# Advanced Optimization
## Lecture 7: Stochastic Gradient Descent and Benchmarking

November 24, 2021
CentraleSupélec / ESSEC Business School
dimo.brockhoff@inria.fr

Dimo Brockhoff
Inria Saclay – Ile-de-France

# Course Overview

| | | Topic |
|---|---|---|
| Wed, 13.10.2021 | PM | Introduction, examples of problems, problem types |
| Wed, 20.10.2021 | PM | Continuous (unconstrained) optimization: convexity, gradients, Hessian, … [technical test Evalmee] |
| Wed, 27.10.2021 | PM | Continous optimization II: [1st mini-exam]<br>Constrained optimization: Lagrangian, optimality conditions |
| Wed, 03.11.2021 | PM | gradient descent, Newton direction, quasi-Newton (BFGS)<br>Linear programming: duality, maxflow/mincut, simplex algo |
| Wed, 10.11.2021 | PM | derivative-free algorithms: Nelder-Mead and CMA-ES |
| Wed, 17.11.2021 | PM | CMA-ES, Part II, Stochastic Gradient Descent, Bayesian optimization |
| Wed, 24.11.2021 | PM | Benchmarking solvers: runtime distributions, the COCO platform, recommendations **[2nd mini-exam]** |
| Wed, 01.12.2021 | PM | Discrete optimization: branch and bound, branch and cut |
| **Fri, 3.12.2021** | 23:59 | Deadline open source project (PDF sent by email) |
| Wed, 15.12.2021 | PM | Exam |

recap

# Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

# The CMA-ES

Input: $\boldsymbol{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, $\lambda$
Initialize: $\mathbf{C} = \mathbf{I}$, and $\boldsymbol{p}_c = \mathbf{0}$, $\boldsymbol{p}_\sigma = \mathbf{0}$,
Set: $c_c \approx 4/n$, $c_\sigma \approx 4/n$, $c_1 \approx 2/n^2$, $c_\mu \approx \mu_w/n^2$, $c_1 + c_\mu \leq 1$, $d_\sigma \approx 1 + \sqrt{\frac{\mu_w}{n}}$,
and $w_{i=1\ldots\lambda}$ such that $\mu_w = \frac{1}{\sum_{i=1}^{\mu} {w_i}^2} \approx 0.3\,\lambda$

While not terminate

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

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

$$\boldsymbol{p}_c \leftarrow (1 - c_c)\,\boldsymbol{p}_c + \mathbb{1}_{\{\|\boldsymbol{p}_\sigma\| < 1.5\sqrt{n}\}} \sqrt{1 - (1 - c_c)^2}\sqrt{\mu_w}\,\boldsymbol{y}_w \qquad \text{cumulation for } \mathbf{C}$$

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

$$\mathbf{C} \leftarrow (1 - c_1 - c_\mu)\,\mathbf{C} + c_1\,\boldsymbol{p}_c\boldsymbol{p}_c^{\mathrm{T}} + c_\mu \sum_{i=1}^{\mu} w_i\,\boldsymbol{y}_{i:\lambda}\boldsymbol{y}_{i:\lambda}^{\mathrm{T}} \qquad \text{update } \mathbf{C}$$

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

Not covered on this slide: termination, restarts, useful output, boundaries and encoding

# CMA-ES: Stochastic Search Template

**A stochastic blackbox search template to minimize $f: \mathbb{R}^n \to \mathbb{R}$**

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

While happy do:

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

For CMA-ES and evolution strategies in general:

sample distributions = multivariate Gaussian distributions

## The CMA-ES

Input: $\boldsymbol{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, $\lambda$
Initialize: $\mathbf{C} = \mathbf{I}$, and $\boldsymbol{p}_\mathrm{c} = \mathbf{0}$, $\boldsymbol{p}_\sigma = \mathbf{0}$,
Set: $c_\mathrm{c} \approx 4/n$, $c_\sigma \approx 4/n$, $c_1 \approx 2/n^2$, $c_\mu \approx \mu_w/n^2$, $c_1 + c_\mu \leq 1$, $d_\sigma \approx 1 + \sqrt{\frac{\mu_w}{n}}$,
and $w_{i=1\ldots\lambda}$ such that $\mu_w = \frac{1}{\sum_{i=1}^{\mu} {w_i}^2} \approx 0.3\,\lambda$

While not terminate

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

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

$$\boldsymbol{p}_\mathrm{c} \leftarrow (1 - c_\mathrm{c})\,\boldsymbol{p}_\mathrm{c} + \mathbb{1}_{\{\|\boldsymbol{p}_\sigma\| < 1.5\sqrt{n}\}} \sqrt{1 - (1 - c_\mathrm{c})^2}\sqrt{\mu_w}\,\boldsymbol{y}_w \qquad \text{cumulation for } \mathbf{C}$$

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

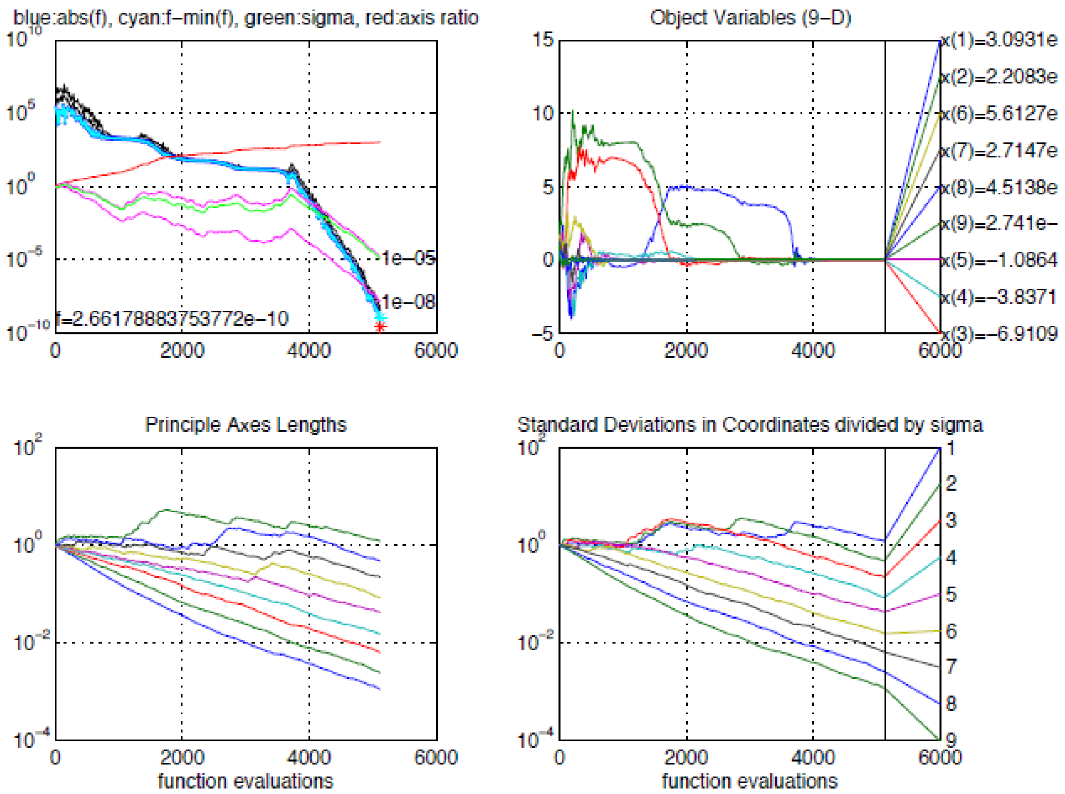
$$\mathbf{C} \leftarrow (1 - c_1 - c_\mu)\,\mathbf{C} + c_1\,\boldsymbol{p}_\mathrm{c}\boldsymbol{p}_\mathrm{c}^\mathrm{T} + c_\mu \sum_{i=1}^{\mu} w_i\,\boldsymbol{y}_{i:\lambda}\boldsymbol{y}_{i:\lambda}^\mathrm{T} \qquad \text{update } \mathbf{C}$$

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

Not covered on this slide: termination, restarts, useful output, boundaries and encoding

# Experimentum Crucis (1)

$f$ convex quadratic, separable

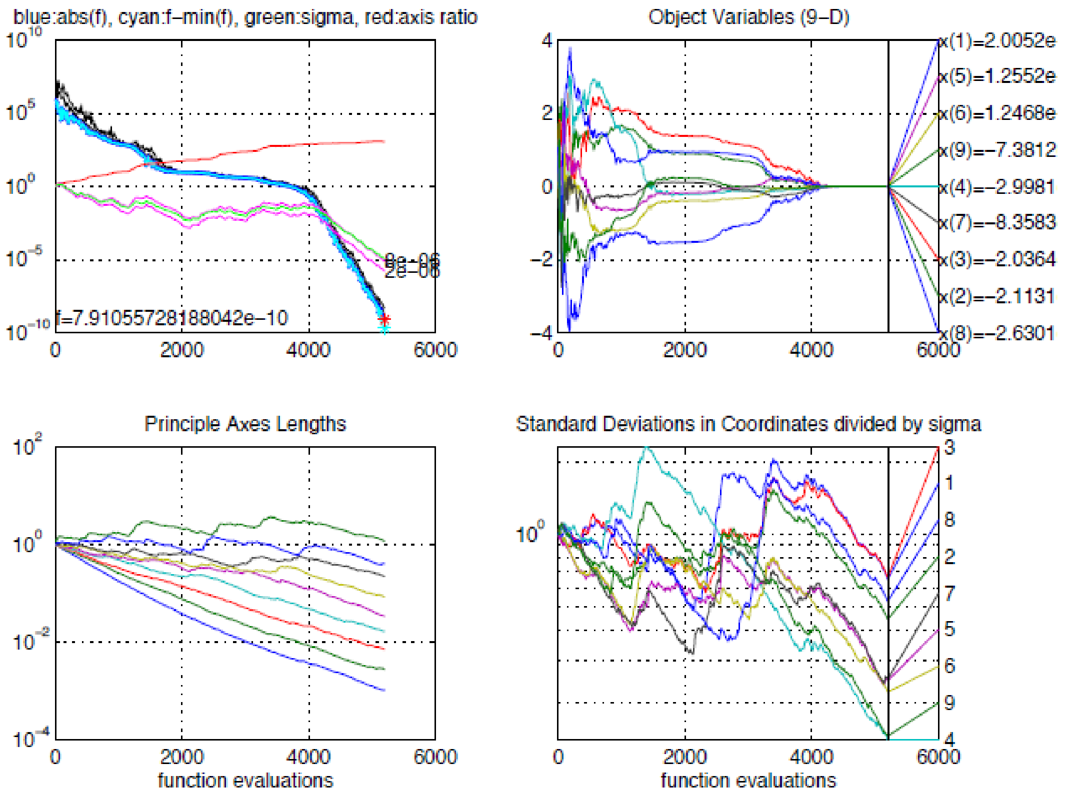



$$f(\boldsymbol{x}) = \sum_{i=1}^{n} 10^{\alpha \frac{i-1}{n-1}} x_i^2, \alpha = 6$$

from [Hansen, p. 92]

# Experimentum Crucis (2)

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




$\mathbf{C} \propto \boldsymbol{H}^{-1}$ for all $g, \mathbf{H}$

$$f(\boldsymbol{x}) = g\left(\boldsymbol{x}^{\mathrm{T}}\mathbf{H}\boldsymbol{x}\right),\ g : \mathbb{R} \to \mathbb{R}$$ stricly increasing

from [Hansen, p. 93]

# showcase cma in python

addendum:

# Stochastic Gradient Descent

# Reminder Gradient Descent

## General principle

1. choose an initial point $\boldsymbol{x}_0$, set $t = 0$
2. while not happy
   - choose the negative gradient as descent direction: $-\nabla f$
   - line search:
     - choose a step size $\sigma_t > 0$
     - set $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \sigma_t \nabla f(\boldsymbol{x}_t)$
   - set $t = t + 1$

# Disadvantage of Basic Gradient Descent

When optimizing weights of a neural net in deep learning:

- we optimize the loss function $\sum_{i=1}^{m}(y_i - \widehat{y_i})$

$y_i$: from training data, $\widehat{y_i}$: neural network output, depending on weight vector $\boldsymbol{w}$

- not only $\boldsymbol{w}$ is of high dimension, but also $m$ is large
- consequence: computing gradient $\nabla f(\boldsymbol{w})$ is costly
- why? → Have to compute many partial derivatives

**Idea:**

- Compute only an approximation of $\nabla f(\boldsymbol{w})$ with respect to *some* data
- More generally applicable when $f(x) = \sum_{i=1}^{m} f_i(x)$

# Stochastic Gradient Descent (SGD)

**Idea:**

Compute only an approximation of $\nabla f(\boldsymbol{x})$ with respect to *some subfunctions* of $f(x) = \sum_{i=1}^{m} f_i(\boldsymbol{x})$

Instead of $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \sigma_t \nabla f(\boldsymbol{x}_t)$, we update the current iterate only with respect to one (random) subfunction $f_i$:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \sigma_t \nabla f_i(\boldsymbol{x}_t)$$

# Batches, Mini-Batches and Epochs

- To make sure, all subfunctions are used equally, we typically choose a (random) permutation $\pi$ and use each subfunction once in this order before to redo this step
- Such step is called an *epoch*

- Using single subfunctions is fast, but also highly stochastic

consequence: practical convergence is slower

- Hence, we shall use batches of subfunctions at a time:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \sigma_t \nabla f_{\mathcal{B}_j}(\boldsymbol{x}_t)$$

with $f_{\mathcal{B}_j}(\boldsymbol{x}_t) = \sum_{i \in \mathcal{B}_j} f_i(\boldsymbol{x})$ and $\mathcal{B}_j \subset \{1, \dots, m\}$ the corresponding indices in the $j$th batch

# SGD vs. Adam

## Stochastic Gradient Descent

Typically applied with a constant step size in the context of Deep Learning

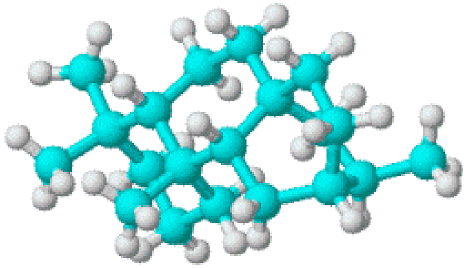
## Adam: Adaptive Moment Adaptation

- decaying average of past gradients $g_t(x) = \nabla f_{\mathcal{B}_j}(x_t)$, also called "momentum":
$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t(x_t)$$
- decaying average of past squared gradients
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2(x_t)$$
- to decrease bias towards 0, we use $\widehat{m_t} = \frac{m_t}{1-\beta_1^t}$ and $\widehat{v_t} = \frac{v_t}{1-\beta_2^t}$
- and then $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \frac{\sigma_t}{\sqrt{\widehat{v_t}}+\epsilon}\widehat{m_t}$

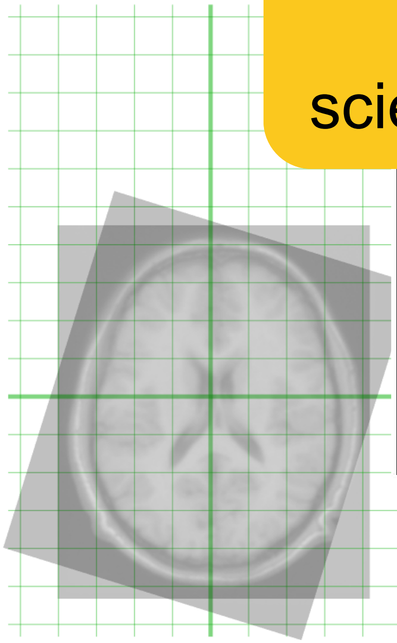
# Mini-Exam #2

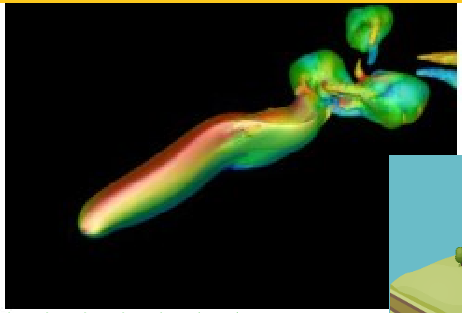
# Benchmarking Optimization Algorithms

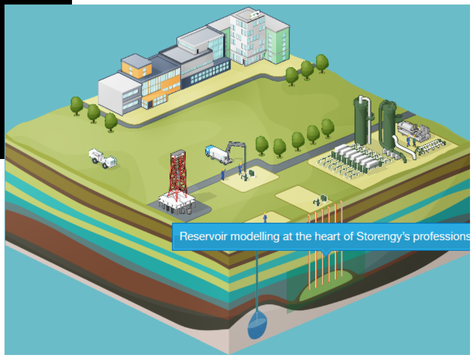
or: critical performance assessment



challenging optimization problems
appear in many
scientific, technological and industrial domains








# Practical (Numerical) Blackbox Optimization

Given:


*derivatives not available or not useful*

Not clear:

which of the many algorithms should I use on my problem?

# Need: Benchmarking

- understanding of algorithms
- algorithm selection
- putting algorithms to a standardized test
    - simplify judgement
    - simplify comparison
    - regression test under algorithm changes

Kind of everybody has to do it (and it is tedious):

- choosing (and implementing) problems, performance measures, visualization, stat. tests, ...
- running a set of algorithms

**How would you compare algorithms?**

assumptions:

- continuous search space $\mathbb{R}^n$
- blackbox scenario w/o constraints
- two algorithms

**a) Define a concrete experimental setup**

- What to do if I want to compare algorithms A and B?
- Which experiment parameters you have to decide on?

**b) What would you display to compare the performance?**