^T)D$. $D$: Diagonal, $v \in \mathbb{R}^N$.

- **LM-CMA** [Loshchilov, 2014]
  - $C = I + \sum_{i=1}^{k} v_i v_i^T$. $v_i \in \mathbb{R}^N$.

- **Vkd-CMA** [Akimoto and Hansen, 2016]
  - $C = D(I + \sum_{i=1}^{k} v_i v_i^T)D$. $v_i \in \mathbb{R}^N$.


What can/should the users do? Restricted Covariance Matrix

Separable CMA (Sep-CMA)

\[ \mathcal{N}(m, \sigma^2 I) \sim m + \sigma \mathcal{N}(0, I) \] one degree of freedom \( \sigma \)

\[ \mathcal{N}(m, D^2) \sim m + D \mathcal{N}(0, I) \] \( n \) degrees of freedom

\[ \mathcal{N}(m, C) \sim m + C^{1/2} \mathcal{N}(0, I) \] \((n^2 + n)/2\) degrees of freedom

CMA

\[ C_{cma}^{(t+1)} = C^{(t)} + c_1 \left( p_c p_c^T - C^{(t)} \right) + c_\mu \sum_{i=1}^{\mu} w_i \left( (x_i - m^{(t)})(x_i - m^{(t)})^T - C^{(t)} \right) \]

SEP

\[ C_{sep}^{(t+1)} k, k = [C^{(t)}] k, k + c_1 \left( [p_c]^2 - [C^{(t)}] k, k \right) + c_\mu \sum_{i=1}^{\mu} w_i \left( [x_i - m^{(t)}]^2 - [C^{(t)}] k, k \right) \]

\( (N + 2)/3 \) times greater than CMA
Demo: On 100D Separable Ellipsoid Function

- CMA needed 10 times more FEs + more CPU time
- However, Sep-CMA won't be able to solve rotated ellipsoid function as efficiently as it solves separable ellipsoid
Summary and Final Remarks
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 $g(f(x))$ 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 $C \propto H^{-1}$ adapts a variable metric $\iff$ new (rotated) problem representation $\implies f : x \mapsto g(x^THx)$ reduces to $x \mapsto x^Tx$
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
http://cma.gforge.inria.fr/cmaes_sourcecode_page.html
Comparison during BBOB at GECCO 2010
24 functions and 20+ algorithms in 20-D