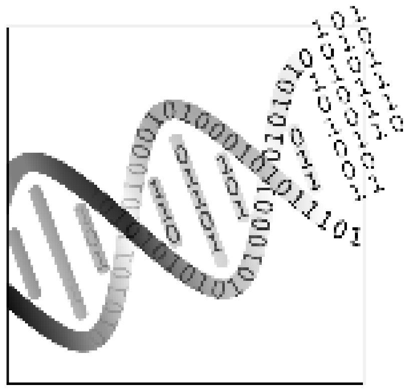
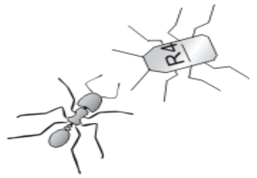
Evolutionary Strategies





Evolutionary Strategies (ES) *Rechenberg, 1973*

Genetic representation = Vector of *n* real-valued numbers

Population = fixed size

μ = number of selected parents

λ = number of individuals in the population

Selection = truncated rank selection

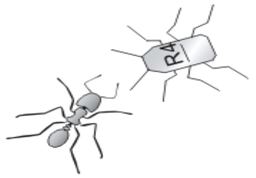
Two variants:

(μ, λ) = selected parents are replaced by their offspring

$(\mu + \lambda)$ = selected parents coexist with their offspring

Mutation = perturbations of all genes with normal probability density function

Crossover = not used

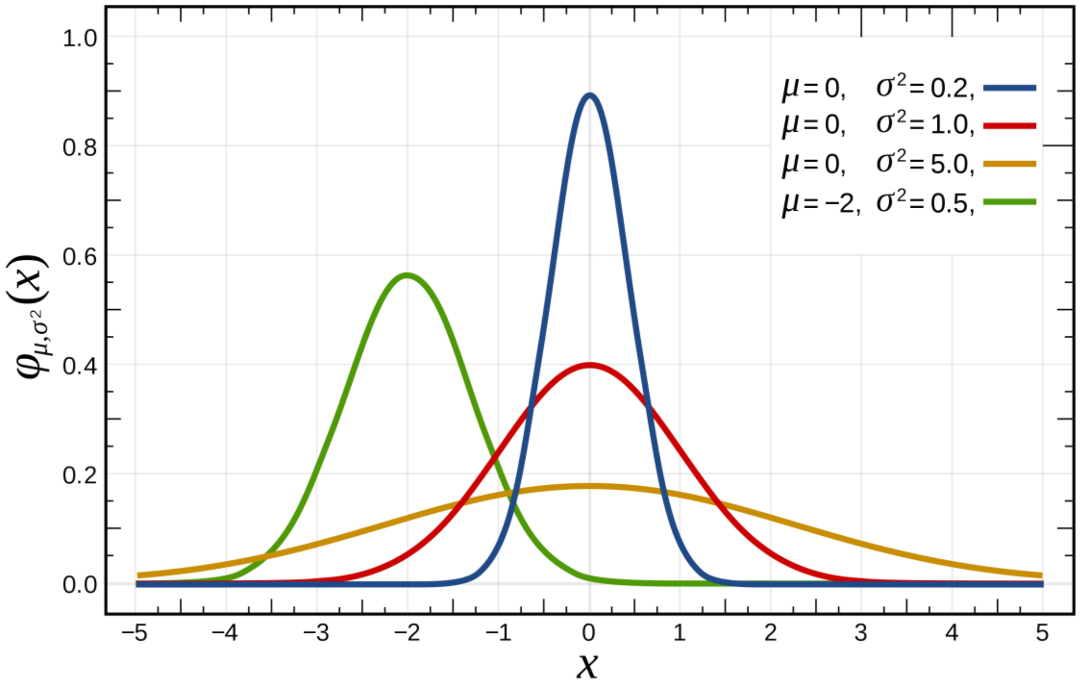


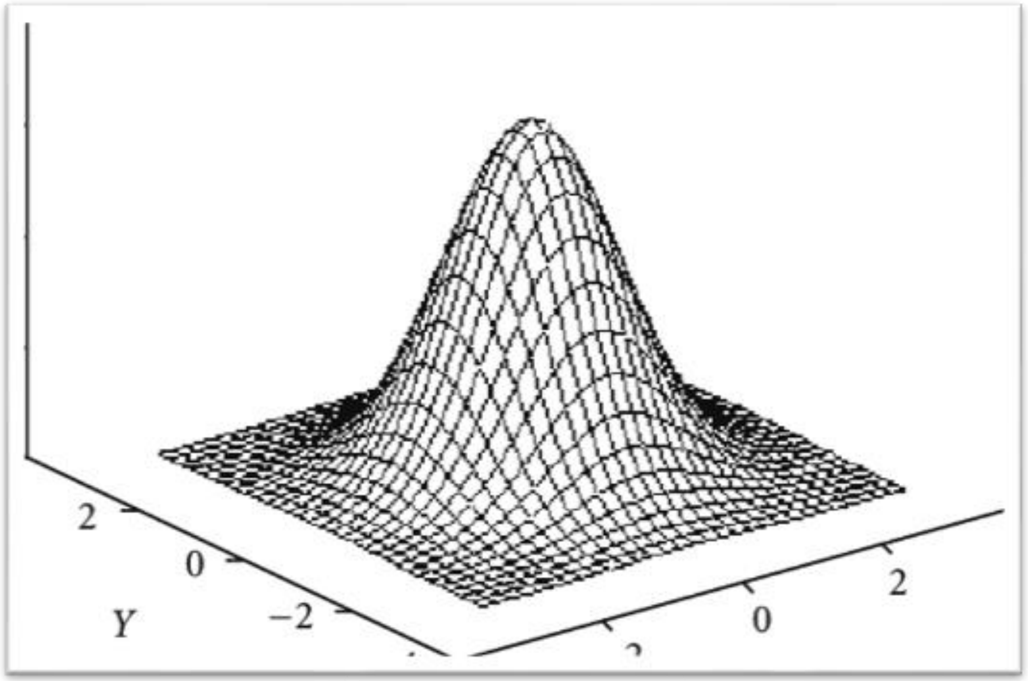
Mutations are Gaussian perturbations

Mutate each gene x by adding a number sampled from a Normal distribution N(0, σ)

$x' = x + N(0,\sigma) = x + \sigma N(0, 1)$

For genetic strings of length *n>1* (e.g. $\langle x_1, x_2 \rangle$), we sample the Normal distribution N(0, I), where I is the Identity Matrix



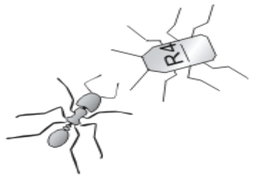


σ is the *mutation size* that defines the amount of change

ES (μ, λ) algorithm: Initialisation

1. Set up population size λ, number of parents μ, mutation step size $\boldsymbol{\sigma}$
2. Create initial real-valued vector $\boldsymbol{\theta}$ (also known as *population mean*)
3. Create population: generate λ offspring by adding mutations to $\boldsymbol{\theta}$

$$\mathbf{x}_i = \theta + \sigma \mathbf{N}_i(0, \mathbf{I}) \qquad \text{for } 0 < i \leq \lambda$$



ES (μ, λ) Algorithm: Selection and Reproduction

4. Evaluate fitness of all λ individuals

5. Select μ parents with Truncated Rank Selection, e.g. top 25%

6. Compute new population mean vector θ as *fitness-weighted* values of μ parents

   $\boldsymbol{\theta} = \sum_{i=1}^{\mu} \mathbf{w}_i \mathbf{x}_i$
   where $w_1 \geq w_2 \geq w_3 \geq w_\mu \geq 0$
   and $\sum_{i=1}^{\mu} \mathbf{w}_i = 1$

7. Create new population [go to step 3] ($\mathbf{x}_i = \theta + \sigma \mathbf{N}_i(0, \mathbf{I})$ for $0 < i \leq \lambda$)

Co-evolution of mutation size

Mutation size σ can be added to genome of individuals and co-evolved $\langle x_1,\ldots,x_n, \sigma \rangle$
$\langle x, \sigma \rangle \rightarrow \langle x', \sigma' \rangle$

Mutation order is important…

1. First apply mutation to mutation size $\sigma \rightarrow \sigma'$
2. Then apply mutation to other locations with mutated mutation size $x \rightarrow x' = x + \sigma' N(0,1)$

…because quality of new offspring $\langle x' ,\sigma' \rangle$ will be evaluated twice

- Primary evaluation: x' is good if f(x') is good
- Secondary evaluation: mutated mutation σ' is good if the x' it created is good

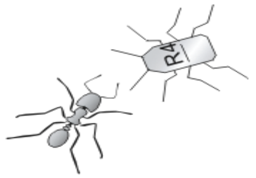
(reversing mutation order this would not work)

Mutations:

- $\sigma' = \sigma \bullet \exp(\tau N(0,1))$

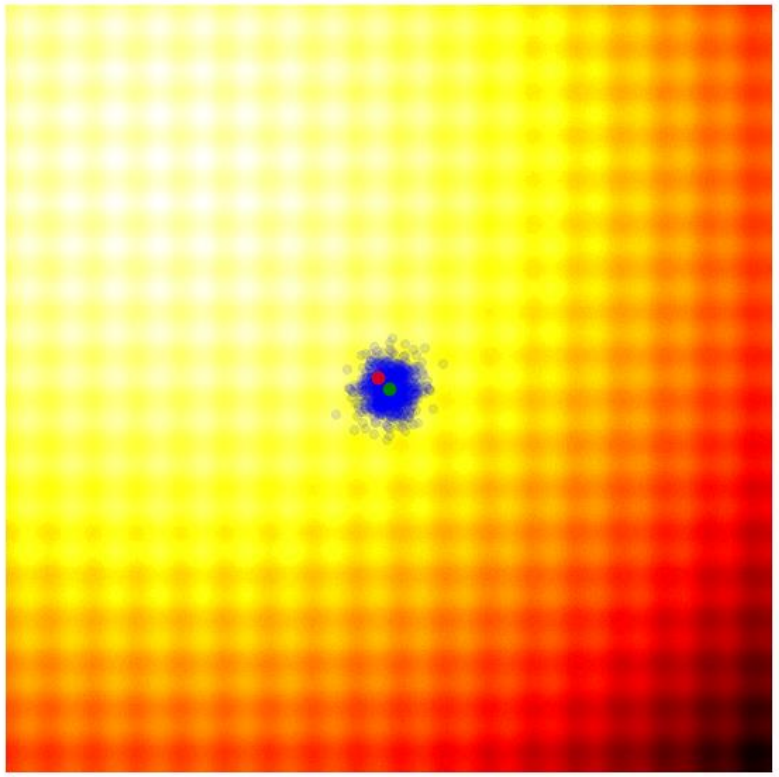
 where $\tau \propto 1/ n^{1/2}$ where n = number of genes; boundary rule: if $\sigma' < \varepsilon_0 \Rightarrow \sigma' = \varepsilon_0$

- $x'_i = x_i + \sigma' N_i(0,1)$

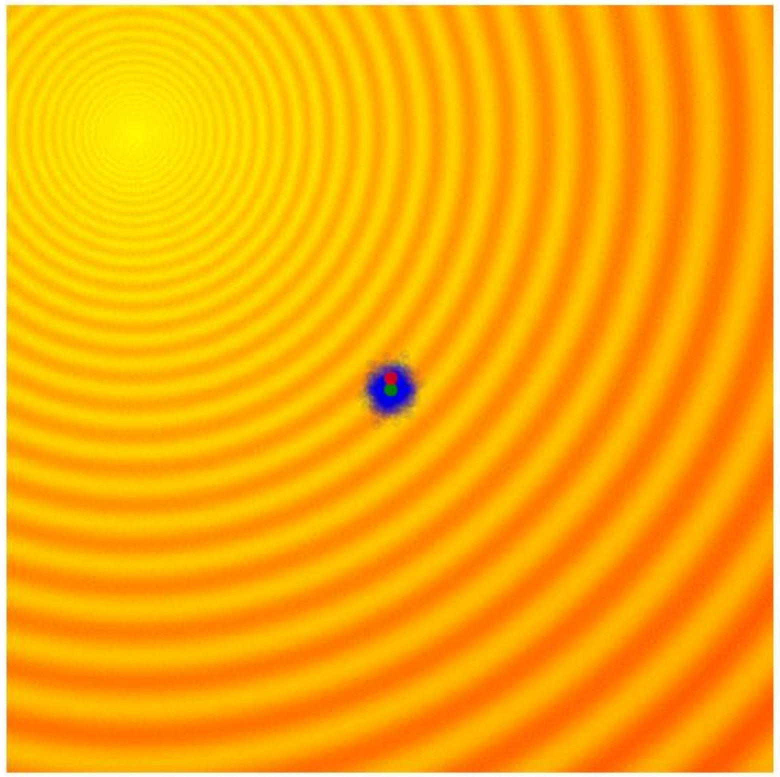


ES (μ, λ) algorithm with adaptive independent mutations

Shifted Schaffer-2D function



Shifted Rastrigin-2D function



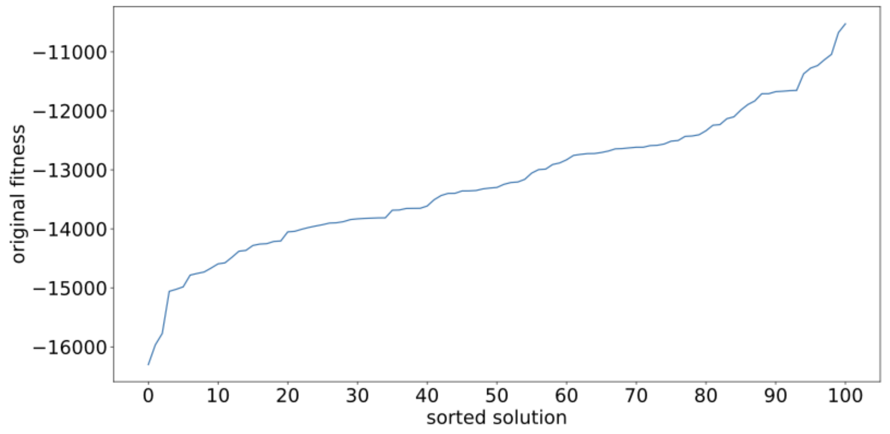
Green dot = mean solution
Blue dots = sampled solutions
Red dot = best solution

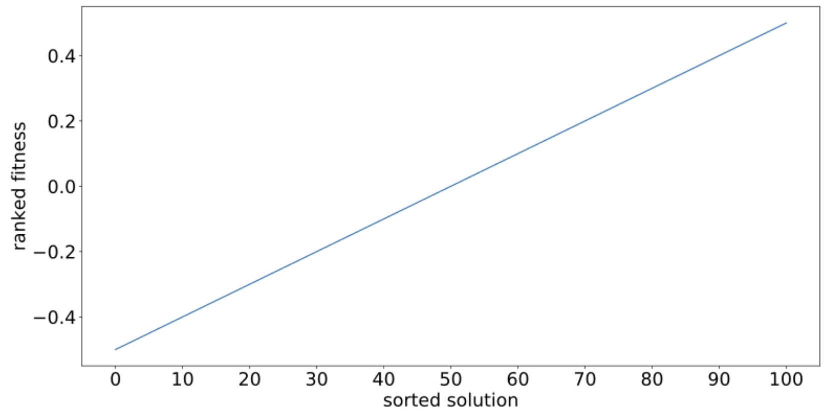
Source: http://blog.otoro.net/2017/10/29/visual-evolution-strategies/

OpenAI ES Salimans et al, 2017

- Constant mutation step size **σ** (not encoded in the genetic string)
- Compute fitness of negative and positive mutated offspring
- Normalized Ranked Fitness of all individuals
- Update population vector with Adam (Adaptive Moment Estimation)

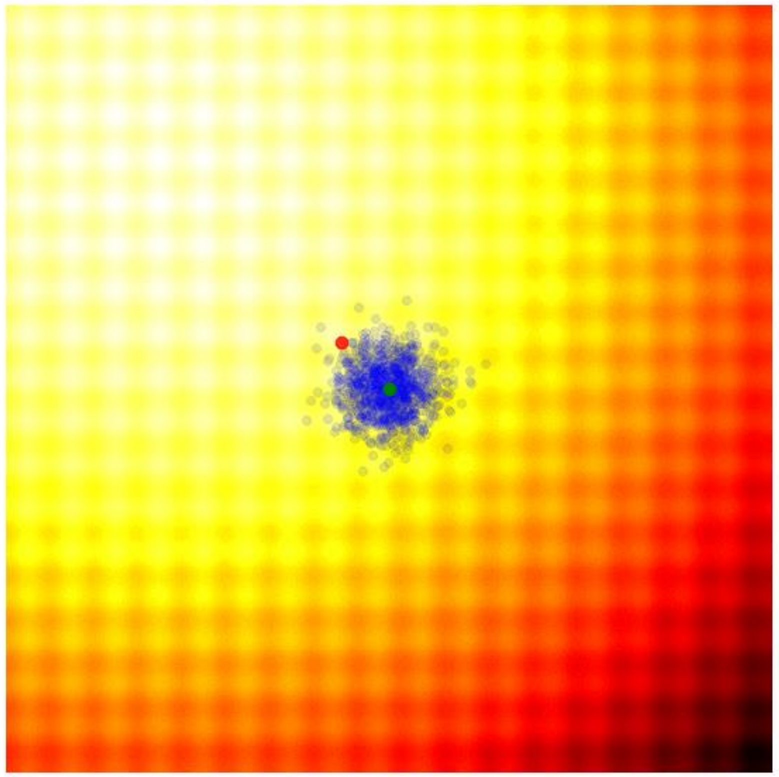
```
σ = 0.02: mutation step size
λ = ~250: population size (total size = λ · 2)   effective λ = 500
θ: policy parameters
F: policy evaluation function
optimizer = Adam

1 initialize $\theta_0$
2 for $g = 1, 2, \ldots$ do
3     for $i = 1, 2, \ldots \lambda$ do
4         sample noise vector: $\varepsilon_i \sim N(0, I)$ # gaussian distribution with variance σ
5         evaluate score: $s_i^+ \leftarrow F(\theta_{t-1} + \sigma * \varepsilon_i)$
6         evaluate score: $s_i^- \leftarrow F(\theta_{t-1} - \sigma * \varepsilon_i)$
7         compute normalized ranks: $u = \text{ranks}(s^+ - s^-)$, $u_i \in [-0.5, 0.5]$
8     estimate gradient: $g_t \leftarrow \frac{1}{\lambda} \sum_{i=1}^{\lambda} (u_i * \varepsilon_i)$
9     $\theta_g = \theta_{g-1}$ + optimizer(g)
```



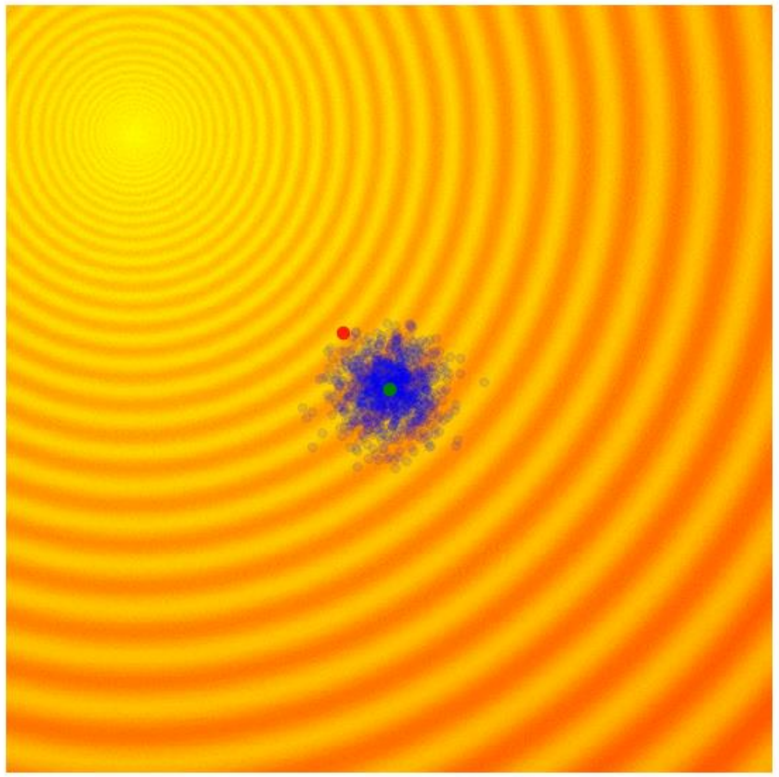


OpenAI ES

Shifted Schaffer-2D function

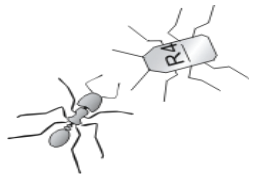


Shifted Rastrigin-2D function



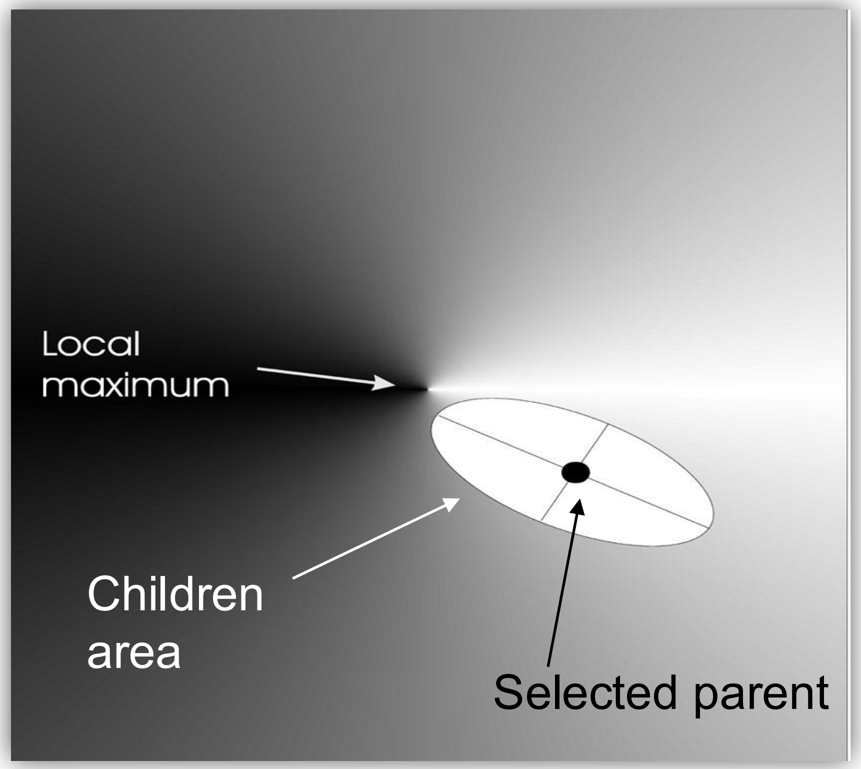
Green dot = mean solution
Blue dots = sampled solutions
Red dot = best solution

Source: http://blog.otoro.net/2017/10/29/visual-evolution-strategies/



Correlated mutation steps

We want to tailor the mutation size at each location to move in the direction of the gradient of the fitness distribution of the population (but without gradient optimizer)



This can be done by using the Covariance matrix of the population instead of the Identity matrix to sample the mutation vector of each individual

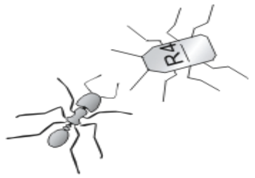
Adapted from: Eiben & Smith: http://www.evolutionarycomputation.org/slides/

Basic reminder of Variance and Covariance

$$var(X) = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2$$, $\bar{X}$ is the mean of the samples of X

$$covar(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})$$

$$covar(X,X) = var(X)$$

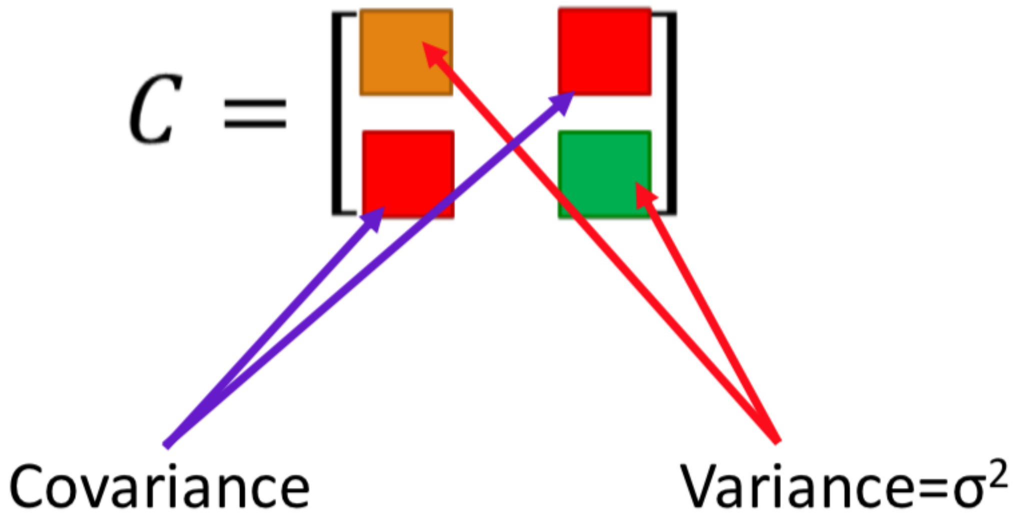


Covariance Matrix

A Covariance Matrix is the matrix whose (ij) element is the covariance between the i and the j element of the data distribution.

For a distribution with two dimensions A (gene x) and B (gene y):

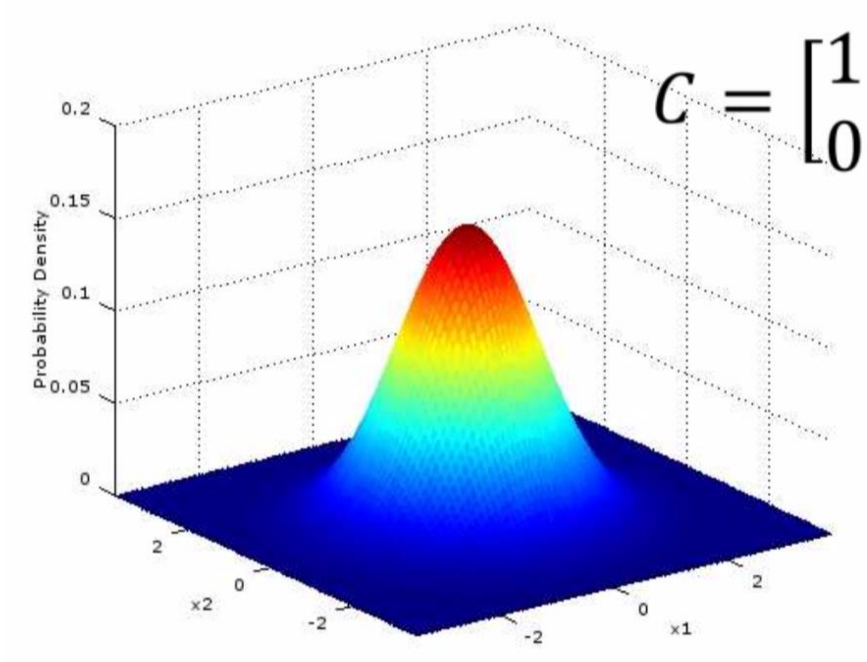
$$\begin{pmatrix} cov(A,A) & cov(A,B) \\ cov(B,A) & cov(B,B) \end{pmatrix} = \begin{pmatrix} var(A) & cov(A,B) \\ cov(B,A) & var(B) \end{pmatrix}$$



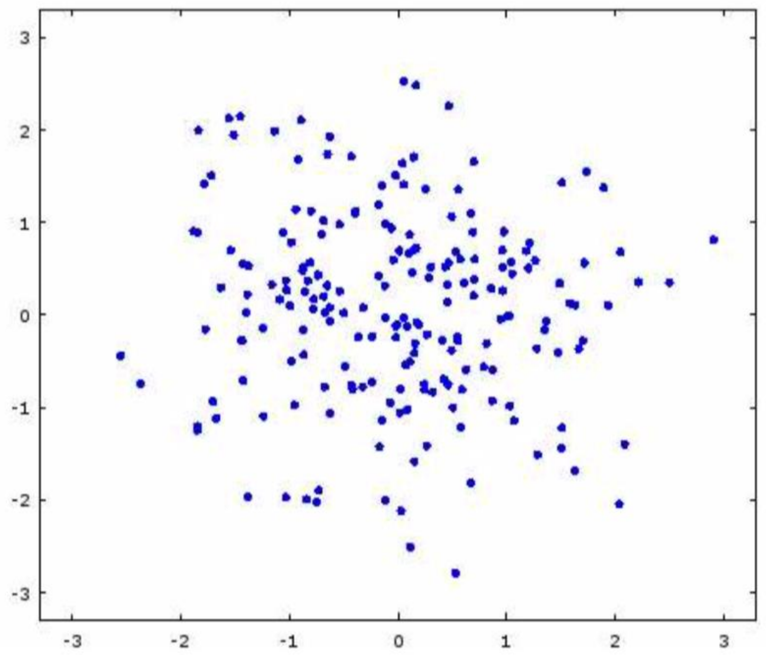
Adapted from: https://www.slideshare.net/OsamaSalaheldin2/cmaes-presentation

Covariance Matrix of variables that don't covary with equal variance

If two elements x, y do not covary and normally distributed, the covariance matrix is equivalent to σ N(0, I)



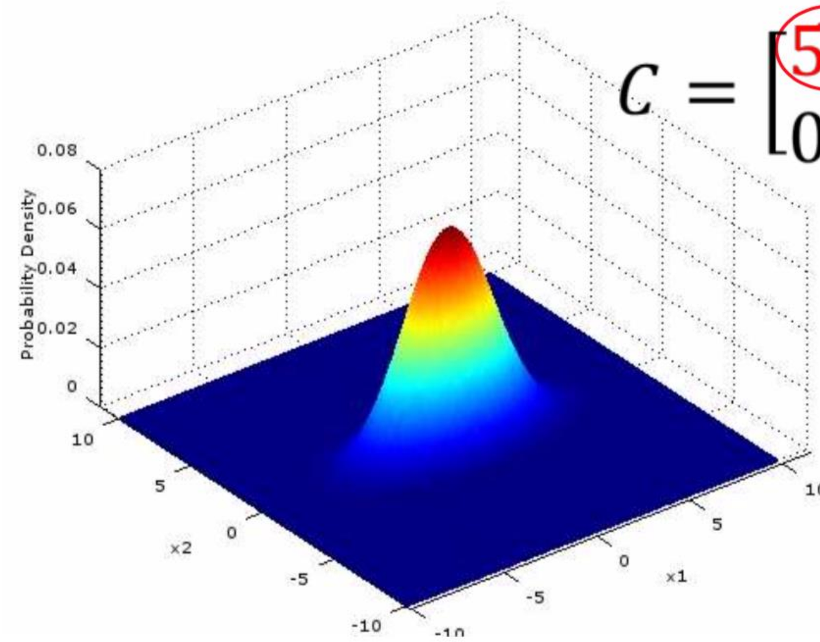
$$C = \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}$$



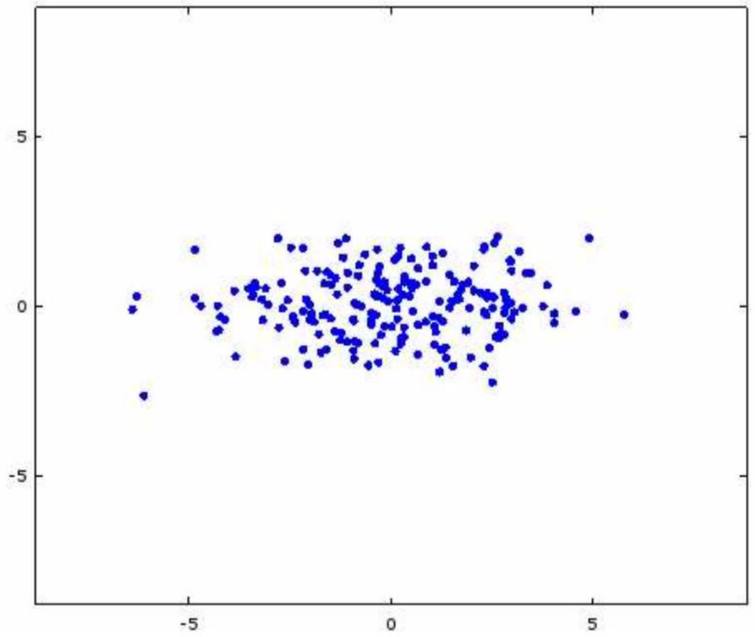
Adapted from: https://www.slideshare.net/OsamaSalaheldin2/cmaes-presentation

Covariance Matrix of variables that don't covary with different variance

If the two elements x, y do not covary, but x has larger variance



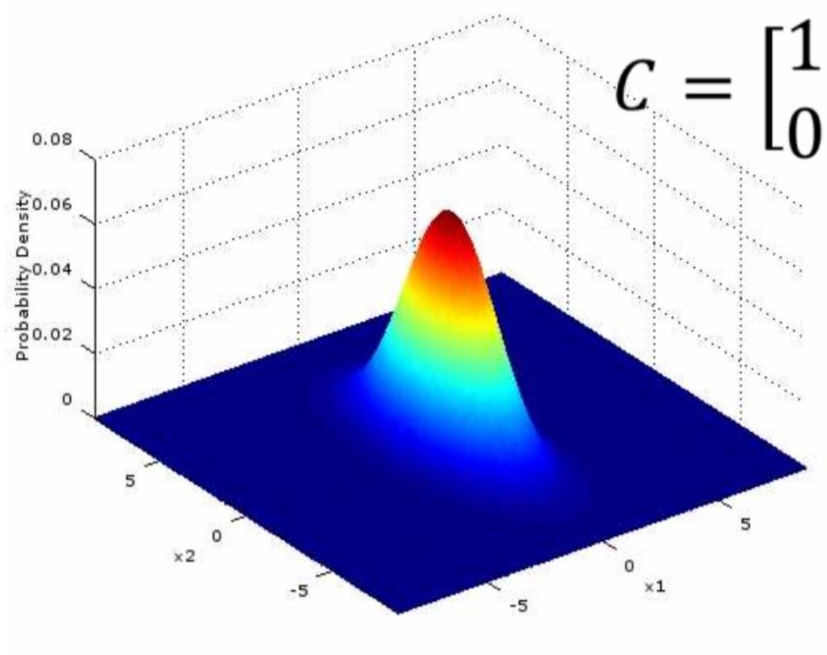
$$C = \begin{bmatrix} \textcolor{red}{\bigcirc\!\!\!\!5.0} & 0.0 \\ 0.0 & 1.0 \end{bmatrix}$$



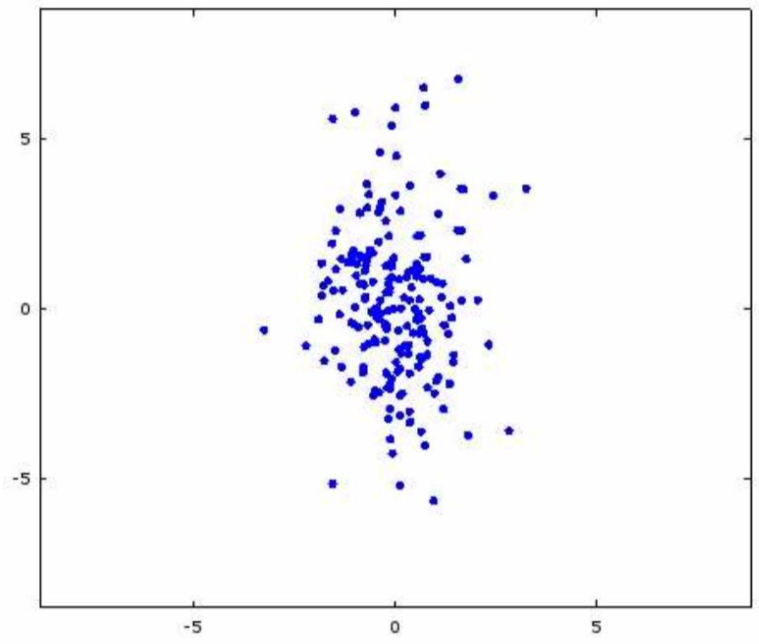
Adapted from: https://www.slideshare.net/OsamaSalaheldin2/cmaes-presentation

Covariance Matrix of variables that don't covary with different variance

If the two elements x, y do not covary, but y has larger variance



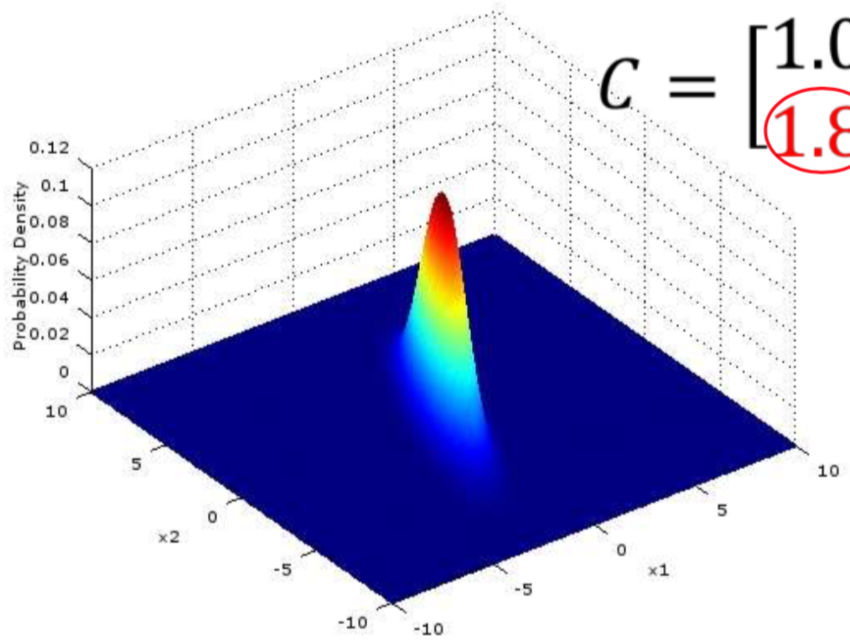
$$C = \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & \boxed{5.0} \end{bmatrix}$$



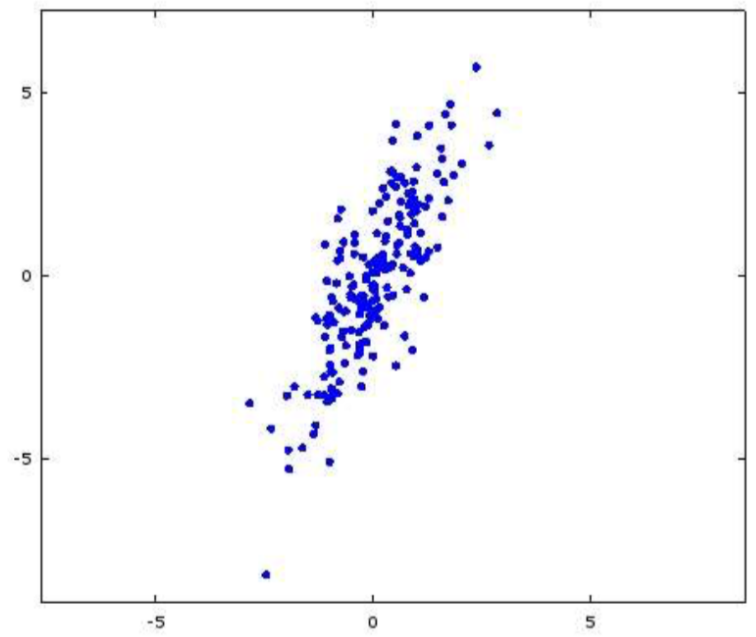
Adapted from: https://www.slideshare.net/OsamaSalaheldin2/cmaes-presentation

Covariance Matrix of variables that covary with different variance

If the two elements x, y covary, and y has larger variance



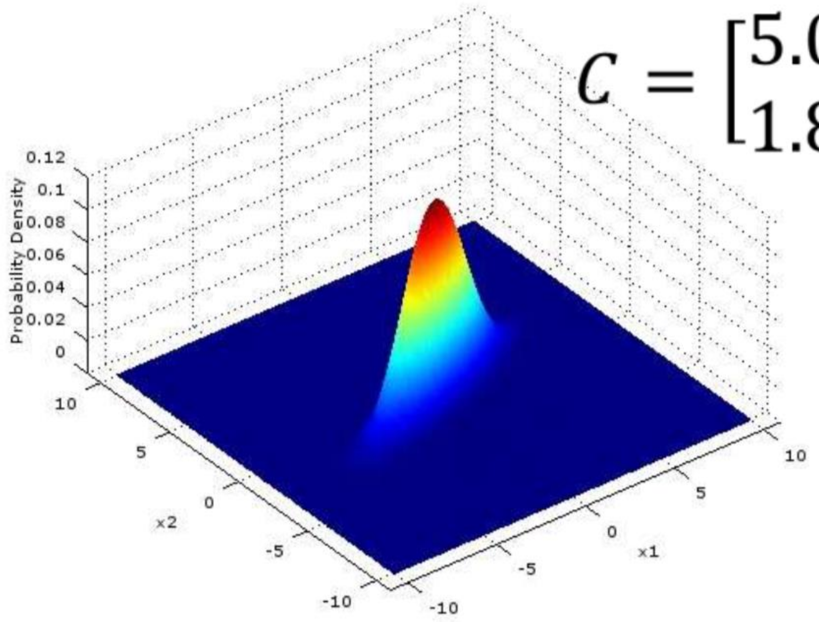
$$C = \begin{bmatrix} 1.0 & 1.8 \\ 1.8 & 5.0 \end{bmatrix}$$



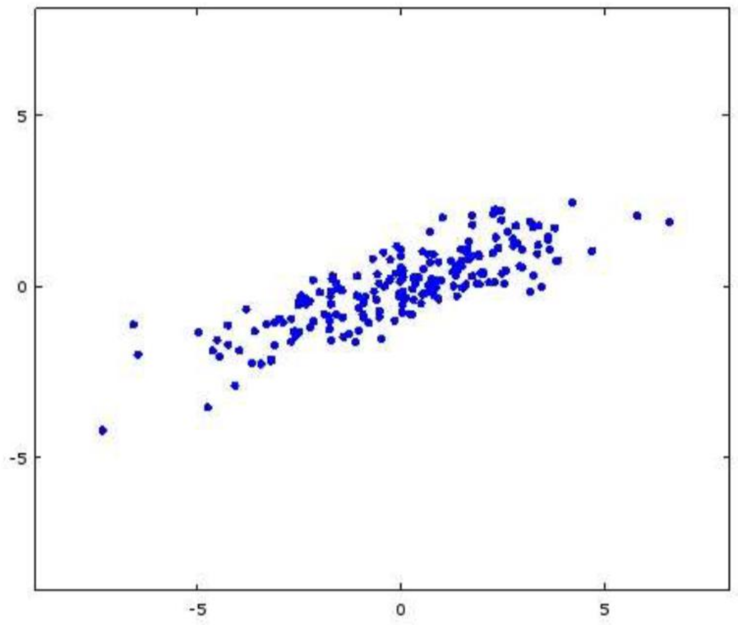
Adapted from: https://www.slideshare.net/OsamaSalaheldin2/cmaes-presentation

Covariance Matrix of variables that covary with different variance

If the two elements x, y covary, and x has larger variance



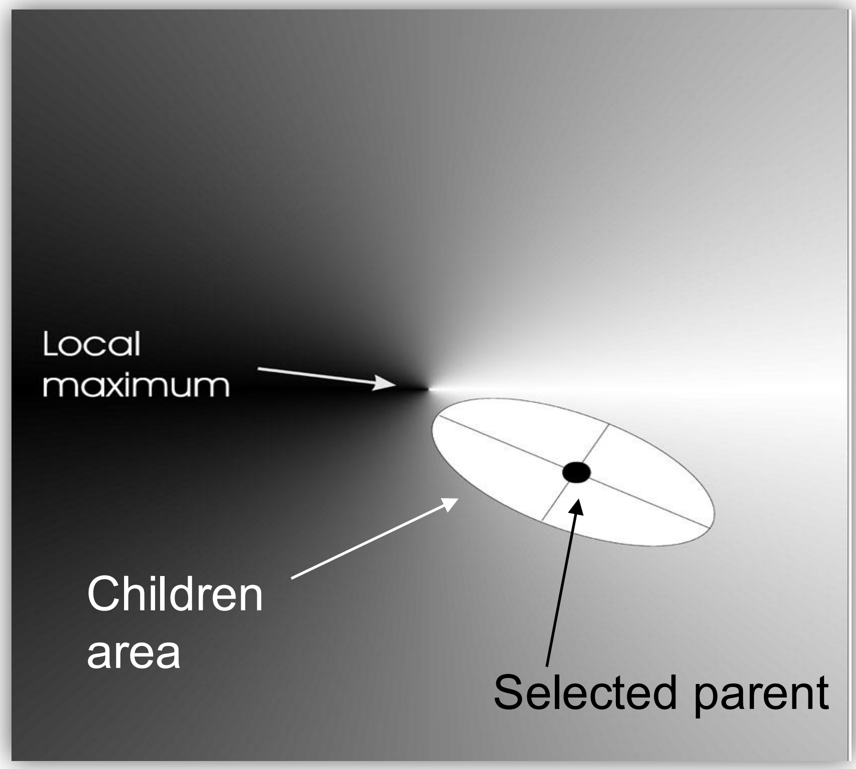
$$C = \begin{bmatrix} 5.0 & 1.8 \\ 1.8 & 1.0 \end{bmatrix}$$



Adapted from: https://www.slideshare.net/OsamaSalaheldin2/cmaes-presentation

Covariance Matrix Adaptation ES (CMA-ES)

Take larger steps in the direction of highest variance, i.e. the population should move faster in the direction of the eigenvector corresponding to the largest eigenvalue of the covariance matrix of the fitness distribution of the current population

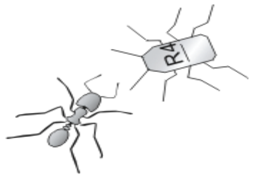


Hansen N, Ostermeier A (2001) *Evolutionary Computation*

CMA-ES Algorithm: Initialise population

1. Initialise Covariance Matrix **C** as n x n Identity Matrix
2. Set up population mean vector $\boldsymbol{\theta}$ (e.g., best guess or distribution center)
3. Set up initial mutation step size vector $\boldsymbol{\sigma}$
4. Generate λ offspring from $\boldsymbol{\theta}$

$$\mathbf{x}_i = \boldsymbol{\theta} + \mathbf{N}_i(\sigma^2, \mathbf{C}) \qquad \text{for } 0 < i \leq \lambda$$



CMA-ES Algorithm: Selection and Reproduction

5. Evaluate λ individuals of the population

6. Identify μ parents with Truncated Rank Selection, e.g. top 25%

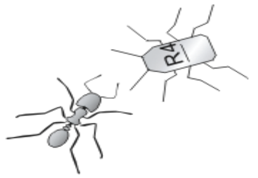
7. Update population mean using weighted values of μ parents

   $\theta = \sum_{i=1}^{\mu} \mathbf{w}_i \mathbf{x}_i$
   where $w_1 \geq w_2 \geq w_3 \geq w_\mu \geq 0$
   and $\sum_{i=1}^{\mu} \mathbf{w}_i = 1$

5. Update **C** covariance matrix with new **C** computed for θ distribution

6. Adapt mutation step size vector **σ**

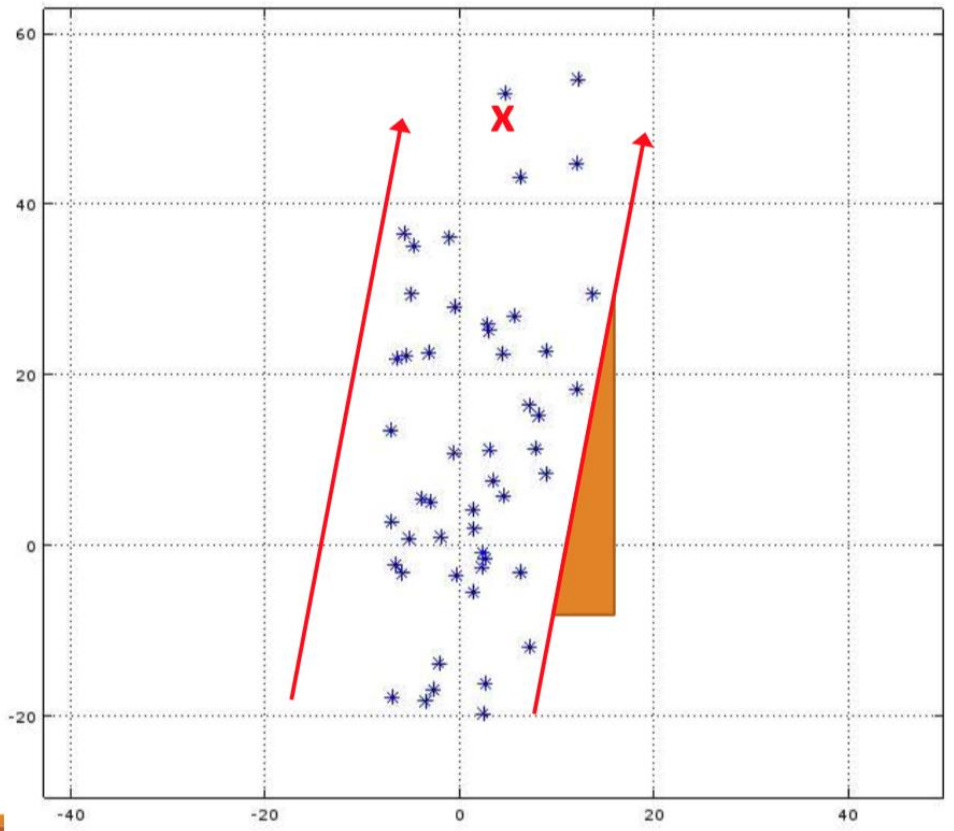
7. Go to step 4 ($\mathbf{x}_i = \theta + \mathbf{N}_i(\sigma^2, \mathbf{C})$ for $0 < i \leq \lambda$)



An example of CMA-ES at work

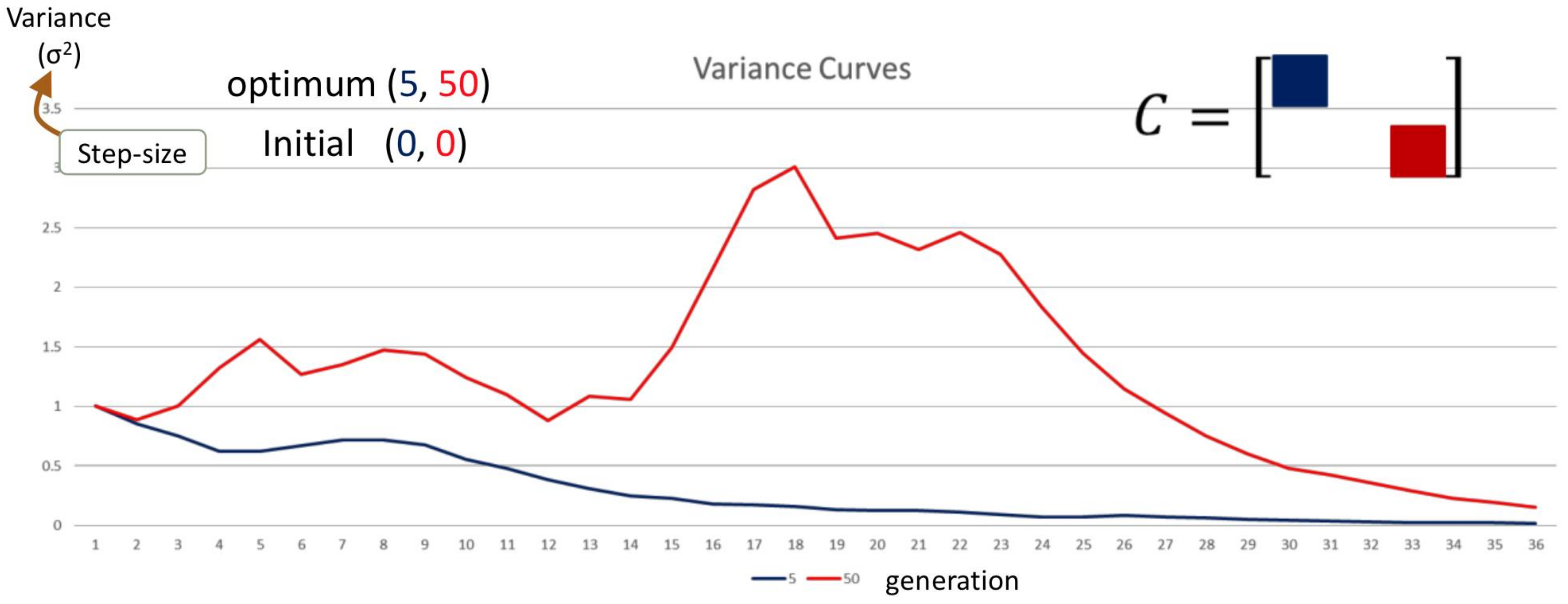
- A practical run of CMA-ES
  - The optimum solution is (5, 50)
  - The initial guess is (0, 0)
  - The population moves faster towards the direction of the second component (50)

$$C = \begin{bmatrix} 1.4011 & 2.0368 \\ 2.0368 & 11.8843 \end{bmatrix}$$



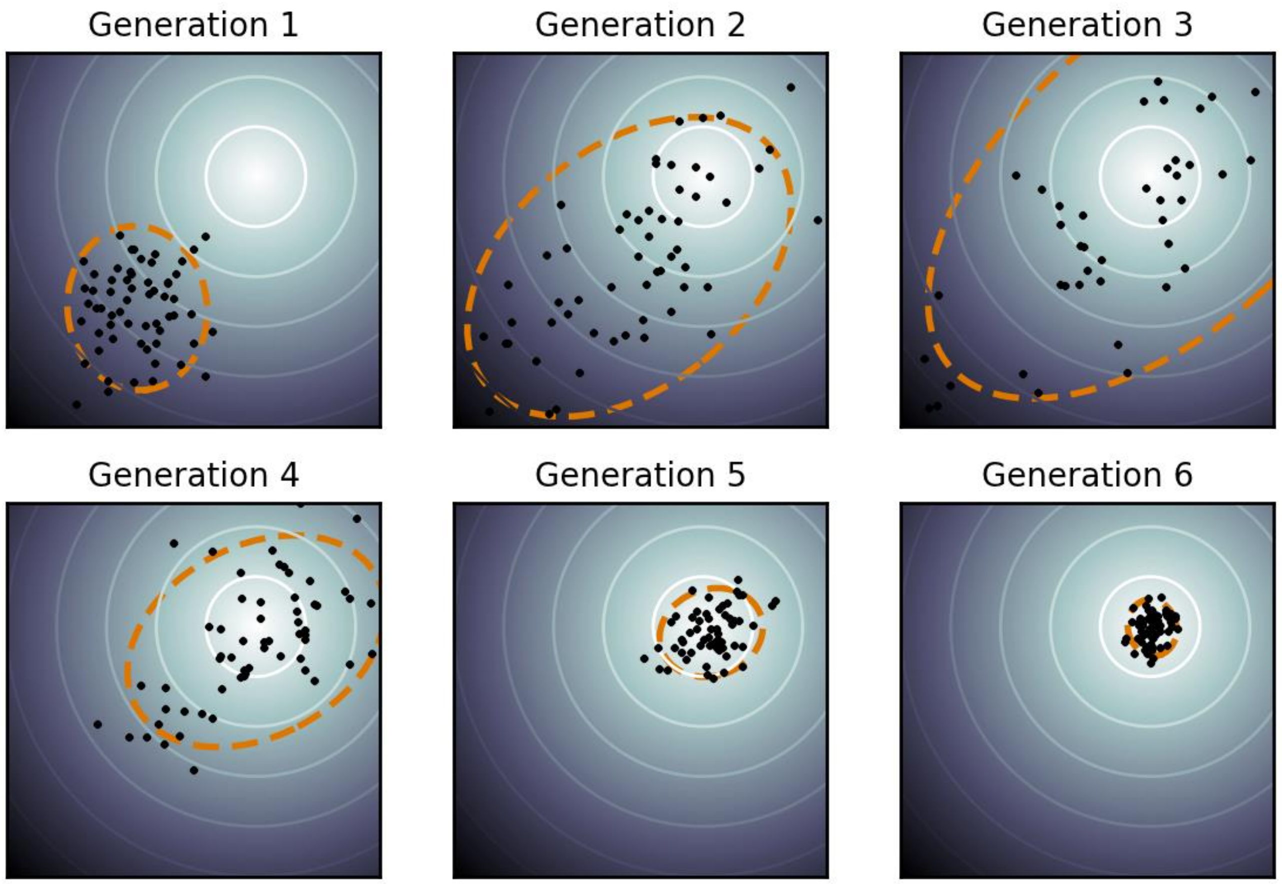
Source: https://www.slideshare.net/OsamaSalaheldin2/cmaes-presentation

An example of CMA-ES at work

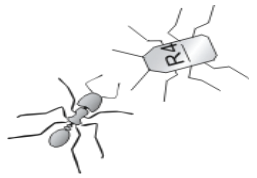


Source: https://www.slideshare.net/OsamaSalaheldin2/cmaes-presentation

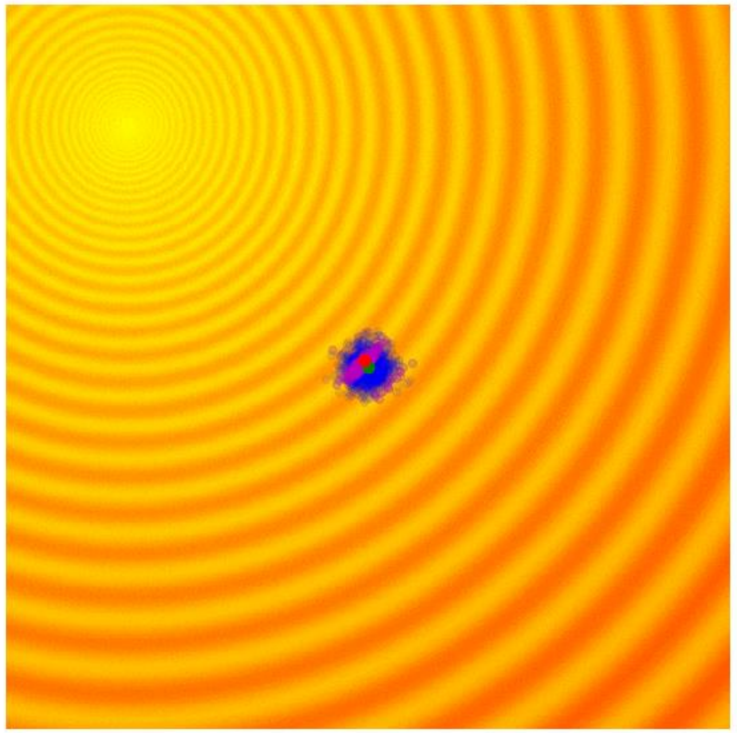
CMA-ES at work

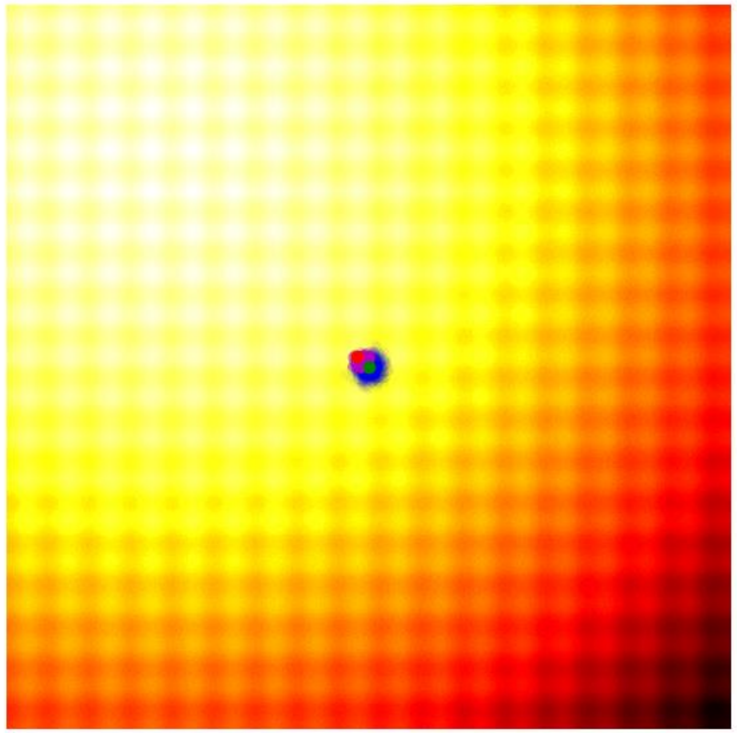


- Full details of **C** and **σ** updates: https://arxiv.org/abs/1604.00772
- Computer code: https://github.com/CMA-ES



CMA-ES adapts direction and spread





- CMA-ES is currently the most powerful evolutionary algorithm for real-value optimization, but for >10K parameters it becomes computationally very expensive
- Open AI ES instead scales to millions of parameters, but can get stuck in local optima

Source: http://blog.otoro.net/2017/10/29/visual-evolution-strategies/

Checkpoints

- Evolutionary Strategies: what do m and I represent?
- Describe the specific choice of algorithm components of Evolutionary Strategies
- Evolutionary Strategies: how does mutation work?
- What are the key differences of OpenAI ES wrt the original ES?
- CMA-ES: how does mutation operate?