Covariance Matrix Adaptation in Python

ME498: Comp. modeling & optimization

Tejaswin Parthasarathy, Mattia Gazzola

February 21, 2019

Matplotlib

Additional packages

- `Seaborn` DEMO
- `bokeh`
- `plotly`
- See Python wiki for more plotting tools

Quick review

GA demo : Results of our GA implementation

How to improve convergence?

- Parameters?
- Strategies? (More mutation, less recombination say...?)
- Tuning is painful

$\Rightarrow$ CMAes (and other algorithms)

Implementation of CMAes

The algorithm

Recap::

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

$$\mathbf{m} \longleftarrow \mathbf{m} + \sigma \langle \mathbf{z} \rangle_w \qquad \langle \mathbf{z} \rangle_w = \sum_{i=1} w_i \mathbf{z}_{i:\lambda}$$

$$\mathbf{p}_c \longleftarrow (1 - c_c)\mathbf{p}_c + \sqrt{1 - (1 - c_c)^2} \sqrt{\frac{1}{\sum_{i=1}^{\mu} w_i^2}} \langle \mathbf{z} \rangle_w$$

$$\mathbf{C} \longleftarrow (1 - c_{cov})\mathbf{C} \; + c_{cov} \frac{1}{\mu_{cov}} \mathbf{p}_c \mathbf{p}_c^T \; + c_{cov} \left(1 - \frac{1}{\mu_{cov}}\right) \mathbf{Z} \qquad \mathbf{Z} = \sum_{i=1}^{\mu} w_i \mathbf{z}_{i:\lambda} \mathbf{z}_{i:\lambda}^T$$

$$\mathbf{p}_\sigma \longleftarrow (1 - c_\sigma)\mathbf{p}_\sigma + \sqrt{1 - (1 - c_\sigma)^2} \sqrt{\frac{1}{\sum_{i=1}^{\mu} w_i^2}} \mathbf{C}^{-\frac{1}{2}} \langle \mathbf{z} \rangle_w \qquad \mu_{cov} = \sqrt{\frac{1}{\sum_{i=1}^{\mu} w_i^2}}$$

$$\sigma \longleftarrow \sigma e^{\left(\frac{1}{d_\sigma}\left(\frac{||p_\sigma||}{E||\mathcal{N}(\mathbf{0}, \mathbf{I})||} - 1\right)\right)}$$

Figure 1: CMAes

Starting CMAes

Problem independent

- Set evolution paths $p_\sigma = \mathbf{0}, p_c = \mathbf{0}$
- Set number of generations $g = 0$
- Set covariance matrix $\mathbf{C} = \mathbf{I}$ (Why?)

Problem dependent

- Distribution mean $m \in \mathbb{R}^n$
- Step size $\sigma \in \mathbb{R}_{>0}$ (Important to set > 0)

Starting CMAes: more on problem dependent parameters

- Optimum presumably be within the initial cube $\boldsymbol{m} \pm 3\sigma\,(1, 1, \cdots, 1)^T$
- $\therefore$ if optima $\in [a, b]^n$ choose $\boldsymbol{m} \in [a, b]^n$ (as a uniformly random vector) and $\sigma = 0.3 * (b - a)$
- Different search intervals Δs_i for different variables can be done using $\mathbf{C}$ as shown below (deferred discussion):

$$\begin{bmatrix} \Delta s_1^2 & 0 & \cdots & 0 \\ 0 & \Delta s_2^2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \Delta s_2^n \end{bmatrix} \tag{1}$$

- Δs_i all must be of similar magnitude (for conditioning). Else, rescale your problem.

First step : Sampling

New population of points, for $k = 1 \cdots \lambda$

- $y_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- $z_k \sim \mathcal{N}(\mathbf{0}, \mathbf{C}) = \mathbf{B}\mathbf{D}y_k$
  - Given $\mathbf{C} = \mathbf{B}\mathbf{D}^2\mathbf{B}^T$
  - Consult [1] for a proof of why $\mathbf{A}\mathcal{N}(\mathbf{0}, \mathbf{I}) = \mathcal{N}(\mathbf{0}, \mathbf{A}\mathbf{A}^T)$
- $x_k = m + \sigma z_k \sim \mathcal{N}(m, \sigma^2\mathbf{C})$

Computing?

- Steps 1 and 3 above using `*/np.multiply` for elementwise multiplication and `+` for elementwise addition

We need a way to sample correlated (across dimensions) populations from numpy

[1] https://math.stackexchange.com/q/2115701

Sampling : Python

How to sample a multivariate normal distribution?

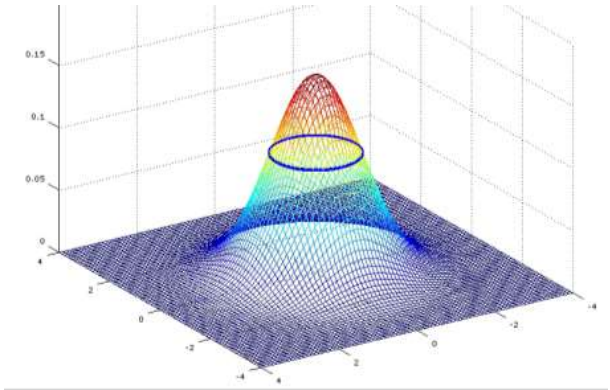
- `np.random.multivariate_normal` DEMO

Caveats?

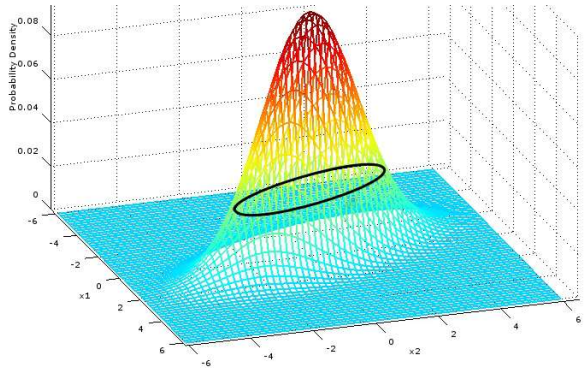
- What is $\mathbf{C}$ / `cov` (in a 2D case) and its meaning?
- `cov` needs to be SP(semi)D. Is it? What about the update step?
- What happens in `numpy` if it is not?

Covariance Matrix [2]

Covariance-Matrix Adaptation (CMA)



How??



$N(0, I)$

$N(0, C)$

Identity Matrix

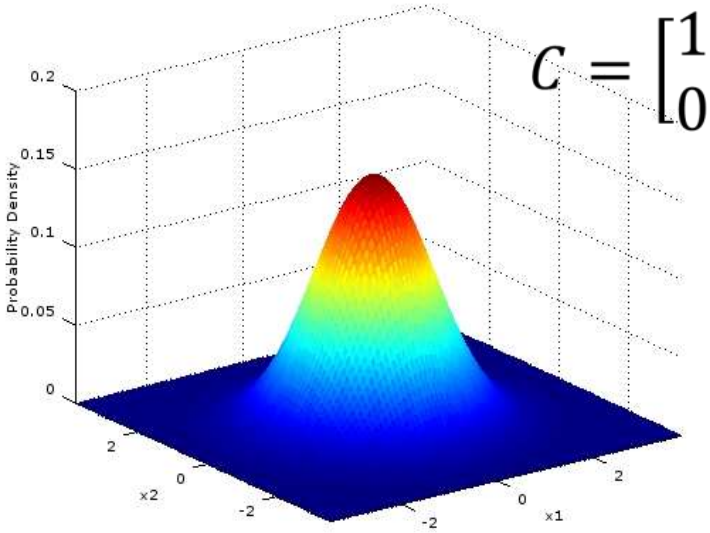
Adapt the **Covariance Matrix**

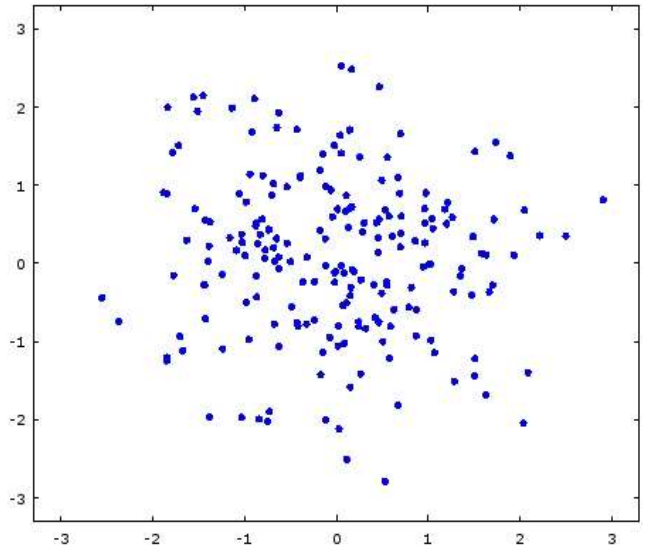
[2] CMAes overview, Slideshare

Covariance Matrix : Example 1

Covariance-Matrix

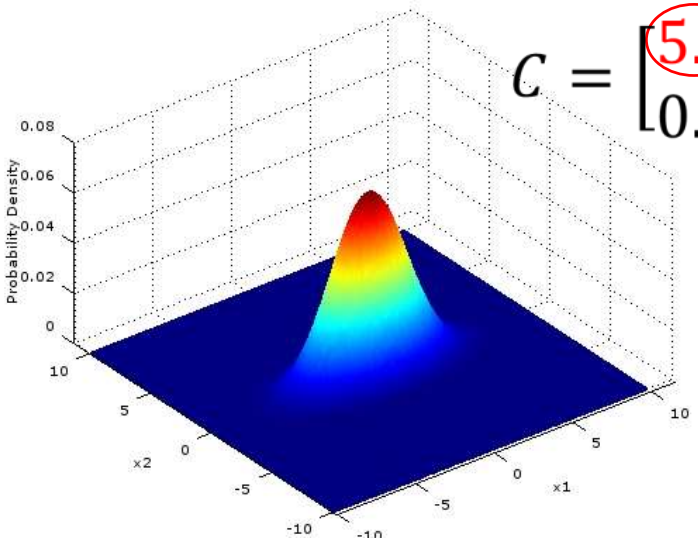
$$C = \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}$$



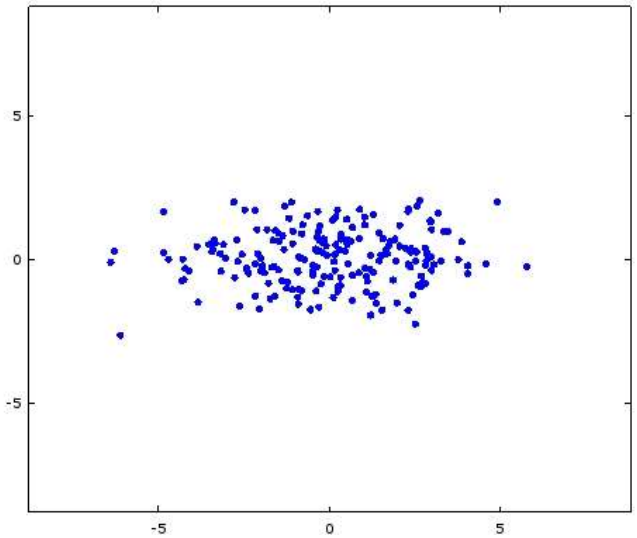


Covariance Matrix : Example 2

Covariance-Matrix



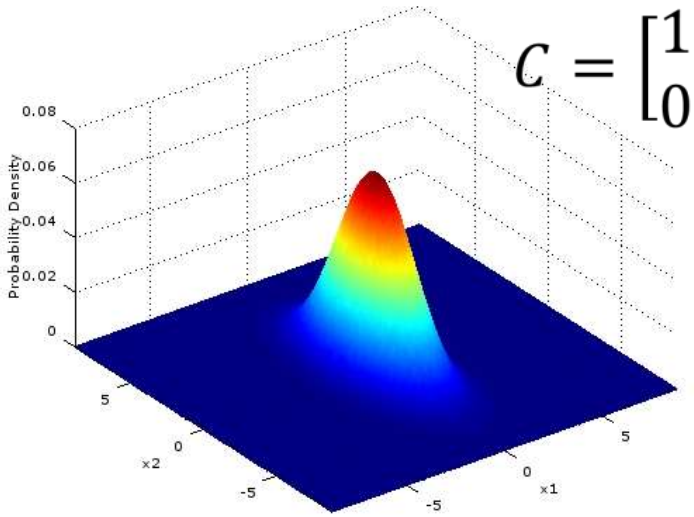
$$C = \begin{bmatrix} 5.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}$$

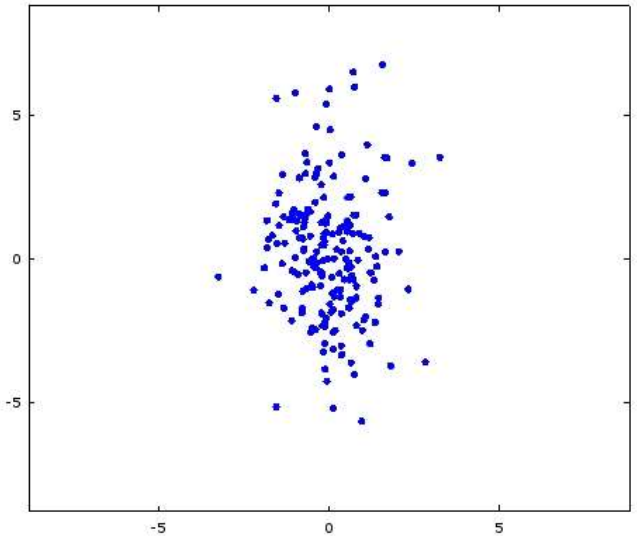


Covariance Matrix : Example 3

Covariance-Matrix

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

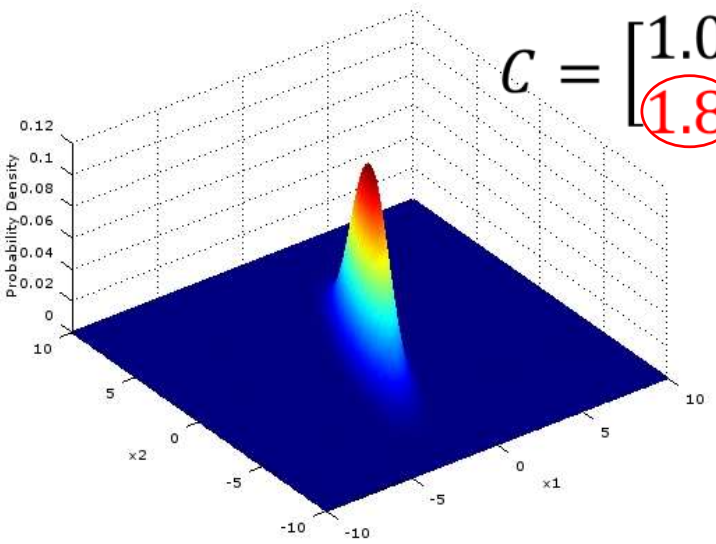


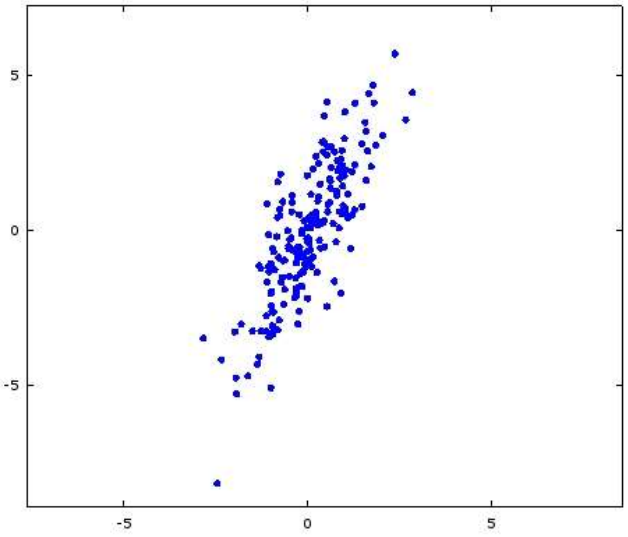


Covariance Matrix : Example 4

Covariance-Matrix

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

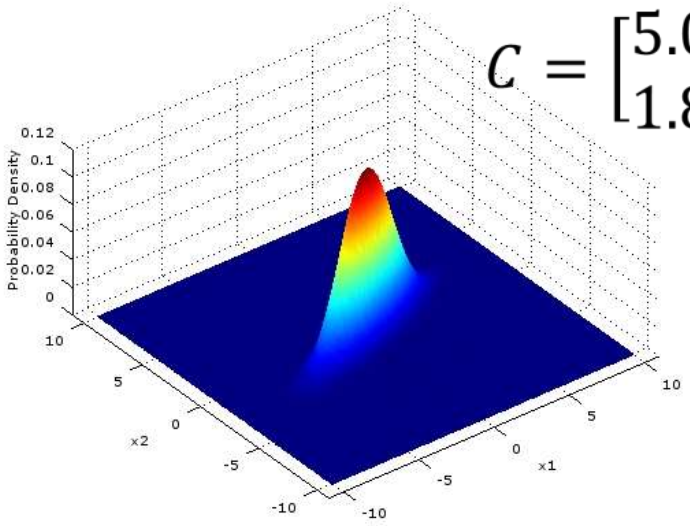


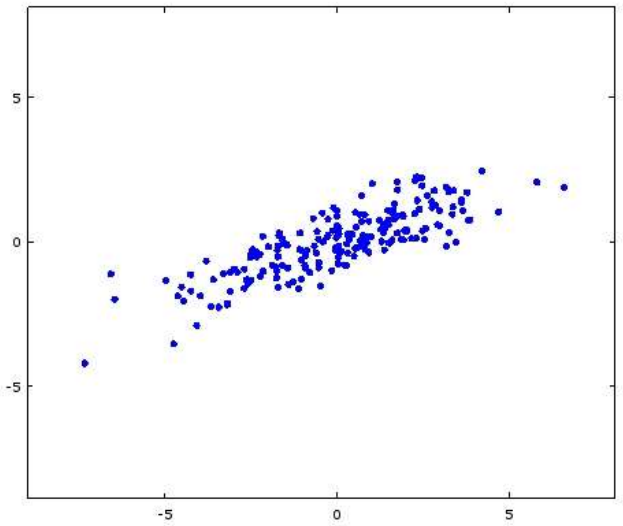


Covariance Matrix : Example 5

Covariance-Matrix

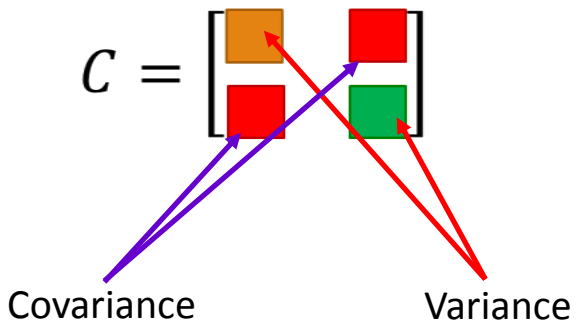
$$C = \begin{bmatrix} 5.0 & 1.8 \\ 1.8 & 1.0 \end{bmatrix}$$





Covariance-Matrix Adaptation (CMA)

- To which direction should the population be directed?



Covariance Matrix : Definition

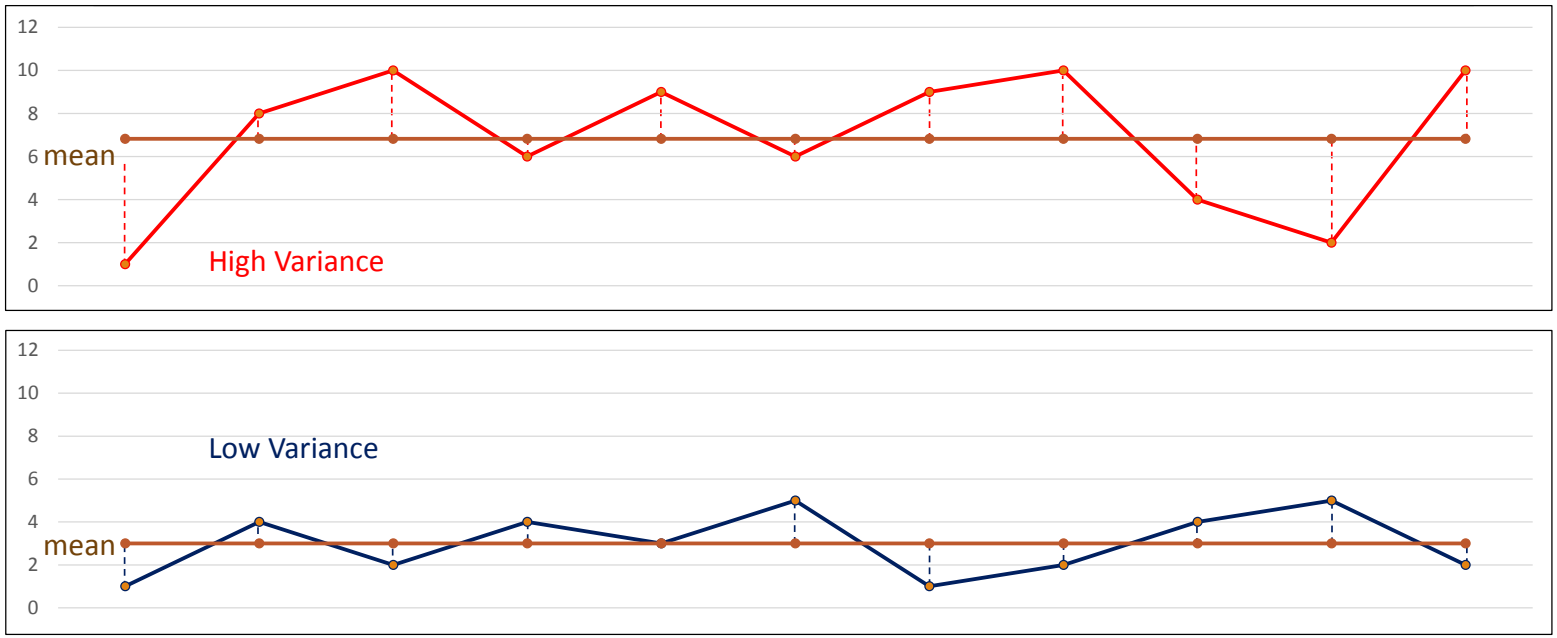
Variance

- Variance is a measure of how far a variable changes away from its mean

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

Covariance Matrix : Example

Variance



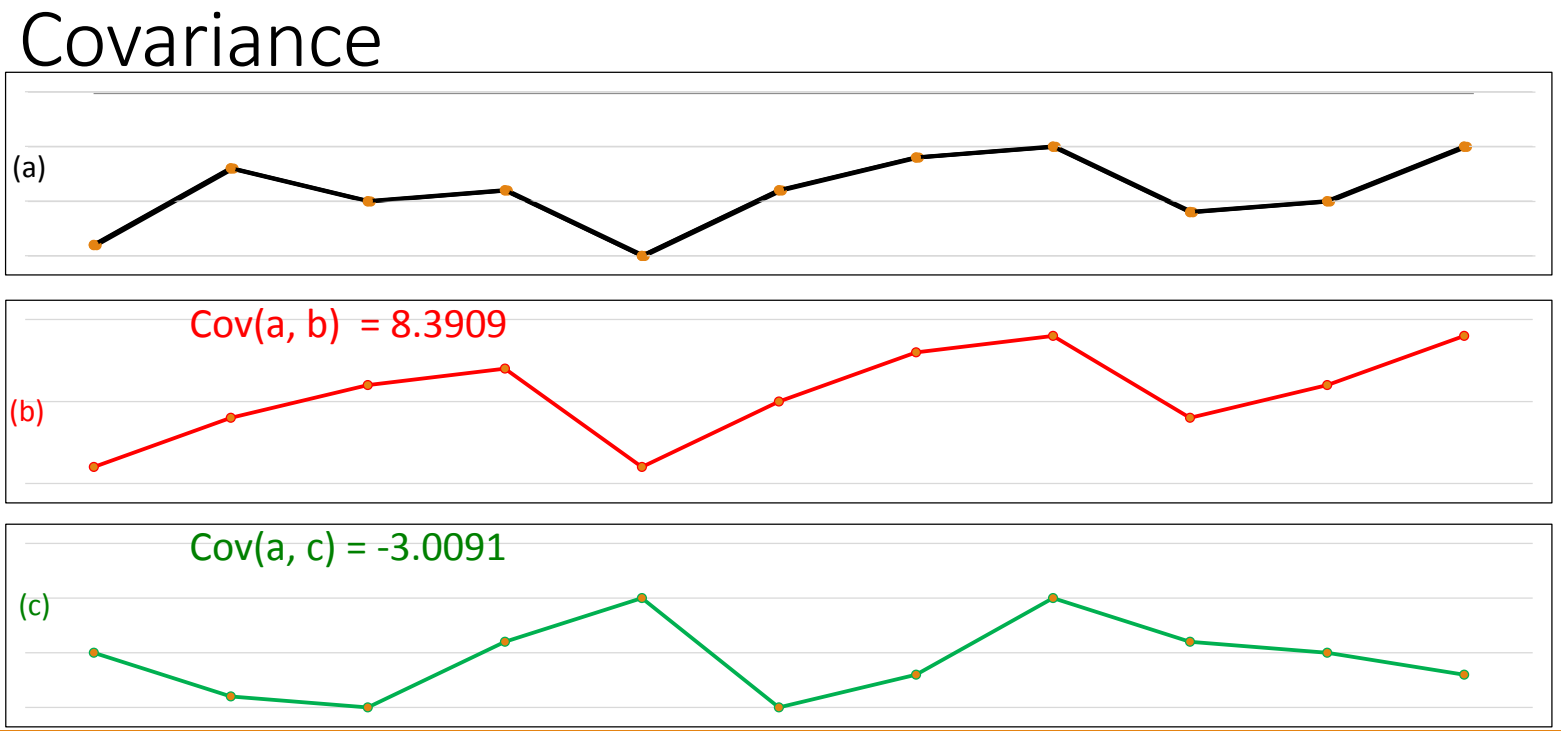
Covariance

- Covariance is a measure of how two variables change together

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

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

Covariance Matrix : Example



Sampling : Python–Answers

- What is $\mathbf{C}$ / `cov` (in a 2D case) and its meaning?
  1. Covariance, how a gene varies with another (across dimensions)
  2. $\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i$
  3. $\sigma_x^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_x)^2$ and $\sigma_{xy} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)$
- `cov` needs to be SPD. Is it? What about the update step?
  1. Symmetric by definition
  2. Symmetric after update too
- What happens in `numpy` if it is not?
  1. `numpy` checks for PD, else throws an exception

Sampling : Idea of $\mathrm{C} \rightarrow$ math

What is CMA-ES doing?

- How does CMA estimate $\mathbf{C}$?
- What about the choice of weights?
- What is CMA doing by adapting $\mathbf{C}$?

Sampling : Idea of $\mathbf{C}$ $\rightarrow$ math–Answers

- How does CMA estimate $\mathbf{C}$?
  1. You can use the new population to get $\mathbf{C}$ too, but information is lost (no information on how the population "evolved", see EMNA from previous slides)
  2. Idea : Use $\mu_x^{(g+1)} = \frac{1}{N_{best}} \sum_{i=1}^{N_{best}} x_i$ rather than $\sigma_x^{2,(g+1)} = \frac{1}{N_{best}} \sum_{i=1}^{N_{best}} (x_i - \mu_x^{(g)})^2$, across N_{best} individuals to estimate covariance between genes (rank μ update)
  3. Exponential weighting, discussed later on
- What about the choice of weights?
  1. Reflect normalization (relates back to the ability of CMA to maintain invariance)
- What is CMA doing by adapting $\mathbf{C}$?
  1. Conducts PCA (eigenvectors), rotated representation $\mathbf{C} = \mathbf{B}\mathbf{D}^2\mathbf{B}^T$, inverse Hessian (second order)

PCA

CMAes performs PCA on the optimization data

PCA?

1. Principal Component Analysis
2. Find directions with
    - High Variance
    - Low Covariance with other components
3. Find dimensions that are “independent” from one another
4. Gives a useful basis (in this case for $\mathbf{C}$)

Sampling : Parameters

Choice of λ?

- Look at the CMA tutorial : The CMA tutorial/CMA tutorial on Arxiv
- Usually $\lambda = \lfloor 4 + 3 \ln n \rfloor$
- And $\mu = \lfloor \lambda / 2 \rfloor$

Second step : Selection

How to select μ best individuals

- $\langle z_k \rangle_w = \sum_{i=1}^{\mu} w_i z_{i:\lambda}$
- Constraint on weights: $\sum_{i=1}^{\mu} w_i = 1,\ w_i > 0\ \forall i = 1 \cdots \mu$ (at least in our version of CMA)

Computing?

- Fitness function evaluation left upto user (including constraints etc.). This determines the μ best individuals.
- The weighted sum can be evaluated using `np.inner()/broadcasting with */np.sum() after *` ...

Selection : Parameters

Choice of w_i?

- Look at the CMA tutorial : The CMA tutorial/CMA tutorial on Arxiv

Third step : Recombination

Recombination to get new m

- $m \leftarrow m + \sigma \langle z \rangle_w$
- No parameters in this step!

Notice!

- σ is the "overall" step size and is a scalar.
- It could also be a matrix. Is this a good idea?
  - What about a diagonal matrix?

Computing?

- Use elementwise addition using `+` operator

Third step : Recombination–Answers

- σ could also be a matrix. Is this a good idea?
  - NO!
  - One dimension depends on another, but not during sampling. This degrades the convergence of the algorithm
- What about a diagonal matrix?
  - NO!
  - $\mathbf{C}$= $\mathbf{BD^2B^T}$ does the job of maintaining scaling, orientation etc. of the elements.

CONCLUSION—Scalar σ is apt.

Fourth step : Step size control

Control for σ and cumulation p_σ

- $p_\sigma \leftarrow (1 - c_\sigma) p_\sigma + \sqrt{c_\sigma (2 - c_\sigma)} \mu_{\text{cov}} \mathbf{C}^{-\frac{1}{2}} \langle z \rangle_w$
- $\sigma \leftarrow \sigma \exp\left(\frac{c_\sigma}{d_\sigma} \left[\frac{\|p_\sigma\|}{\mathsf{E}\|\mathcal{N}(0, \mathbf{I})\|} - 1 \right] \right)$

Computing/Python?

- Notice you need to invert the covariance matrix! How will you do it?
  - Hint: Exploit properties of $\mathbf{C}$!
  - This means you just need `np.linalg.eigh()` for now (there are many other powerful methods for general symmetric matrix inverse)
  - Can reduce $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$ in practice? (See B2. Strategy internal numerical effort in CMA tutorial)

Step size control: Computing/Python

Computing continued

- Extensive use of matvecs (@)
- What about the norm in the σ update?
  - What is a norm?
  - So what norm should we use?
    - The two-norm is widely used (Euclidean distance)
- What's E?
  - What's E $\|\mathcal{N}(0, \mathbf{I})\|$?
    - $\approx \sqrt{n}\left(1 - \frac{1}{4n} + \frac{1}{21n^2}\right)$

Step size control $\rightarrow$ math

What is path update doing?

- Increase probability of reproducing successful solution paths...
- Weighting with exponential decay...
- What about the choice of weights?
  - Makes the expected length independent of the direction
  - "Follows" the random choice of $\boldsymbol{p}_\sigma^{(0)}$

What is σ update doing?

- Decrease/Increase size until path steps are uncorrelated...
- How does the two norm of the path reflect this "un"correlation?
- What about the choice of weights?

Step size : Parameters

Choice of c_σ, d_σ?

- Look at the CMA tutorial : The CMA tutorial/CMA tutorial on Arxiv
- c_σ is learning rate for cumulation usually set to $\approx \frac{4}{n}$
- d_σ is the damping parameter for step size update $\approx 1 + \frac{\mu_{\text{cov}}}{\sqrt{n}}$

Fifth step : Covariance matrix adaptation

Control for $\mathbf{C}$ and cumulation p_c

- $p_c \leftarrow (1 - c_c)p_c + \sqrt{c_c\,(2 - c_c)}\,\mu_{\text{cov}}\langle z\rangle_w$
- $\mathbf{C} \leftarrow (1 - c_{\text{cov}})\mathbf{C} + \frac{c_{\text{cov}}}{\mu_{\text{cov}}} p_c p_c^T + c_{\text{cov}}\left(1 - \frac{1}{\mu_{cov}}\right)\mathbf{Z}$ where $\mathbf{Z} = \sum_{i=1}^{\mu} w_i z_{i:\lambda} z_{i:\lambda}^T$

Computing/Python?

- Usual operations (`*`, `+`)
- For calculating outer products, use `np.outer()`

CMA $\rightarrow$ math

What is cumulation for p_c doing?

- Weighting with exponential decay for prior values
- New information from PCA of steps updated into $\mathbf{C}$ path
- What about the choice of weights?

What is $\mathbf{C}$ update doing?

- Weighting with exponential decay for prior values
- Rank one update using p_c (What's rank?)
  - Why is the update rank one? (One-dimensional information)
  - Why use p_c rather than $\langle z \rangle$?
- Rank μ update
  - As seen earlier, CMA cleverly estimates $\mathbf{C}$ using old step information

CMA: Parameters

Choice of c_c, c_{cov}?

- Look at the CMA tutorial : The CMA tutorial/CMA tutorial on Arxiv
- c_c is learning rate for path cumulation set to $\approx \frac{4}{n}$
- $c_{\text{cov}} \approx \frac{2+\mu_{\text{cov}}^2}{n^2}$

Terminating CMA

Algorithm should be stopped when CPU-time is wasted. Then we can:

1. restart (eventually with increased population size)
2. reconsider encoding and objective function formulation

Problem independent

- `NoEffectAxis` : Stop if adding 0.1 std.dev. vector to any direction of basis $\mathbf{B}$ does not change m
- `NoEffectCoord` : Stop if adding 0.2 std.dev. to any coordinate does not change m
- `ConditionCov`: stop if condition number of covariance matrix exceeds 10^{14}
  - Whats condition number of a matrix?
  - `np.linalg.cond()`, although you can directly check `D`

Terminating CMA contd.

Problem independent

- `EqualFunValues`: stop if the range of the best $f(x)$ of the last $10 + \lceil 30n/\lambda \rceil$ generations is zero.
- `Stagnation`: Track history of the best and the median fitness in each iteration over the last 20% but at least $120 + 30n/\lambda$ and no more than 20000 iterations. Stop, if in both histories the median of the last (most recent) 30% values is not better than the median of the first 30%.
- `TolXUp`: stop if $\sigma \times max(diag(\mathbf{D}))$ increased by more than 10^4. This indicates a far too small initial σ, or divergence.

We note that there are problem dependent diagnostics too!

Boundaries/Constraints in CMA : Best solution strictly inside

- Set fitness (for minimization problem) as

$$f_{\text{fitness}}(x) = f_{\max} + \|x - x_{\text{feasible}}\|$$

1. Notation
   1.1 $f_{\max}$ is larger than worst feasible fitness
   1.2 x_{feasible} is constant,in the middle of feasible region
2. Caveat : Optimal solution not too close to the infeasible region

- Alternatively, resample any infeasible point until it becomes feasible

Boundaries/Constraints in CMA : Repair

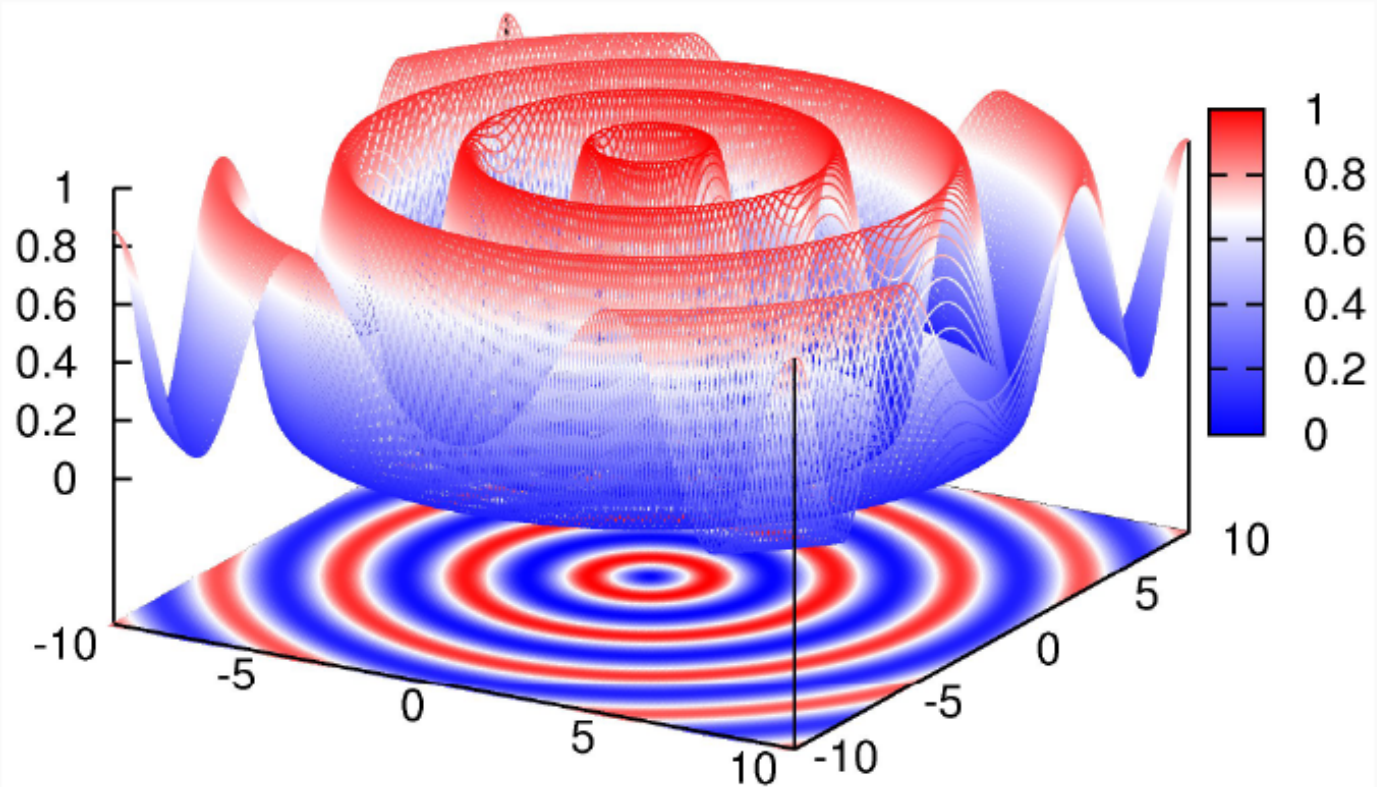
- Simply “repair” infeasible individuals (say when boundary is a box) before update so that they satisfy the constraint
  1. Caveat : Repairs are dangerous
     - Distribution affected by repair, hurting CMA’s convergence
  2. “Re-repair” mechanisms to prevent divergence are also reported
- Alternatively, penalize the repaired solutions

$$f_{\text{fitness}}(\boldsymbol{x}) = f(\boldsymbol{x}_{\text{repaired}}) + \alpha \left\| \boldsymbol{x} - \boldsymbol{x}_{\text{repaired}} \right\|^2$$

Comparing CMA against GA

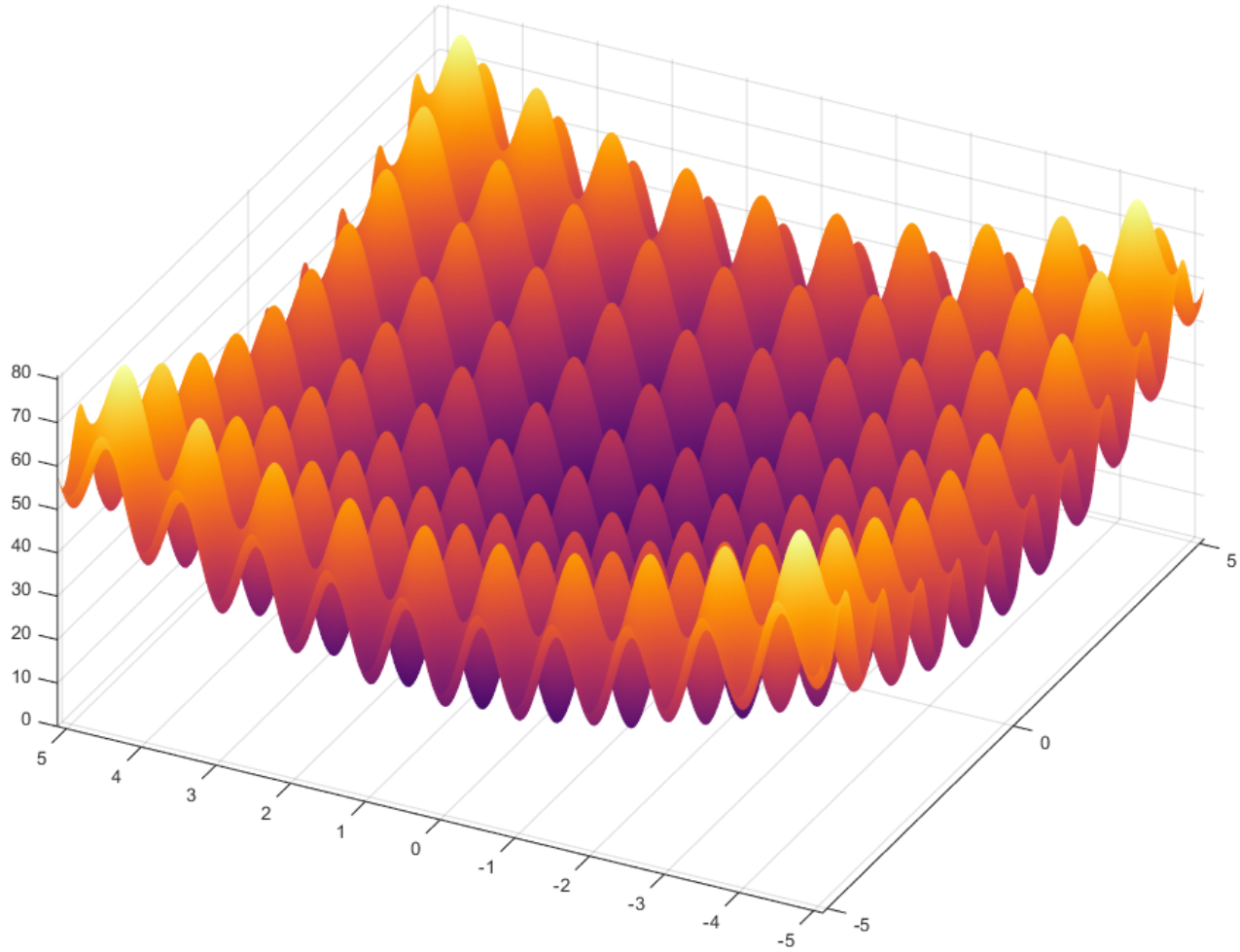
Optimization on smooth functions

- Two dimensional, C^∞ functions $f(\boldsymbol{x}) : \left(\mathbb{R}^2, \mathbb{R}, f, \leq\right)$
- shifted Schaffer function (optima in the middle well)



CMAes vs GA–setup

- shifted Rastrigin function (optima in the middle well)



Comparison between functions[3]

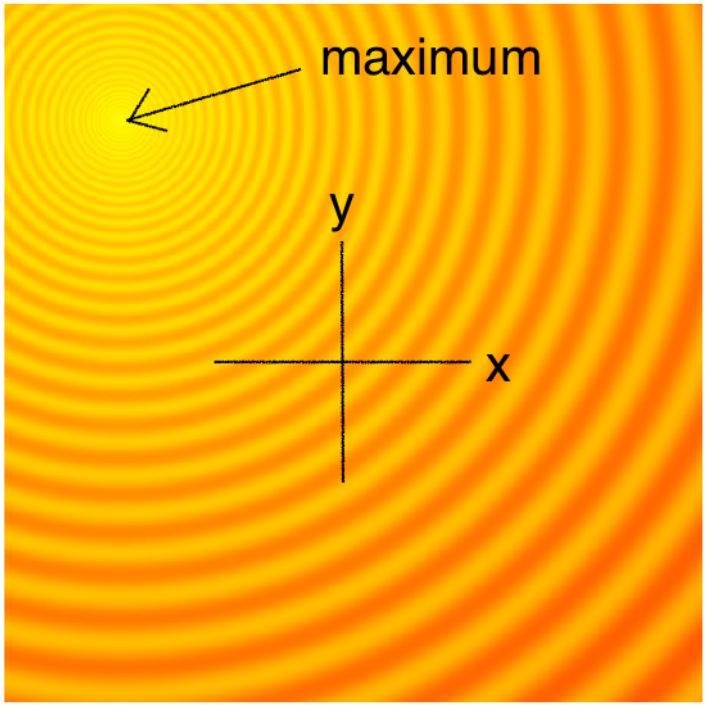


Figure 2: Schaffer–setup

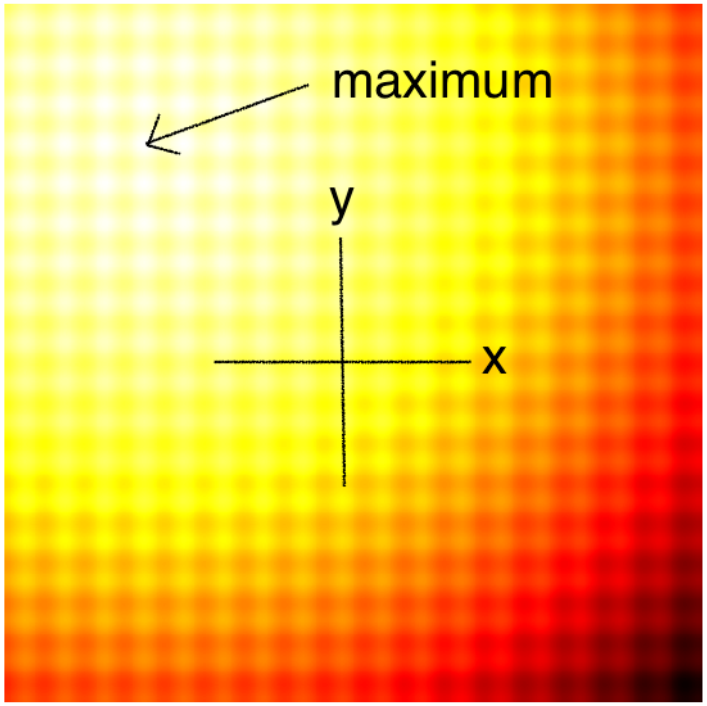


Figure 3: Rastrigin–setup

Lighter region indicates smaller values

[3] Otoro

Simple ES

Scheme

- Sampling : $z_i \sim \mathcal{N}(m, \mathbf{C})$
- Mean-update : $m \leftarrow z_{1:\lambda}$
- Covariance-update : $\mathbf{C} = \begin{bmatrix} \sigma_x^2 & \sigma_x\sigma_y \\ \sigma_x\sigma_y & \sigma_y^2 \end{bmatrix}$ σ_x, σ_y are fixed.
- No other updates (on path etc.)

Legend

- Green : Tracks the mean m.
- Blue : Tracks the sampled solutions at generetation g.
- Red : Tracks the best individual so far.

Results

Simple Evolution strategy from Otoro shown for 20 generations

Simple ES-Observations

Convergence

- What do you expect for general problems?
-

Rate of convergence

- Is this fast/slow convergence?
-

Number of function evaluations?

- High? Low? Not bad?
-

Simple ES-Observations

Convergence

- What do you expect for general problems?
- Will get stuck–lack of diversity, keeps only best population
  (See rastrigin, which temporarily gets stuck)
- Heavy parameter dependence too

Rate of convergence

- Is this fast/slow convergence?
- Slow–no history information

Number of function evaluations?

- High? Low? Not bad?
- Decent–but no promises for real life black-box optimization problems

Simple GA

Scheme

- Environmental selection : Keep only best 10%
- Sampling : Crossover from parents selected above with $p_c = 1$
- Crossover : Select two parents, obtain x or y from either parent with 0.5 probability (two coin tosses)
- Mutation : Introduce Gaussian noise with fixed σ
- No other updates (on path etc.)

Legend

- Green : Tracks the elites from prior generation g.
- Blue : Offsprings from candidate solutions.
- Red : Tracks the best individual so far.

Simple GA-Observations

Convergence

- What do you expect for general problems?
-

Rate of convergence

- Is this fast/slow convergence?
-

Number of function evaluations?

- High? Low? Not bad?
-

Simple GA-Observations

Convergence

- What do you expect for general problems?
- Will get stuck–lack of diversity, keeps only elitist population
- Heavy parameter dependence
- Tracks modality well (for both Schaffer and Rastrigin)

Rate of convergence

- Is this fast/slow convergence?
- Slower than simple ES

Number of function evaluations?

- High? Low? Not bad?
- High

CMAes-Observations

Can you spot the updates?

- m update (fairly obvious)
- Step size update
  - Path update
- Covariance matrix upfate
  - Rank μ updates
  - Rank 1 update (Path update)

CMAes-Observations

Convergence

- What do you expect for general problems?
- Good for problems of “moderate” dimensions

Rate of convergence

- Is this fast/slow convergence?
- Fast (Approximately brackets minima in $\mathcal{O}(n)$ functional evaluations)

Number of function evaluations?

- High? Low? Not bad?
- Low (same as above)

CMAes-Some interesting videos

- Mario
  https://www.youtube.com/watch?v=0iipyd7Gi70
- Rastrigin :
  https://www.youtube.com/watch?v=aP31Q7o2UGU
- Biped:
  https://www.youtube.com/watch?v=lOaWvOA9cb4
- Robot Invivo:
  https://www.youtube.com/watch?v=trR2Gc1tLzg
- Robot invitro:
  https://www.youtube.com/watch?v=fjTd06L-9bQ
- Knifefish :
  https://www.youtube.com/watch?v=3XjgZbs0t2g
- https:
  //blog.openai.com/evolution-strategies/