Covariance Matrix Adaptation
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

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

initial distribution, \( C = I \)
Covariance Matrix Adaptation
Rank-One Update

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

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

Rank-One Update

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

\[ y_w, \text{ movement of the population mean } m \text{ (disregarding } \sigma) \]
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

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

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

Rank-One Update

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

new distribution (disregarding \( \sigma \))
Covariance Matrix Adaptation
Rank-One Update

\[ m \leftarrow m + \sigma y_w, \quad y_w = \sum_{i=1}^{\mu} w_i y_{i:v}, \quad y_i \sim \mathcal{N}_i(0, C) \]
Covariance Matrix Adaptation
Rank-One Update

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

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

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

new distribution, 
\[ C \leftarrow 0.8 \times C + 0.2 \times y_w y_w^T \]
the ruling principle: the adaptation increases the likelihood of successful steps, \( y_w \), to appear again
Covariance Matrix Adaptation

Rank-One Update

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

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

\[
\mathbf{m} \leftarrow \mathbf{m} + \sigma \mathbf{y}_w \quad \text{where } \mathbf{y}_w = \sum_{i=1}^{\mu} w_i \mathbf{y}_i : \lambda
\]

\[
\mathbf{C} \leftarrow (1 - c_{\text{cov}})\mathbf{C} + c_{\text{cov}} \mu_w \mathbf{y}_w \mathbf{y}_w^T \quad \text{where } \mu_w = \frac{1}{\sum_{i=1}^{\mu} w_i^2} \geq 1
\]
Problem Statement

Stochastic search algorithms - basics

Adaptive Evolution Strategies
   Mean Vector Adaptation
   Step-size control
   Covariance Matrix Adaptation
      Rank-One Update
   Cumulation—the Evolution Path
   Rank-\(\mu\) Update
Evolution Path

Conceptually, the evolution path is the search path the strategy takes over a number of iteration 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:

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

exponentially
fading weights
Evolution Path

Conceptually, the evolution path is the search path the strategy takes over a number of iteration 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

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

exponentially fading weights

The recursive construction of the evolution path (cumulation):

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

where $\mu_w = \frac{1}{\sum w_i^2}, c_c \ll 1$. History information is accumulated in the evolution path.
We used $y_wy_w^T$ for updating $C$. Because $y_wy_w^T = -y_w(-y_w)^T$ the sign of $y_w$ is lost.
We used $y_w y_w^T$ for updating $C$. Because $y_w y_w^T = -y_w (-y)^T$ the sign of $y_w$ is lost.
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 is (re-)introduced by using the evolution path.

$$
\begin{align*}
\mathbf{p}_c & \leftarrow (1 - \mathbf{c}_c) \mathbf{p}_c + \sqrt{1 - (1 - \mathbf{c}_c)^2} \sqrt{\mu_w} \mathbf{y}_w \\
\mathbf{C} & \leftarrow (1 - \mathbf{c}_{cov}) \mathbf{C} + \mathbf{c}_{cov} \mathbf{p}_c \mathbf{p}_c^T
\end{align*}
$$

where $\mu_w = \frac{1}{\sum_{w} v_w}$, $\mathbf{c}_c \ll 1$. 

The sign information is introduced by using the evolution path.
Using an evolution path for the rank-one update of the covariance matrix reduces the number of function evaluations to adapt to a straight ridge from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$.\(^{(3)}\)

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

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

\[
\mathbf{C}_\mu = \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda} \mathbf{y}_{i:\lambda}^T
\]

computes a weighted mean of the outer products of the best \(\mu\) steps and has rank \(\min(\mu, n)\) with probability one.
The rank-$\mu$ update extends the update rule for large population sizes $\lambda$ using $\mu > 1$ vectors to update $\cC$ at each iteration step. The matrix
\[
\cC_{\mu} = \sum_{i=1}^{\mu} \cW_i \cY_{i:\lambda} \cY_{i:\lambda}^T
\]
computes a weighted mean of the outer products of the best $\mu$ steps and has rank $\min(\mu, n)$ with probability one.

The rank-$\mu$ update then reads
\[
\cC \leftarrow (1 - c_{\text{cov}}) \cC + c_{\text{cov}} \cC_{\mu}
\]
where $c_{\text{cov}} \approx \mu w / n^2$ and $c_{\text{cov}} \leq 1$. 
\[ x_i = m + \sigma y_i, \quad y_i \sim \mathcal{N}(0, C) \]

\[ C_\mu = \frac{1}{\mu} \sum y_{i;\lambda} y_{i;\lambda}^T \quad (1 - 1) \times C + 1 \times C_\mu \]

\[ m_{\text{new}} \leftarrow m + \frac{1}{\mu} \sum y_{i;\lambda} \]

- **Sampling of** \( \lambda = 150 \) solutions
- **Calculating** \( C \) where \( \mu = 50 \)
- **New distribution**

where \( C = I \) and \( \sigma = 1 \)

\[ w_1 = \cdots = w_\mu = \frac{1}{\mu}, \quad c_{\text{cov}} = 1 \]
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 iterations roughly from $O(n^2)$ to $O(n)$ \(^{(4)}\)

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

\[ \text{given } \mu_w \propto \lambda \propto n \]

say $\lambda \geq 3n + 10$

---

The rank-$\mu$ update

- increases the possible learning rate in large populations roughly from $2/n^2$ to $\frac{\mu_w}{n^2}$
- can reduce the number of necessary iterations roughly from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ \(^4\)

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

---

The rank-$\mu$ update

- increases the possible learning rate in large populations roughly from $\frac{2}{n^2}$ to $\frac{\mu_w}{n^2}$
- can reduce the number of necessary iterations roughly from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ \(^4\)

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

Summary of Equations
The Covariance Matrix Adaptation Evolution Strategy

Input: \( m \in \mathbb{R}^n, \sigma \in \mathbb{R}_+, \lambda \)

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

Set: \( c_c \approx 4/n, c_\sigma \approx 4/n, c_1 \approx 2/n^2, c_\mu \approx \mu_w/n^2, c_1 + c_\mu \leq 1, \)
\( d_\sigma \approx 1 + \frac{\mu_w}{n} \), and \( w_i=1...\lambda \) such that \( \mu_w = \frac{1}{\sum_{i=1}^{\mu} w_i^2} \approx 0.3 \lambda \)

While not terminate

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

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

\[ p_c \leftarrow (1 - c_c) p_c + \mathds{1}_{\{\|p_\sigma\| < 1.5 \sqrt{n}\}} \sqrt{1 - (1 - c_c)^2} \sqrt{\mu_w y_w} \]

\[ p_\sigma \leftarrow (1 - c_\sigma) p_\sigma + \sqrt{1 - (1 - c_\sigma)^2} \sqrt{\mu_w} C^{-\frac{1}{2}} y_w \]

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

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

Not covered on this slide: termination, restarts, useful output, boundaries and encoding
Rank-one and Rank-mu updates
Rank-one and Rank-\(\mu\) update - default pop size

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

\( \lambda = 10 \) (default for \( N = 10 \))
Rank-one and Rank-\(\mu\) update - larger pop size

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

\(\lambda = 50\)
What did we want to achieve?

- reduce any convex-quadratic function

\[ f(x) = x^T H x \]

- to the sphere model

\[ f(x) = x^T x \]

- lines of equal density align with lines of equal fitness

\[ \mathbb{C} \propto H^{-1} \]

in a stochastic sense

- e.g. \( f(x) = \sum_{i=1}^{n} 10^6 \frac{i-1}{n-1} x_i^2 \)
Experimentum Crucis (1)

\[ f \text{ convex quadratic, separable} \]

\[ f(x) = \sum_{i=1}^{n} 10^{\alpha \frac{i-1}{n-1}} x_i^2, \quad \alpha = 6 \]
Experimentum Crucis (2)

$f$ convex quadratic, as before but non-separable (rotated)

\[ f(x) = g(x^T H x), \quad g : \mathbb{R} \to \mathbb{R} \text{ strictly increasing} \]

\[ f = 7.91055728188042 \times 10^{-10} \]

\[ x(1) = 2.0052 \times 10^{-06} \]
\[ x(2) = 2.1131 \times 10^{-06} \]
\[ x(3) = 2.0364 \times 10^{-06} \]
\[ x(4) = 2.9981 \times 10^{-07} \]
\[ x(5) = 1.2468 \times 10^{-07} \]
\[ x(6) = 1.2552 \times 10^{-06} \]
\[ x(7) = 8.3583 \times 10^{-08} \]
\[ x(8) = 1.2468 \times 10^{-06} \]
\[ x(9) = 7.3812 \times 10^{-08} \]

\[ C \propto H^{-1} \text{ for all } g, H \]
On Invariances
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
Rotational Invariance in Search Space

- Invariance to orthogonal (rigid) transformations $\mathbf{R}$, where $\mathbf{R}\mathbf{R}^T = \mathbf{I}$
  
e.g. true for simple evolution strategies
  recombination operators might jeopardize rotational invariance

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

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

\[
\begin{align*}
  f : & \quad \mathbf{x} \mapsto f(\mathbf{x}), \quad \mathbf{x}^{(t=0)} = \mathbf{x}_0 \\
  f_\mathbf{R} : & \quad \mathbf{x} \mapsto f(\mathbf{R}\mathbf{x}), \quad \mathbf{x}^{(t=0)} = \mathbf{R}^{-1}(\mathbf{x}_0)
\end{align*}
\]

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

---


Main Invariances in Optimization

Invariance to strictly increasing transformations of $f$: identical behavior when optimizing

$$x \mapsto f(x)$$
$$x \mapsto g(f(x)) \quad \text{where } g : \text{Im}(f) \to \mathbb{R} \text{ is strictly increasing}$$

Translation invariance: identical behavior when optimizing

$$x \mapsto f(x)$$
$$x \mapsto f(x - a) \text{ for all } a \in \mathbb{R}^n$$

Scale invariance: identical behavior when optimizing

$$x \mapsto f(x)$$
$$x \mapsto f(\alpha x) \text{ for all } \alpha \in \mathbb{R}_>$$

Rotational invariance: identical behavior when optimizing

$$x \mapsto f(x)$$
$$x \mapsto f(Rx) \text{ for all } R \text{ is an orthogonal matrix}$$

Affine invariance: identical behavior when optimizing

$$x \mapsto f(x)$$
$$x \mapsto f(Ax + b) \text{ for all } A \in \mathbb{R}^{n \times n} \text{ an invertible matrix and } b \in \mathbb{R}^n$$
Main Invariances in Optimization

Invariance to strictly increasing transformations of f: identical behavior when optimizing

\[ x \mapsto f(x) \]
\[ x \mapsto g(f(x)) \text{ where } g : \text{Im}(f) \to \mathbb{R} \text{ is strictly increasing} \]

Translation invariance: identical behavior when optimizing

\[ x \mapsto f(x) \]
\[ x \mapsto f(x - a) \text{ for all } a \in \mathbb{R}^n \]

Scale invariance: identical behavior when optimizing

\[ x \mapsto f(x) \]
\[ x \mapsto f(\alpha x) \text{ for all } \alpha \in \mathbb{R}_> \]

Rotational invariance: identical behavior when optimizing

\[ x \mapsto f(x) \]
\[ x \mapsto f(Rx) \text{ for all } R \text{ an orthogonal matrix} \]

Affine invariance: identical behavior when optimizing

\[ x \mapsto f(x) \]
\[ x \mapsto f(Ax + b) \text{ for all } A \in \mathbb{R}^{n \times n} \text{ an invertible matrix and } b \in \mathbb{R}^n \]
Hierarchy of Invariance

Affine invariance

Rotational Invariance  Scale-invariance  translation invariance
Exercice - Invariances of (1+1)-ES and CMA-ES

<table>
<thead>
<tr>
<th>CMA-ES</th>
<th>(1+1)-ES with one-fifth success rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>translation invariance</td>
<td></td>
</tr>
<tr>
<td>scale invariance</td>
<td></td>
</tr>
<tr>
<td>rotational invariance</td>
<td></td>
</tr>
<tr>
<td>affine invariance</td>
<td></td>
</tr>
</tbody>
</table>
Testing for invariances
Comparison to BFGS, NEWUOA, PSO and DE

$f$ convex quadratic, separable with varying condition number $\alpha$

Ellipsoid dimension 20, 21 trials, tolerance $1e^{-09}$, eval max $1e+07$

BFGS (Broyden et al 1970)
NEWUOA (Powell 2004)
DE (Storn & Price 1996)
PSO (Kennedy & Eberhart 1995)
CMA-ES (Hansen & Ostermeier 2001)

$f(x) = g(x^T H x)$ with

- $H$ diagonal
- $g$ identity (for BFGS and NEWUOA)
- $g$ any order-preserving = strictly increasing function (for all other)

$SP1 = \text{average number of objective function evaluations}^5$ to reach the target function value of $g^{-1}(10^{-9})$

---

$^5$ Auger et.al. (2009): Experimental comparisons of derivative free optimization algorithms, SEA
Comparison to BFGS, NEWUOA, PSO and DE

$f$ convex quadratic, non-separable (rotated) with varying condition number $\alpha$

Rotated Ellipsoid dimension 20, 21 trials, tolerance $1e^{-09}$, eval max $1e+07$

BFGS (Broyden et al 1970)
NEWUOA (Powell 2004)
DE (Storn & Price 1996)
PSO (Kennedy & Eberhart 1995)
CMA-ES (Hansen & Ostermeier 2001)

$f(x) = g(x^T H x)$ with $H$ full
$g$ identity (for BFGS and NEWUOA)
$g$ any order-preserving = strictly increasing function (for all other)

SP1 = average number of objective function evaluations\(^6\) to reach the target function value of $g^{-1}(10^{-9})$

---

\(^6\) Auger et.al. (2009): Experimental comparisons of derivative free optimization algorithms, SEA
Comparison to BFGS, NEWUOA, PSO and DE

$f$ non-convex, non-separable (rotated) with varying condition number $\alpha$

BFGS (Broyden et al 1970)
NEWUOA (Powell 2004)
DE (Storn & Price 1996)
PSO (Kennedy & Eberhart 1995)
CMA-ES (Hansen & Ostermeier 2001)

$f(x) = g(x^T H x)$ with $H$ full
$g : x \mapsto x^{1/4}$ (for BFGS and NEWUOA)
$g$ any order-preserving = strictly increasing function (for all other)

\[ SP1 = \text{average number of objective function evaluations} \] to reach the target function value of $g^{-1}(10^{-9})$

---

Auger et al. (2009): Experimental comparisons of derivative free optimization algorithms, SEA
Comparison during BBOB at GECCO 2009
24 functions and 31 algorithms in 20-D
Comparison during BBOB at GECCO 2010
24 functions and 20+ algorithms in 20-D
Comparison during BBOB at GECCO 2009

30 noisy functions and 20 algorithms in 20-D

---

![Comparison diagram showing the performance of 20 algorithms across 30 noisy functions in 20-D. The x-axis represents the running length/dimension, and the y-axis represents the proportion of functions. Various algorithms are plotted, with lines indicating their performance over the dimensions.](image-url)
Comparison during BBOB at GECCO 2010
30 noisy functions and 10+ algorithms in 20-D
Problem Statement

Stochastic search algorithms - basics

Adaptive Evolution Strategies
  Mean Vector Adaptation
  Step-size control
  Covariance Matrix Adaptation
    Rank-One Update
    Cumulation—the Evolution Path
  Rank-\(\mu\) Update
The Continuous Search Problem

Difficulties of a non-linear optimization problem are

- dimensionality and non-separabibility
demands to exploit problem structure, e.g. neighborhood

- ill-conditioning
demands to acquire a second order model

- ruggedness
demands a non-local (stochastic?) approach

Approach: population based stochastic search, coordinate system independent and with second order estimations (covariances)
Main Features 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
   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

   \[ C \propto H^{-1} \iff \text{adapts a variable metric} \]
   \[ \iff \text{new (rotated) problem representation} \]
   \[ \implies f(x) = g(x^T H x) \text{ reduces to } g(x^T x) \]
Limitations of CMA Evolution Strategies

- **internal CPU-time**: $10^{-8} n^2$ seconds per function evaluation on a 2GHz PC, tweaks are available. 100,000 $f$-evaluations in 1000-D take 1/4 hours internal CPU-time

- better methods are presumably available in case of
  - partly separable problems
  - specific problems, for example with cheap gradients specific methods
  - small dimension ($n \ll 10$) for example Nelder-Mead
  - small running times (number of $f$-evaluations $\ll 100n$) model-based methods