Optimization for Machine Learning

Lecture 2: Continuous Optimization I

October 18, 2021

TC2 - Optimisation

Université Paris-Saclay

Dimo Brockhoff

Inria Saclay – Ile-de-France

Course Overview

Date		Topic
Thu, 4.11.2021	DB	Introduction
Thu, 11.11.2021		no lecture
Thu, 18.11.2021	AA	Continuous Optimization I: differentiability, gradients, convexity, optimality conditions
Thu, 25.11.2021	AA	Continuous Optimization II: constrained optimization, gradient-based algorithms, stochastic gradient [written test / « contrôle continue »]
Thu, 2.12.2021	AA	Continuous Optimization III: stochastic algorithms, derivative-free optimization
Thu, 9.12.2021	DB	Discrete Optimization: greedy algorithms, branch&bound, dynamic programming
Thu 16.12.2021	DB	Written exam
		! Starting from the 18th: 13h15 till 16h00

Details on Continuous Optimization Lectures

Introduction to Continuous Optimization

- examples (from ML / black-box problems)
- typical difficulties in optimization

Mathematical Tools to Characterize Optima

- reminders about differentiability, gradient, Hessian matrix
- unconstraint optimization
  - first and second order conditions
  - convexity
- constraint optimization

Gradient-based Algorithms

- stochastic gradient
- quasi-Newton method (BFGS)

Learning in Optimization / Stochastic Optimization

- CMA-ES (adaptive algorithms / Information Geometry)
- PhD thesis possible on this topic

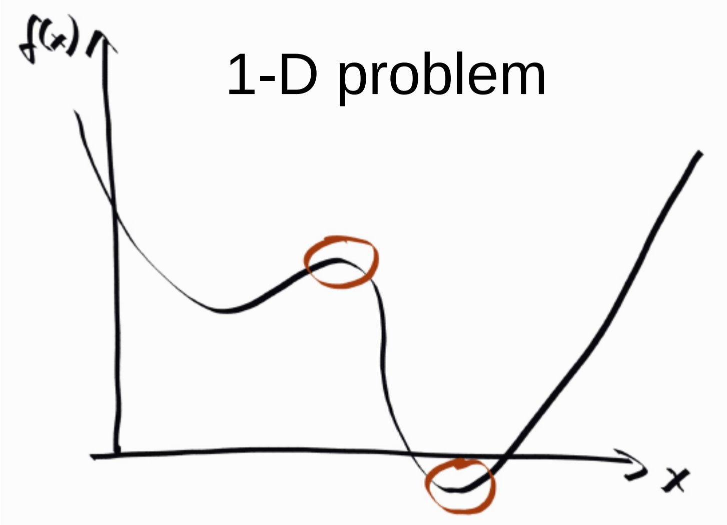
method strongly related to ML / new promising research area
interesting open questions

Continuous Optimization

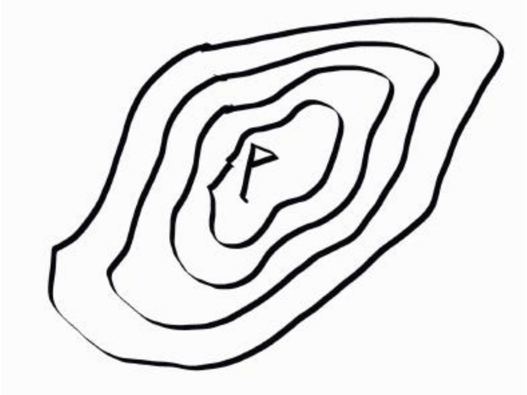
- Optimize f: $\begin{cases} \Omega \subset \mathbb{R}^n \to \mathbb{R} \\ x = (\underset{\nwarrow \in \mathbb{R}}{x_1}, \dots, x_n) \to f(x_1, \dots, x_n) \end{cases}$

unconstrained optimization

- Search space is continuous, i.e. composed of real vectors $x \in \mathbb{R}^n$
- $n = \begin{cases} \text{dimension of the problem} \\ \text{dimension of the search space } \mathbb{R}^n \text{ (as vector space)} \end{cases}$


1-D problem

2-D level sets

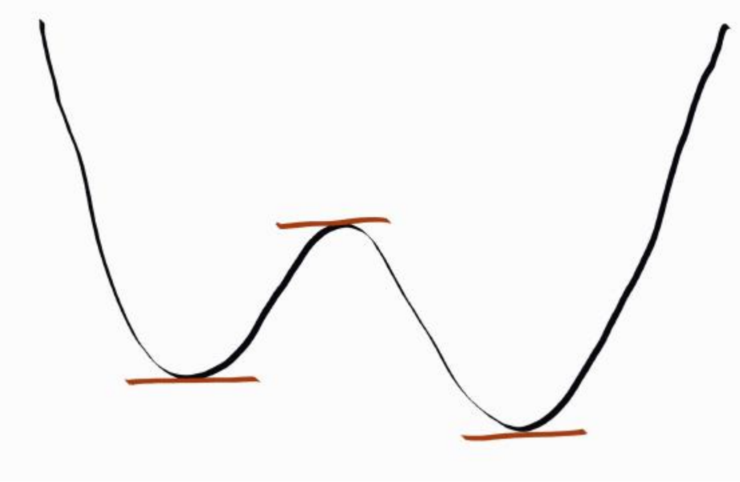


Mathematical Characterization of Optima

Objective: Derive general characterization of optima

Example: if $f: \mathbb{R} \to \mathbb{R}$ differentiable,

$f'(x) = 0$ at optimal points



- generalization to $f: \mathbb{R}^n \to \mathbb{R}$?
- generalization to constrained problems?

Reminder: Geometrical Interpretation of Gradient

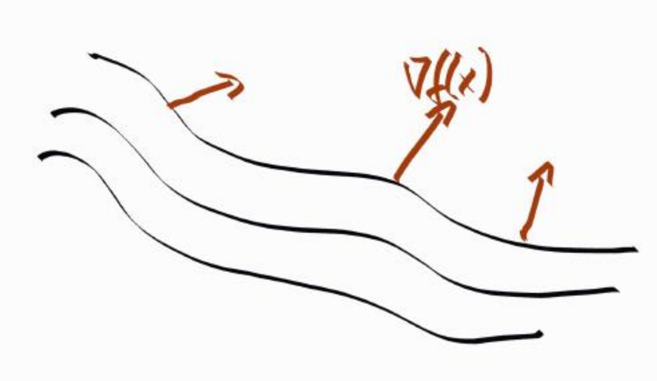
Exercise:

Let $L_c = \{\boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) = c\}$ be again a level set of a function $f(\boldsymbol{x})$.
Let $\boldsymbol{x}_0 \in L_c \neq \emptyset$.

Compute the level sets for $f_1(\boldsymbol{x}) = \boldsymbol{a}^T\boldsymbol{x}$ and $f_2(\boldsymbol{x}) = ||\boldsymbol{x}||^2$ and the gradient in a chosen point x_0 and observe that $\nabla f(\boldsymbol{x_0})$ is ***orthogonal*** to the level set in x_0.

Again: if this seems too difficult, do it for two variables (and a concrete $\boldsymbol{a} \in \mathbb{R}^2$) and draw the level sets and the gradients.

More generally, the gradient of a differentiable function is orthogonal to its level sets.



Differentiability in $\mathbb{R}^n$

Taylor Formula – Order One

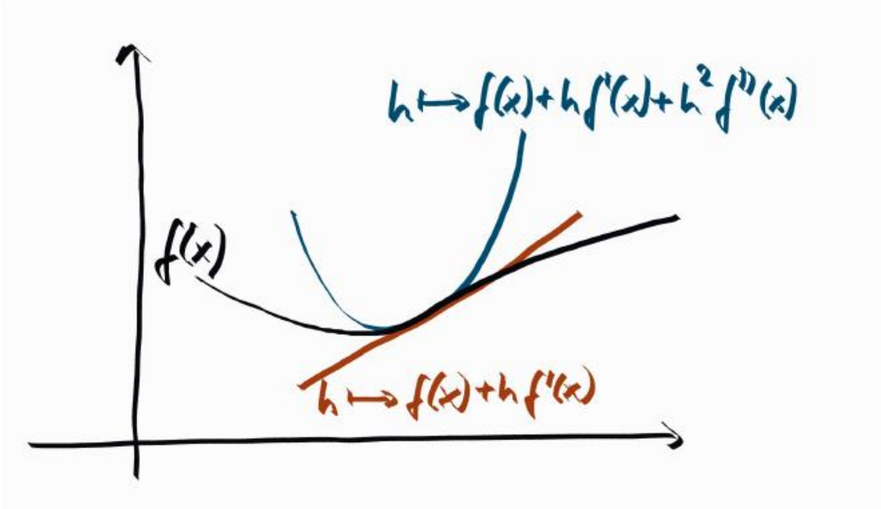
$$f(\boldsymbol{x}+\boldsymbol{h}) = f(\boldsymbol{x}) + \left(\nabla f(\boldsymbol{x})\right)^T \boldsymbol{h} + o(||\boldsymbol{h}||)$$

Reminder: Second Order Derivability in 1D

- Let $f: \mathbb{R} \to \mathbb{R}$ be a differentiable function and let $f': x \to f'(x)$ be its derivative.
- If f' is differentiable in x, then we denote its derivative as $f''(x)$
- $f''(x)$ is called the *second order derivative* of f.

Taylor Formula: Second Order Derivative

- If $f: \mathbb{R} \to \mathbb{R}$ is two times differentiable then
$$f(x+h) = f(x) + f'(x)h + f''(x)h^2 + o(||h||^2)$$
i.e. for h small enough, $h \to f(x) + hf'(x) + h^2 f''(x)$ approximates $h \to f(x+h)$
- $h \to f(x) + hf'(x) + h^2 f''(x)$ is a quadratic approximation (or order 2) of f in a neighborhood of x


$h \mapsto f(x) + hf'(x) + h^2 f''(x)$
$f(x)$
$h \mapsto f(x) + hf'(x)$

- The second derivative of $f: \mathbb{R} \to \mathbb{R}$ generalizes naturally to larger dimension.

Hessian Matrix

In $(\mathbb{R}^n, \langle x, y \rangle = x^T y)$, $\nabla^2 f(x)$ is represented by a symmetric matrix called the Hessian matrix. It can be computed as

$$\nabla^2(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Exercise on Hessian Matrix

Exercise:

Let $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T A\,\boldsymbol{x}$, $\boldsymbol{x} \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times n}$.

Compute the Hessian matrix of f.

If it is too complex, consider f: $\begin{cases} \mathbb{R}^2 \to \mathbb{R} \\ \boldsymbol{x} \to \frac{1}{2}\boldsymbol{x}^T A\,\boldsymbol{x} \end{cases}$ with $A = \begin{pmatrix} 9 & 0 \\ 0 & 1 \end{pmatrix}$

Second Order Differentiability in $\mathbb{R}^n$

Taylor Formula – Order Two

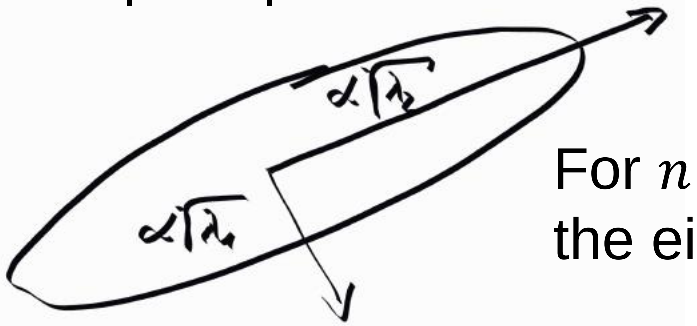
$$f(\boldsymbol{x}+\boldsymbol{h}) = f(\boldsymbol{x}) + \left(\nabla f(\boldsymbol{x})\right)^T \boldsymbol{h} + \frac{1}{2}\boldsymbol{h}^T\left(\nabla^2 f(\boldsymbol{x})\right) \boldsymbol{h} + o(||\boldsymbol{h}||^2)$$

Back to Ill-Conditioned Problems

We have seen that for a convex quadratic function

$f(x) = \frac{1}{2}(x - x_0)^T A(x - x_0) + b$ of $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \times n}$, A SPD, $b \in \mathbb{R}^n$:

1) The level sets are ellipsoids. The eigenvalues of A determine the lengths of the principle axes of the ellipsoid.



For $n = 2$, let λ_1, λ_2 be the eigenvalues of A.

2) The Hessian matrix of f equals to A.

Ill-conditioned convex quadratic problems are problems with large ratio between largest and smallest eigenvalue of A which means large ratio between longest and shortest axis of ellipsoid.

This corresponds to having an ill-conditioned Hessian matrix.

Gradient Direction Vs. Newton Direction

Gradient direction: $\nabla f(\boldsymbol{x})$

Newton direction: $\left(H(\boldsymbol{x})\right)^{-1} \cdot \nabla f(\boldsymbol{x})$

with $H(\boldsymbol{x}) = \nabla^2 f(\boldsymbol{x})$ being the Hessian at $\boldsymbol{x}$

Exercise:

Let again $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T A\, \boldsymbol{x}$, $\boldsymbol{x} \in \mathbb{R}^2$, $A = \begin{pmatrix} 9 & 0 \\ 0 & 1 \end{pmatrix} \in \mathbb{R}^{2\times 2}$.

Plot the gradient and Newton direction of f in a point $x \in \mathbb{R}^n$ of your choice (which should not be on a coordinate axis) into the same plot with the level sets, we created before.

Gradient Direction Vs. Newton Direction

Gradient direction: $\nabla f(\boldsymbol{x})$

Newton direction: $-\left(H(\boldsymbol{x})\right)^{-1} \cdot \nabla f(\boldsymbol{x})$

with $H(\boldsymbol{x}) = \nabla^2 f(\boldsymbol{x})$ being the Hessian at $\boldsymbol{x}$

Exercise:

Let again $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T A\,\boldsymbol{x}$, $\boldsymbol{x} \in \mathbb{R}^2$, $A = \begin{pmatrix} 9 & 0 \\ 0 & 1 \end{pmatrix} \in \mathbb{R}^{2\times 2}$.

Plot the gradient and Newton direction of f in a point $x \in \mathbb{R}^n$ of your choice (which should not be on a coordinate axis) into the same plot with the level sets, we created before.

- remind level sets: axis-parallel ellipsoids, axis-ratio=3
- remind gradient: $A\boldsymbol{x}$
- remind Hessian: A

Optimality Conditions
for Unconstrained Problems

Optimality Conditions: First Order Necessary Cond.

For 1-dimensional optimization problems $\boldsymbol{f}$: $\mathbb{R} \to \mathbb{R}$

Assume f is differentiable

- $\boldsymbol{x}^*$ is a local optimum $\Longrightarrow f'(\boldsymbol{x}^*) = 0$

  not a sufficient condition: consider $f(\boldsymbol{x}) = \boldsymbol{x}^3$

  proof via Taylor formula: $f(\boldsymbol{x}^* + \boldsymbol{h}) = f(\boldsymbol{x}^*) + f'(\boldsymbol{x}^*)\boldsymbol{h} + o(||\boldsymbol{h}||)$

- points $\boldsymbol{y}$ such that $f'(\boldsymbol{y}) = 0$ are called critical or stationary points

Generalization to $\boldsymbol{n}$-dimensional functions

If $f: U \subset \mathbb{R}^n \longmapsto \mathbb{R}$ is differentiable

- necessary condition: If $\boldsymbol{x}^*$ is a local optimum of f, then $\nabla f(\boldsymbol{x}^*) = 0$

  proof via Taylor formula

Second Order Necessary and Sufficient Opt. Cond.

If f is twice continuously differentiable

- Necessary condition: if $\boldsymbol{x}^*$ is a local minimum, then $\nabla f(\boldsymbol{x}^*) = 0$ and $\nabla^2 f(\boldsymbol{x}^*)$ is positive semi-definite

proof via Taylor formula at order 2

- Sufficient condition: if $\nabla f(\boldsymbol{x}^*) = 0$ and $\nabla^2 f(\boldsymbol{x}^*)$ is positive definite, then $\boldsymbol{x}^*$ is a strict local minimum

Proof of Sufficient Condition:

- Let $\lambda > 0$ be the smallest eigenvalue of $\nabla^2 f(\boldsymbol{x}^*)$, using a second order Taylor expansion, we have for all $\boldsymbol{h}$:
- $f(\boldsymbol{x}^* + \boldsymbol{h}) - f(\boldsymbol{x}^*) = \nabla f(\boldsymbol{x}^*)^T \boldsymbol{h} + \frac{1}{2} \boldsymbol{h}^T \nabla^2 f(\boldsymbol{x}^*) \boldsymbol{h} + o(||\boldsymbol{h}||^2)$

$$> \frac{\lambda}{2} ||\boldsymbol{h}||^2 + \mathrm{o}(||\boldsymbol{h}||^2) = \left(\frac{\lambda}{2} + \frac{o(||\boldsymbol{h}||^2)}{||\boldsymbol{h}||^2} \right) ||\boldsymbol{h}||^2$$

Convex Functions

Let U be a convex open set of $\mathbb{R}^n$ and $f: U \to \mathbb{R}$. The function f is said to be convex if for all $\boldsymbol{x}, \boldsymbol{y} \in U$ and for all $t \in [0,1]$

$$f\big((1-t)\boldsymbol{x} + t\boldsymbol{y}\big) \leq (1-t)f(\boldsymbol{x}) + tf(\boldsymbol{y})$$

Theorem

If f is differentiable, then f is convex if and only if for all $\boldsymbol{x}, \boldsymbol{y}$

$$f(\boldsymbol{y}) - f(\boldsymbol{x}) \geq \big(\nabla f(x)\big)^T (\boldsymbol{y} - \boldsymbol{x})$$

if $n = 1$, the curve is on top of the tangent

If f is twice continuously differentiable, then f is convex if and only if $\nabla^2 f(\boldsymbol{x})$ is positive semi-definite for all $\boldsymbol{x}$.

Convex Functions: Why Convexity?

Examples of Convex Functions:

- $f(\boldsymbol{x}) = a^T\boldsymbol{x} + b$
- $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T A\boldsymbol{x} + a^T\boldsymbol{x} + b$, A symmetric positive definite
- the negative of the entropy function $(\mathrm{i.e.}\, f(\boldsymbol{x}) = -\sum_{i=1}^{n} \boldsymbol{x}_i \ln(\boldsymbol{x}_{\boldsymbol{i}})\,)$

Exercise:

Let $f: U \to \mathbb{R}$ be a convex and differentiable function on a convex open U.
Show that if $\nabla f(\boldsymbol{x}^*) = 0$, then $\boldsymbol{x}^*$ is a global minimum of f

Why is convexity an important concept?

Convex Functions: Why Convexity?

Examples of Convex Functions:

- $f(\boldsymbol{x}) = a^T \boldsymbol{x} + b$
- $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T A\boldsymbol{x} + a^T\boldsymbol{x} + b$, A symmetric positive definite
- the negative of the entropy function $(\mathrm{i.e.}\, f(\boldsymbol{x}) = -\sum_{i=1}^{n} \boldsymbol{x}_i \ln(\boldsymbol{x}_i)\,)$

Exercise:

Let $f: U \to \mathbb{R}$ be a convex and differentiable function on a convex open U.
Show that if $\nabla f(\boldsymbol{x}^*) = 0$, then $\boldsymbol{x}^*$ is a global minimum of f

Why is convexity an important concept?

local minima are also global under convexity assumption.

Constrained Optimization

Equality Constraint

Objective:

Generalize the necessary condition of $\nabla f(x) = 0$ at the optima of f *when f is in $\mathcal{C}^1$, i.e. is differentiable and its differential is continuous*

Theorem:

Be U an open set of $(E, \| \ \|)$, and $f: U \to \mathbb{R}$, $g: U \to \mathbb{R}$ in $\mathcal{C}^1$.

Let $a \in E$ satisfy

$$\begin{cases} f(a) = \inf\{f(x) \mid x \in \mathbb{R}^n, g(x) = 0\} \\ g(a) = 0 \end{cases}$$

i.e. a is optimum of the problem

If $\nabla g(a) \neq 0$, then there exists a constant $\lambda \in \mathbb{R}$ called *Lagrange multiplier*, such that

$$\underbrace{\nabla f(a) + \lambda \nabla g(a) = 0}_{\text{i.e. gradients of } f \text{ and } g \text{ in } a \text{ are colinear}} \quad \text{Euler} - \text{Lagrange equation}$$

Geometrical Interpretation Using an Example

Exercise:

Consider the problem

$$\inf \ \{ f(x,y) \mid (x,y) \in \mathbb{R}^2, g(x,y) = 0\}$$

$f(x,y) = y - x^2 \qquad g(x,y) = x^2 + y^2 - 1 = 0$

1) Plot the level sets of f, plot $g = 0$
2) Compute ∇f and ∇g
3) Find the solutions with $\nabla f + \lambda \nabla g$ = 0

   equation solving with 3 unknowns (x, y, λ)
4) Plot the solutions of 3) on top of the level set graph of 1)

Answer

- $(x_1, y_1, \lambda_1) = \left(0, 1, -\frac{1}{2}\right)$ [max local]
- $= \left(0, -1, \frac{1}{2}\right)$ [max local]
- $= \left(\sqrt{\frac{3}{4}}, -\frac{1}{2}, 1\right)$ [min global]
- $= \left(-\sqrt{\frac{3}{4}}, -\frac{1}{2}, 1\right)$ [min global]

Note:

Here we see clearly that the previous conditions are necessary conditions but not sufficient conditions.

Interpretation of Euler-Lagrange Equation

Intuitive way to retrieve the Euler-Lagrange equation:

- In a local minimum a of a constrained problem, the hypersurfaces (or level sets) $f = f(a)$ and $g = 0$ are necessarily tangent (otherwise we could decrease f by moving along $g = 0$).
- Since the gradients $\nabla f(a)$ and $\nabla g(a)$ are orthogonal to the level sets $f = f(a)$ and $g = 0$, it follows that $\nabla f(a)$ and $\nabla g(a)$ are colinear.

Generalization to More than One Constraint

Theorem

- Assume $f : U \to \mathbb{R}$ and $g_k : U \to \mathbb{R}$ $(1 \leq k \leq p)$ are $\mathcal{C}^1$.
- Let a be such that

$$\begin{cases} f(a) = \inf \{ f(x) \mid x \in \mathbb{R}^n, \quad g_k(x) = 0, \quad 1 \leq k \leq p \} \\ g_k(a) = 0 \ \text{ for all } 1 \leq k \leq p \end{cases}$$

- If $\left(\nabla g_k(a) \right)_{1 \leq k \leq p}$ are linearly independent, then there exist p real constants $(\lambda_k)_{1 \leq k \leq p}$ such that

$$\nabla f(a) + \sum_{k=1}^{p} \lambda_k \nabla g_k(a) = 0$$

Lagrange multiplier

again: a does not need to be global but local minimum

The Lagrangian

- Define the Lagrangian on $\mathbb{R}^n \times \mathbb{R}^p$ as

$$\mathcal{L}(x, \{\lambda_k\}) = f(x) + \sum_{k=1}^{p} \lambda_k g_k(x)$$

- To find optimal solutions, we can solve the optimality system

$$\begin{cases} \text{Find } (x, \{\lambda_k\}) \in \mathbb{R}^n \times \mathbb{R}^p \text{ such that } \nabla f(x) + \sum_{k=1}^{p} \lambda_k \nabla g_k(x) = 0 \\ g_k(x) = 0 \ \text{ for all } 1 \leq k \leq p \end{cases}$$

$$\Leftrightarrow \begin{cases} \text{Find } (x, \{\lambda_k\}) \in \mathbb{R}^n \times \mathbb{R}^p \text{ such that } \nabla_x \mathcal{L}(x, \{\lambda_k\}) = 0 \\ \nabla_{\lambda_k} \mathcal{L}(x, \{\lambda_k\})(x) = 0 \ \text{ for all } 1 \leq k \leq p \end{cases}$$

Inequality Constraint: Definitions

Let $\mathcal{U} = \{x \in \mathbb{R}^n \mid g_k(x) = 0 \text{ (for } k \in E),\ g_k(x) \leq 0 \text{ (for } k \in I)\}$.

Definition:

The points in $\mathbb{R}^n$ that satisfy the constraints are also called *feasible* points.

Definition:

Let $a \in \mathcal{U}$, we say that the constraint $g_k(x) \leq 0$ (for $k \in I$) is *active* in a if $g_k(a) = 0$.

Inequality Constraint: Karush-Kuhn-Tucker Theorem

Theorem (Karush-Kuhn-Tucker, KKT):

Let U be an open set of $(\mathbb{R}^n, ||\ ||)$ and $f: U \to \mathbb{R}$, $g_k: U \to \mathbb{R}$, all $\mathcal{C}^1$

Furthermore, let $a \in U$ satisfy

$$\begin{cases} f(a) = \inf(f(x) \mid x \in \mathbb{R}^n, g_k(x) = 0 \text{ (for } k \in E), g_k(x) \leq 0 \text{ (for } k \in \text{I}) \\ g_k(a) = 0 \text{ (for } k \in E) \\ g_k(a) \leq 0 \text{ (for } k \in I) \end{cases}$$

also works again for a being a local minimum

Let I_a^0 be the set of constraints that are active in a. Assume that $\left(\nabla g_k(a)\right)_{k \in E \cup I_a^0}$ are linearly independent.

Then there exist $(\lambda_k)_{1 \leq k \leq p}$ that satisfy

$$\begin{cases} \nabla f(a) + \sum_{k=1}^{p} \lambda_k \nabla g_k(a) = 0 \\ g_k(a) = 0 \text{ (for } k \in E) \\ g_k(a) \leq 0 \text{ (for } k \in I) \\ \lambda_k \geq 0 \text{ (for } k \in I_a^0) \\ \lambda_k g_k(a) = 0 \text{ (for } k \in E \cup I) \end{cases}$$

Inequality Constraint: Karush-Kuhn-Tucker Theorem

Theorem (Karush-Kuhn-Tucker, KKT):

Let U be an open set of $(E, || \,||)$ and $f: U \to \mathbb{R}$, $g_k: U \to \mathbb{R}$, all $\mathcal{C}^1$

Furthermore, let $a \in U$ satisfy

$$\begin{cases} f(a) = \inf(f(x) \mid x \in \mathbb{R}^n, g_k(x) = 0\ (\text{for } k \in E), g_k(x) \leq 0\ (\text{for } k \in \mathrm{I}) \\ g_k(a) = 0\ (\text{for } k \in E) \\ g_k(a) \leq 0\ (\text{for } k \in I) \end{cases}$$

Let I_a^0 be the set of constraints that are active in a. Assume that $\left(\nabla g_k(a)\right)_{k \,\in\, E \,\cup\, I_a^0}$ are linearly independent.

Then there exist $(\lambda_k)_{1 \leq k \leq p}$ that satisfy

$$\begin{cases} \nabla f(a) + \displaystyle\sum_{k=1}^{p} \lambda_k \nabla g_k(a) = 0 \\ g_k(a) = 0\ (\text{for } k \in E) \\ g_k(a) \leq 0\ (\text{for } k \in I) \\ \lambda_k \geq 0\ (\text{for } k \in I_a^0) \\ \lambda_k g_k(a) = 0\ (\text{for } k \in E \cup I) \end{cases}$$

either active constraint or $\lambda_k = 0$

Descent Methods

Descent Methods

General principle

❶ choose an initial point $\boldsymbol{x}_0$, set $t = 0$

❷ while not happy

- choose a descent direction $\boldsymbol{d}_t \neq 0$
- line search:
    - choose a step size $\sigma_t > 0$
    - set $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \sigma_t \boldsymbol{d}_t$
- set $t = t + 1$

Remaining questions

- how to choose $\boldsymbol{d}_t$?
- how to choose σ_t?

Gradient Descent

Rationale: $\boldsymbol{d}_t = -\nabla f(\boldsymbol{x}_t)$ is a descent direction

indeed for f differentiable

$$f\big(x - \sigma\nabla f(x)\big) = f(x) - \sigma||\nabla f(x)||^2 + o(\sigma||\nabla f(x)||)$$
$$< f(x) \text{ for } \sigma \text{ small enough}$$

Step-size

- optimal step-size: $\sigma_t = \underset{\sigma}{\operatorname{argmin}} f(\boldsymbol{x}_t - \sigma\nabla f(\boldsymbol{x}_t))$
- **Line Search:** total or partial optimization w.r.t. σ
  Total is however often too "expensive" (needs to be performed at each iteration step)
  Partial optimization: execute a limited number of trial steps until a loose approximation of the optimum is found. Typical rule for partial optimization: Armijo rule (see next slides)

Typical stopping criterium:

norm of gradient smaller than ϵ

The Armijo-Goldstein Rule

Choosing the step size:

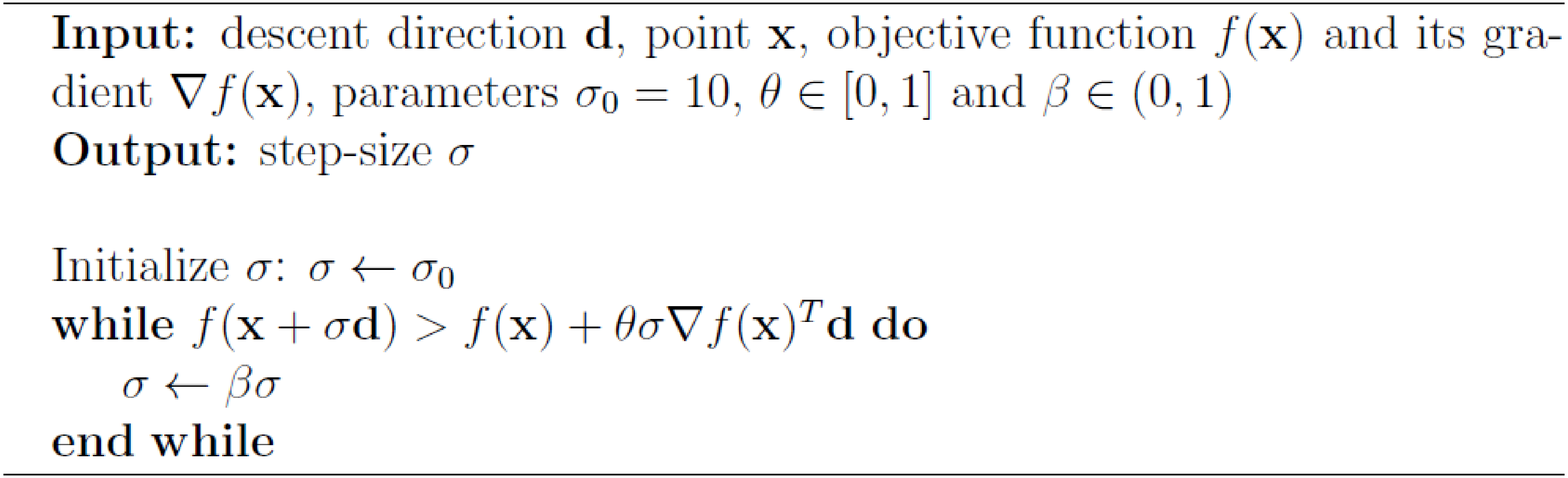
- Only to decrease f-value not enough to converge (quickly)
- Want to have a reasonably large decrease in f

Armijo-Goldstein rule:

- also known as backtracking line search
- starts with a (too) large estimate of σ and reduces it until f is reduced enough
- what is enough?
    - assuming a linear f e.g. $m_k(x) = f(x_k) + \nabla f(x_k)^T(x - x_k)$
    - expected decrease if step of σ_k is done in direction $\boldsymbol{d}$: $\sigma_k \nabla f(x_k)^T \boldsymbol{d}$
    - actual decrease: $f(x_k) - f(x_k + \sigma_k \boldsymbol{d})$
    - stop if actual decrease is at least constant times expected decrease (constant typically chosen in [0, 1])

The Armijo-Goldstein Rule

The Actual Algorithm:



Input: descent direction $\mathbf{d}$, point $\mathbf{x}$, objective function $f(\mathbf{x})$ and its gradient $\nabla f(\mathbf{x})$, parameters $\sigma_0 = 10$, $\theta \in [0, 1]$ and $\beta \in (0, 1)$

Output: step-size σ

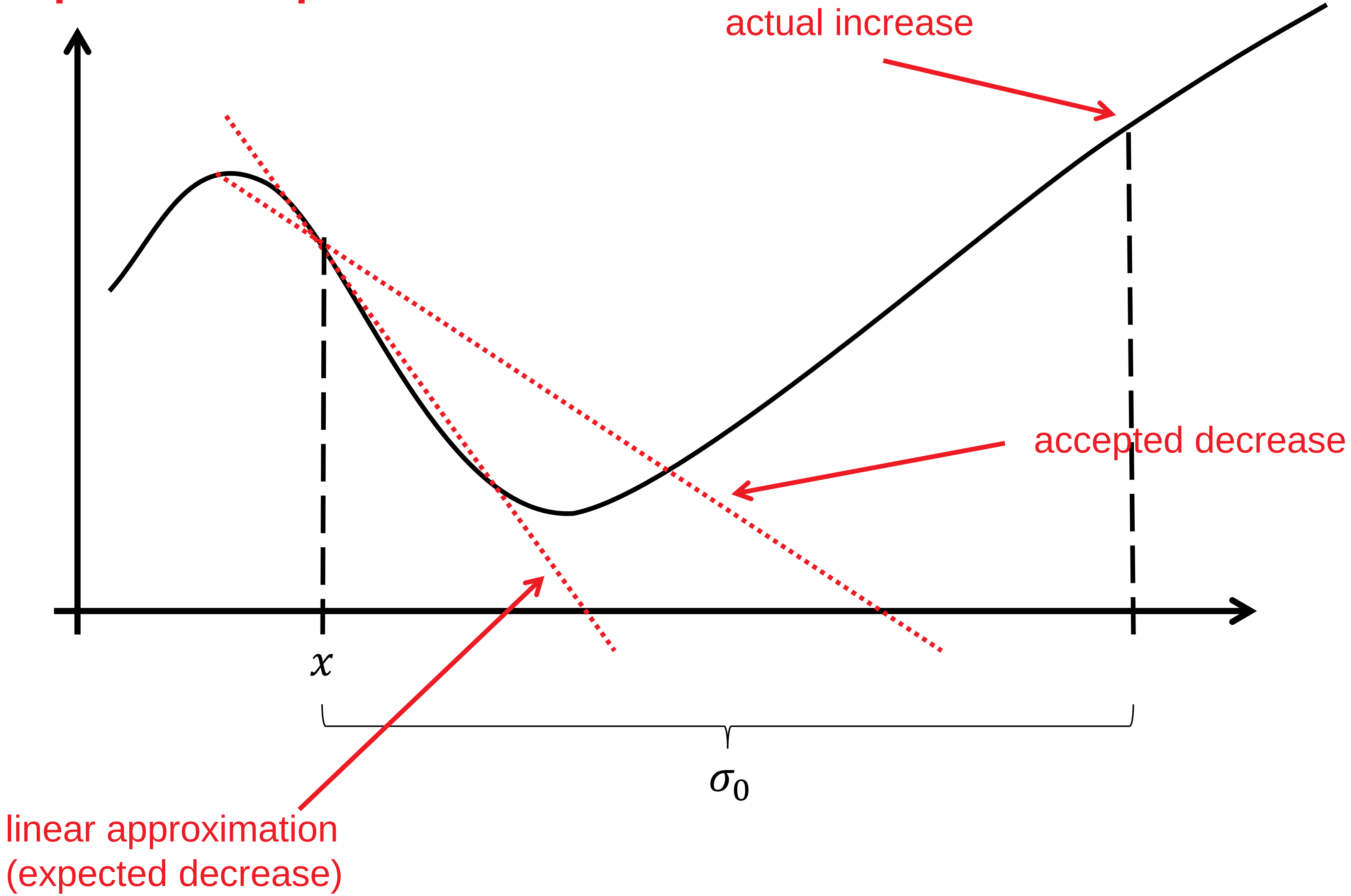
```
Initialize $\sigma$: $\sigma \leftarrow \sigma_0$
while $f(\mathbf{x} + \sigma \mathbf{d}) > f(\mathbf{x}) + \theta \sigma \nabla f(\mathbf{x})^T \mathbf{d}$ do
    $\sigma \leftarrow \beta \sigma$
end while
```

Armijo, in his original publication chose $\beta = \theta = 0.5$.

Choosing $\theta = 0$ means the algorithm accepts any decrease.

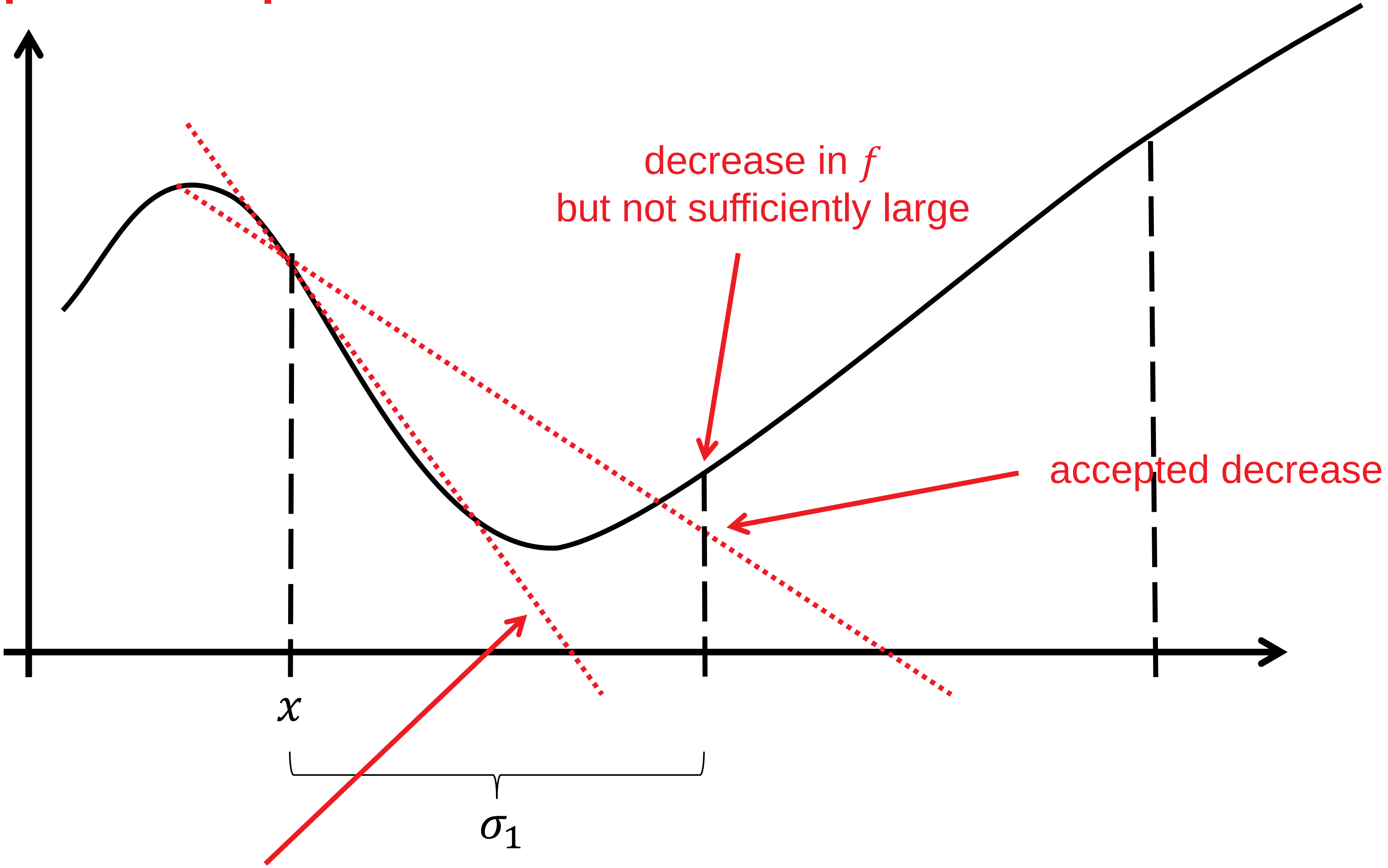
The Armijo-Goldstein Rule

Graphical Interpretation



The Armijo-Goldstein Rule

Graphical Interpretation



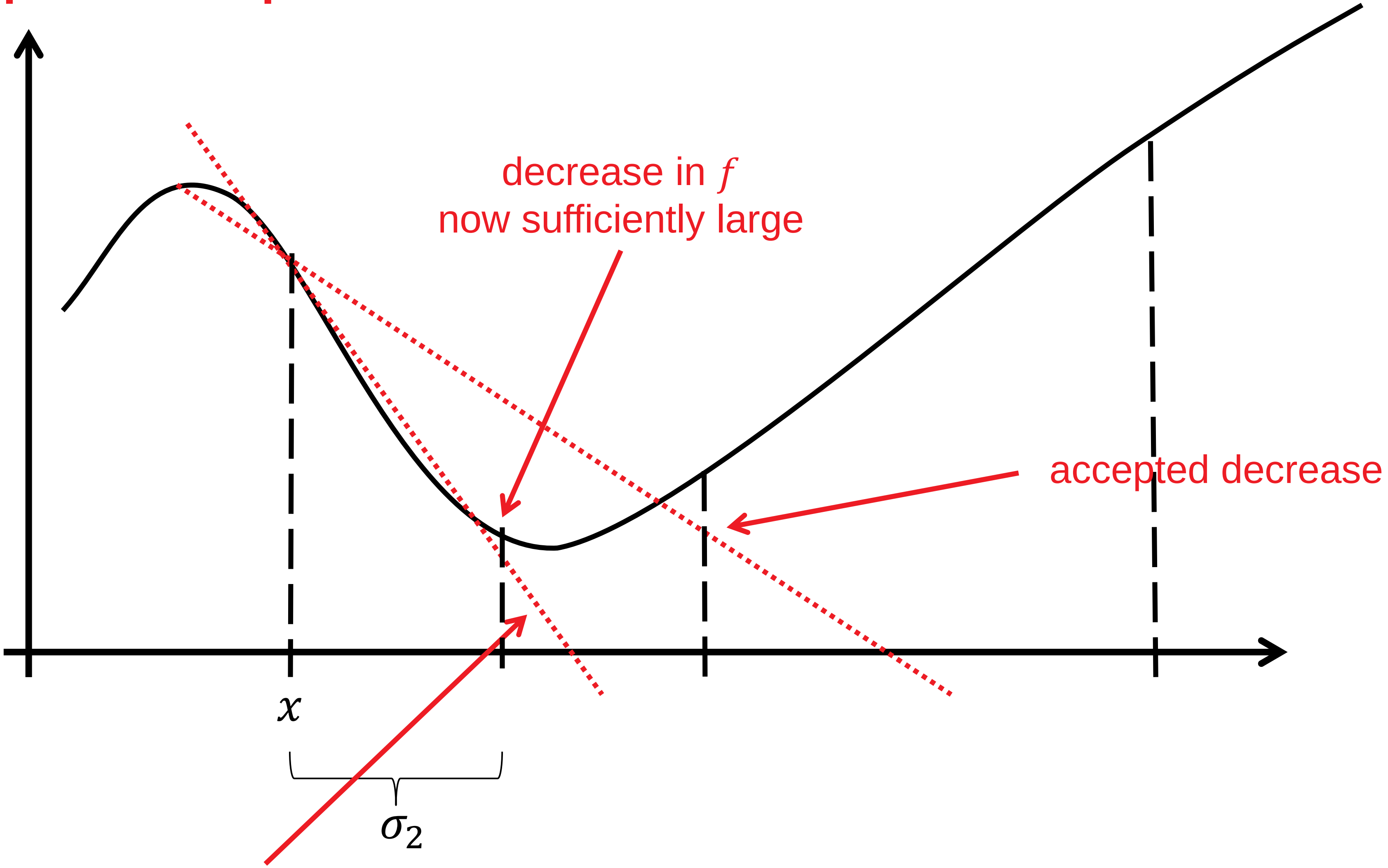
decrease in f
but not sufficiently large

accepted decrease

linear approximation
(expected decrease)

The Armijo-Goldstein Rule

Graphical Interpretation



decrease in f
now sufficiently large

accepted decrease

linear approximation
(expected decrease)

Newton Algorithm

Newton Method

- descent direction: $-[\nabla^2 f(x_k)]^{-1}\nabla f(x_k)$ [so-called Newton direction]
- The Newton direction:
  - minimizes the best (locally) quadratic approximation of f: $\tilde{f}(x+\Delta x) = f(x) + \nabla f(x)^T \Delta x + \frac{1}{2}(\Delta x)^T \nabla^2 f(x) \Delta \mathrm{x}$
  - points towards the optimum on $f(x) = (x - x^*)^T A (x - x^*)$
- however, Hessian matrix is expensive to compute in general and its inversion is also not easy

quadratic convergence

$\left(\text{i.e.} \quad \lim_{k\to\infty} \frac{|x_{k+1}-x^*|}{|x_k - x^*|^2} = \mu > 0\right)$

Remark: Affine Invariance

Affine Invariance: same behavior on $f(x)$ and $f(Ax+b)$ for $A \in \mathrm{GLn}(\mathbb{R})$ = set of all invertible $n \times n$ matrices over $\mathbb{R}$

- Newton method is affine invariant

  see **http://users.ece.utexas.edu/~cmcaram/EE381V_2012F/Lecture_6_Scribe_Notes.final.pdf**

- same convergence rate on all convex-quadratic functions
- Gradient method not affine invariant

Quasi-Newton Method: BFGS

$x_{t+1} = x_t - \sigma_t H_t \nabla f(x_t)$ where H_t is an approximation of the inverse Hessian

Key idea of Quasi Newton:

successive iterates x_t, x_{t+1} and gradients $\nabla f(x_t), \nabla f(x_{t+1})$ yield second order information

$$q_t \approx \nabla^2 f(x_{t+1}) p_t$$

where $p_t = x_{t+1} - x_t$ and $q_t = \nabla f(x_{t+1}) - \nabla f(x_t)$

Most popular implementation of this idea: Broyden-Fletcher-Goldfarb-Shanno (BFGS)

- default in MATLAB's **`fminunc`** and python's **`scipy.optimize.minimize`**

Conclusions

I hope it became clear...

...what are the difficulties to cope with when solving numerical optimization problems

in particular dimensionality, non-separability and ill-conditioning

...what are gradient and Hessian

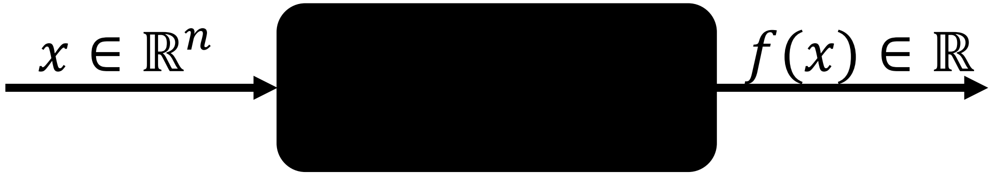
...what is the difference between gradient and Newton direction

...and that adapting the step size in descent algorithms is crucial.

Derivative-Free Optimization

Derivative-Free Optimization (DFO)

DFO = blackbox optimization



Why blackbox scenario?

- gradients are not always available (binary code, no analytical model, ...)
- or not useful (noise, non-smooth, ...)

- problem domain specific knowledge is used only within the black box, e.g. within an appropriate encoding
- some algorithms are furthermore function-value-free, i.e. *invariant* wrt. monotonous transformations of f.

Derivative-Free Optimization Algorithms

- (gradient-based algorithms which approximate the gradient by finite differences)

- coordinate descent
- pattern search methods, e.g. Nelder-Mead
- surrogate-assisted algorithms, e.g. NEWUOA or other trust-region methods
- other function-value-free algorithms
  - typically stochastic
  - evolution strategies (ESs) and Covariance Matrix Adaptation Evolution Strategy (CMA-ES)
  - differential evolution
  - particle swarm optimization
  - simulated annealing
  - ...

Downhill Simplex Method by Nelder and Mead

```
While not happy do:
    [assuming minimization of $f$ and that $x_1, \dots, x_{n+1} \in \mathbb{R}^n$ form a simplex]
  1) Order according to the values at the vertices: $f(x_1) \leq f(x_2) \leq \cdots \leq f(x_{n+1})$
  2) Calculate $x_o$, the centroid of all points except $x_{n+1}$.
  3) Reflection
    Compute reflected point $x_r = x_o + \alpha (x_o - x_{n+1})$ $(\alpha > 0)$
    If $x_r$ better than second worst, but not better than best: $x_{n+1} := x_r$ , and go to 1)
  4) Expansion
    If $x_r$ is the best point so far: compute the expanded point
                      $x_e = x_o + \gamma (x_r - x_o) (\gamma > 0)$
        If $x_e$ better than $x_r$ then $x_{n+1} \coloneqq x_e$ and go to 1)
        Else $x_{n+1} \coloneqq x_r$ and go to 1)
    Else (i.e. reflected point is not better than second worst) continue with 5)
  5) Contraction (here: $f(x_r) \geq f(x_n)$)
    Compute contracted point $x_c = x_o + \rho(x_{n+1} - x_o)$ $(0 < \rho \leq 0.5)$
    If $f(x_c) < f(x_{n+1})$: $x_{n+1} \coloneqq x_c$ and go to 1)
    Else go to 6)
  6) Shrink
    $x_i = x_1 + \sigma(x_i - x_1)$ for all $i \in \{2, \dots, n+1\}$ $(\sigma < 1)$ and go to 1)
```

J. A Nelder and R. Mead (1965). "A simplex method for function minimization". Computer Journal. ***7****: 308–313. doi:10.1093/comjnl/7.4.308*

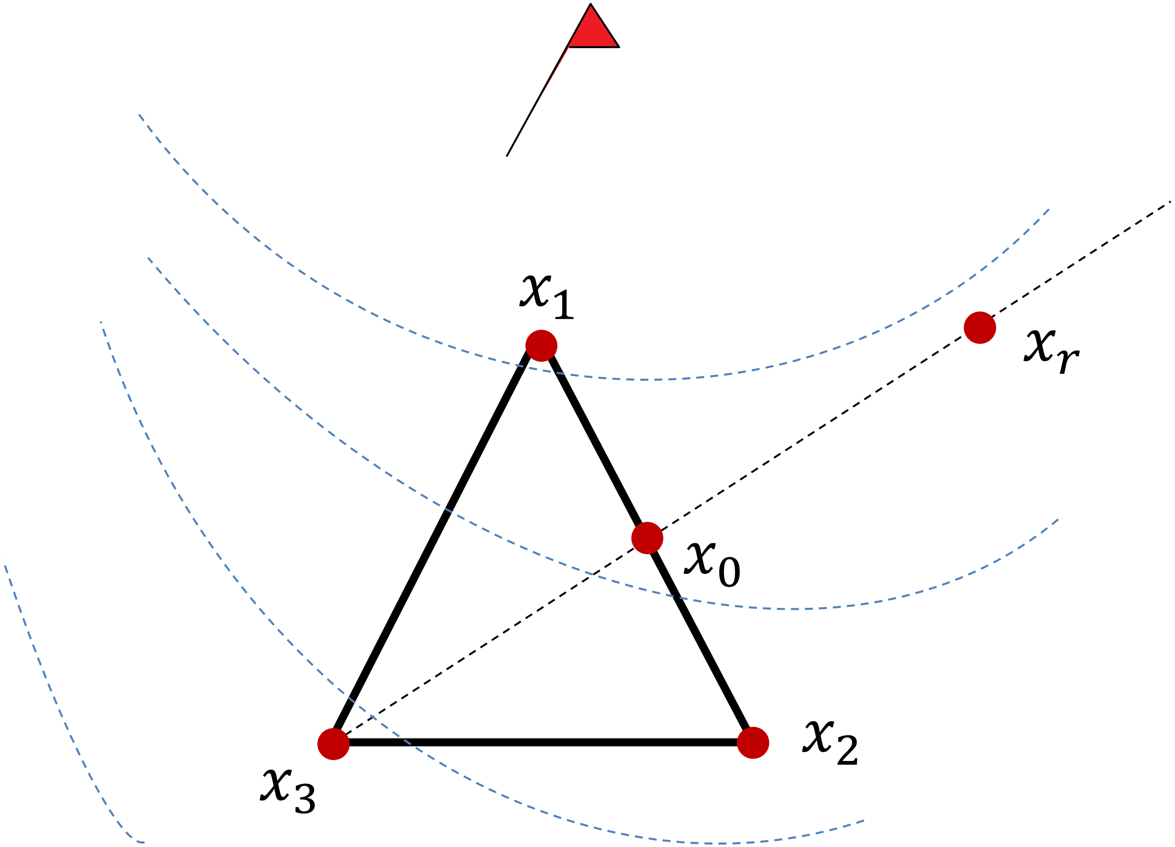
Nelder-Mead: Reflection

2) Calculate x_o, the centroid of all points except x_{n+1}.

3) Reflection

Compute reflected point $x_r \ = \ x_o \ + \alpha \ (x_o \ - x_{n+1}) \ (\alpha > 0)$

If x_r better than second worst, but not better than best: $x_{n+1} := x_r$, and go to 1)



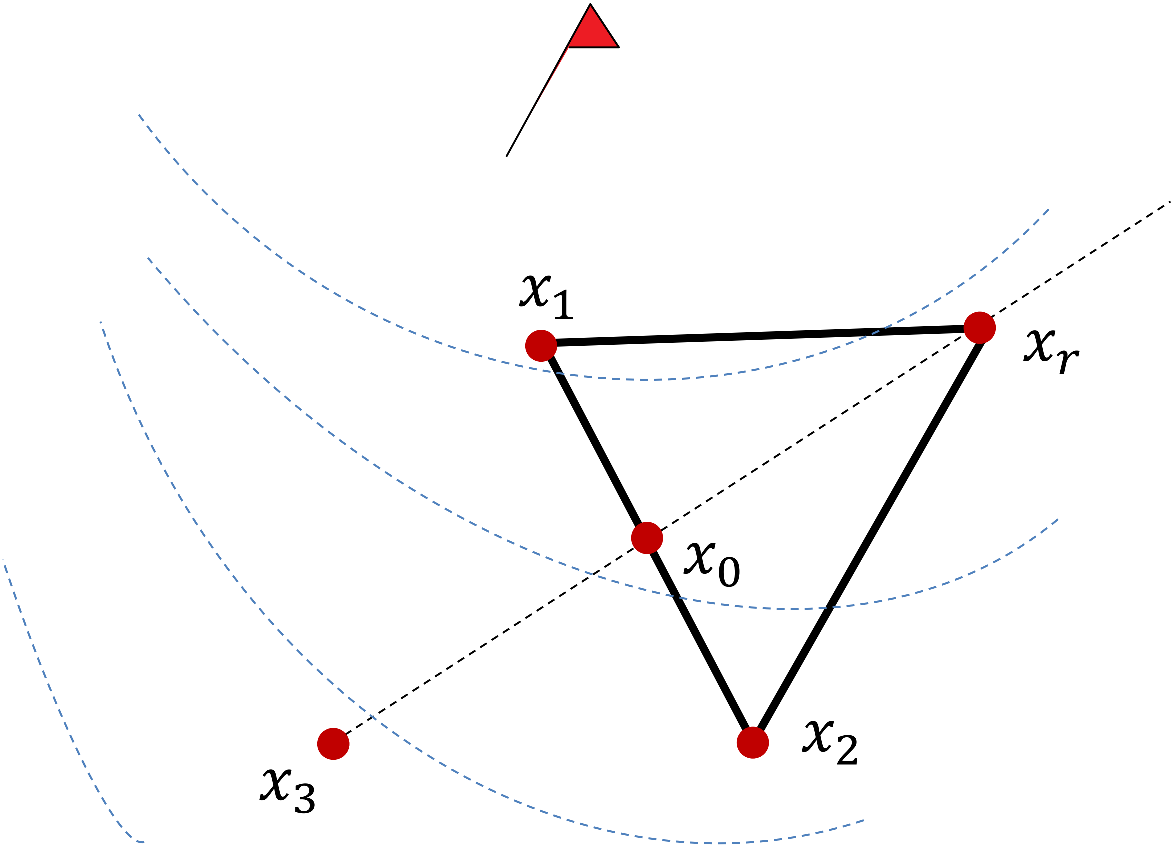
Nelder-Mead: Reflection

2) Calculate x_o, the centroid of all points except x_{n+1}.

3) Reflection

Compute reflected point $x_r \; = \; x_o \; + \alpha \; (x_o \; - x_{n+1}) \; (\alpha > 0)$

If x_r better than second worst, but not better than best: $x_{n+1} := x_r$, and go to 1)



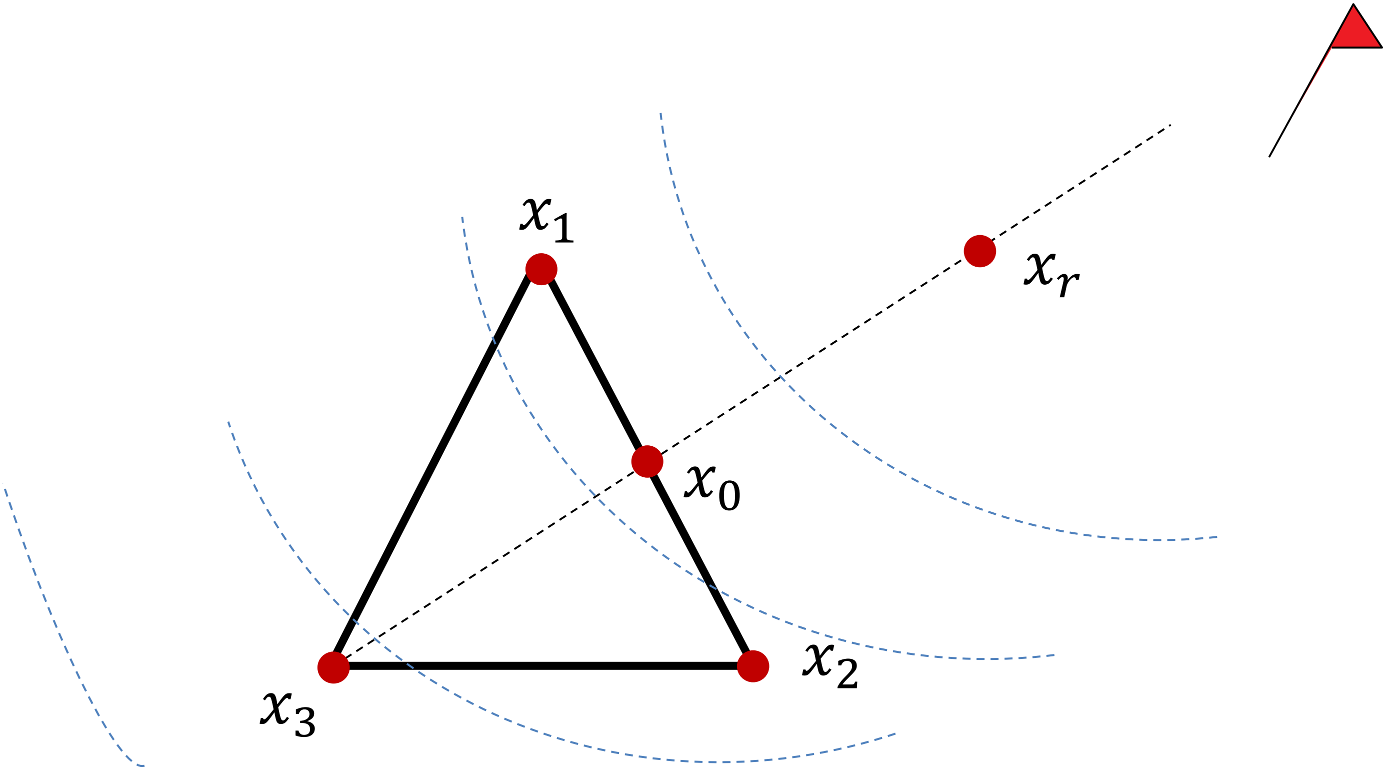
Nelder-Mead: Reflection

2) Calculate x_o, the centroid of all points except x_{n+1}.

3) Reflection

Compute reflected point $x_r \ = \ x_o \ + \alpha \ (x_o \ - x_{n+1}) \ (\alpha > 0)$

If x_r better than second worst, but not better than best: $x_{n+1} := x_r$, and go to 1)



Nelder-Mead: Expansion

2) Calculate x_o, the centroid of all points except x_{n+1}.

4) Expansion

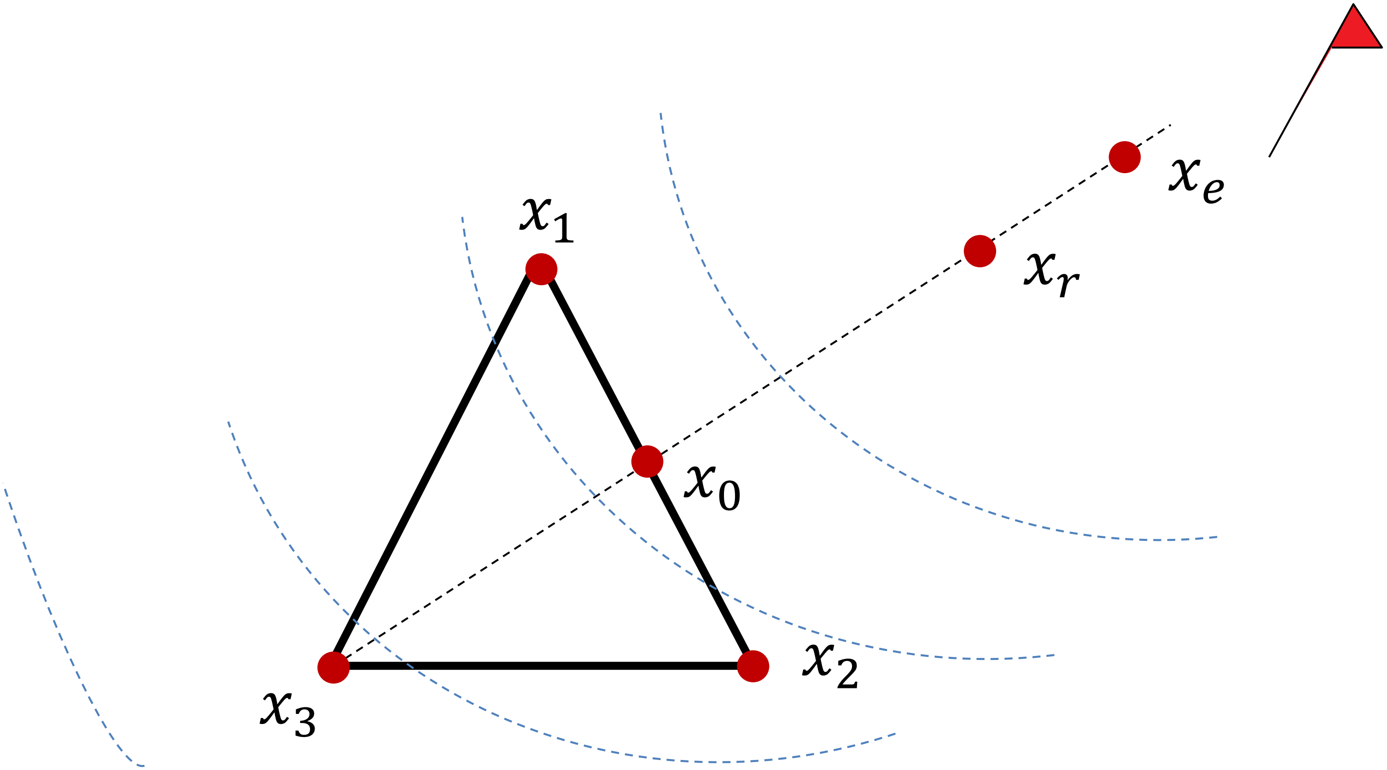
If x_r is the best point so far: compute the expanded point

$$x_e \;= x_o \;+ \gamma\,(x_r \;-\; x_o)(\gamma \;>\; 0)$$

If x_e better than x_r then $x_{n+1} \coloneqq x_e$ and go to 1)

Else $x_{n+1} \coloneqq x_r$ and go to 1)

Else (i.e. reflected point is not better than second worst) continue with 5)



Nelder-Mead: Expansion

2) Calculate x_o, the centroid of all points except x_{n+1}.

4) Expansion

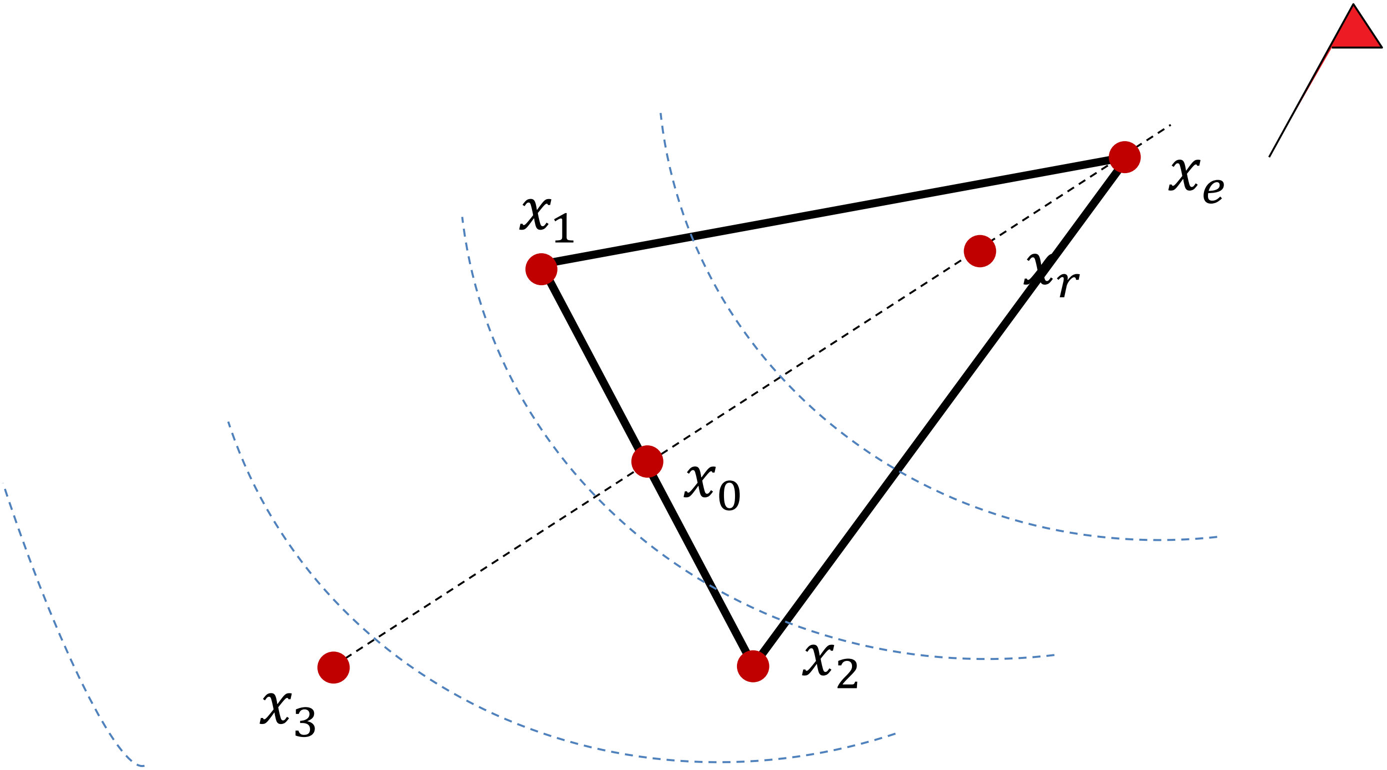
If x_r is the best point so far: compute the expanded point

$$x_e = x_o + \gamma \, (x_r - x_o)(\gamma > 0)$$

If x_e better than x_r then $x_{n+1} \coloneqq x_e$ and go to 1)

Else $x_{n+1} \coloneqq x_r$ and go to 1)

Else (i.e. reflected point is not better than second worst) continue with 5)



Nelder-Mead: Expansion

2) Calculate x_o, the centroid of all points except x_{n+1}.

4) Expansion

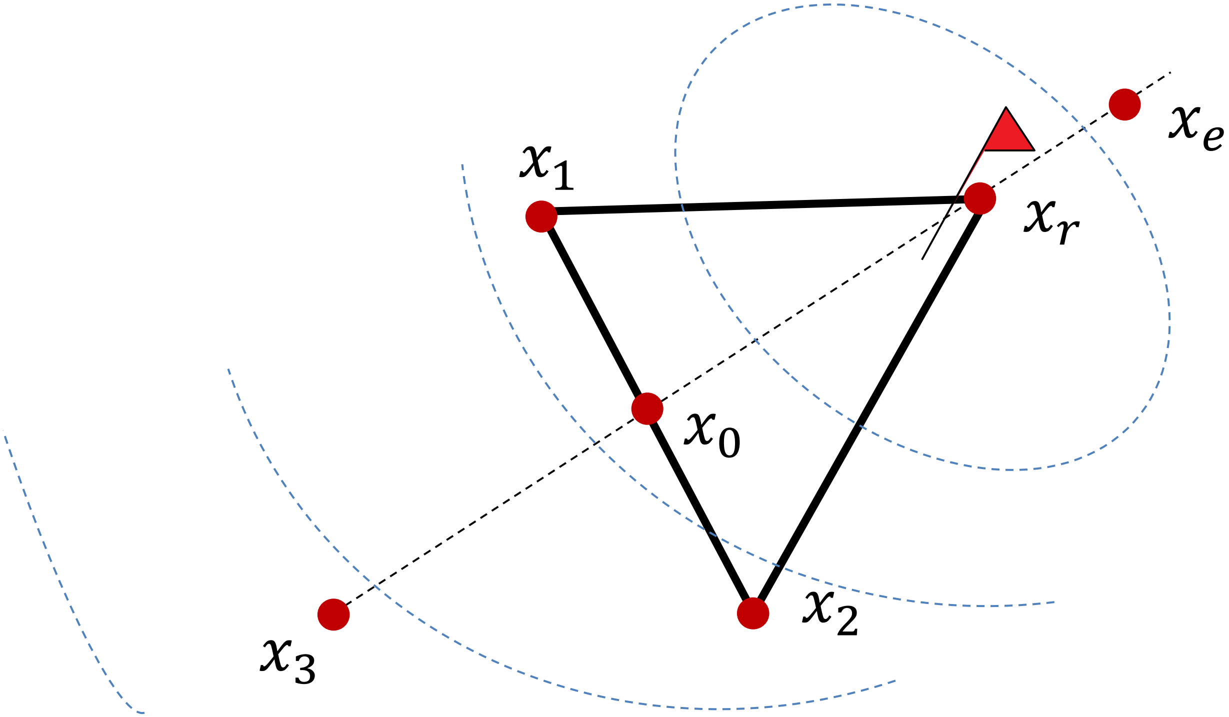
If x_r is the best point so far: compute the expanded point

$$x_e \ = x_o \ + \gamma \ (x_r \ - \ x_o)(\gamma \ > \ 0)$$

If x_e better than x_r then $x_{n+1} \coloneqq x_e$ and go to 1)

Else $x_{n+1} \coloneqq x_r$ and go to 1)

Else (i.e. reflected point is not better than second worst) continue with 5)



Nelder-Mead: Expansion

2) Calculate x_o, the centroid of all points except x_{n+1}.

4) Expansion

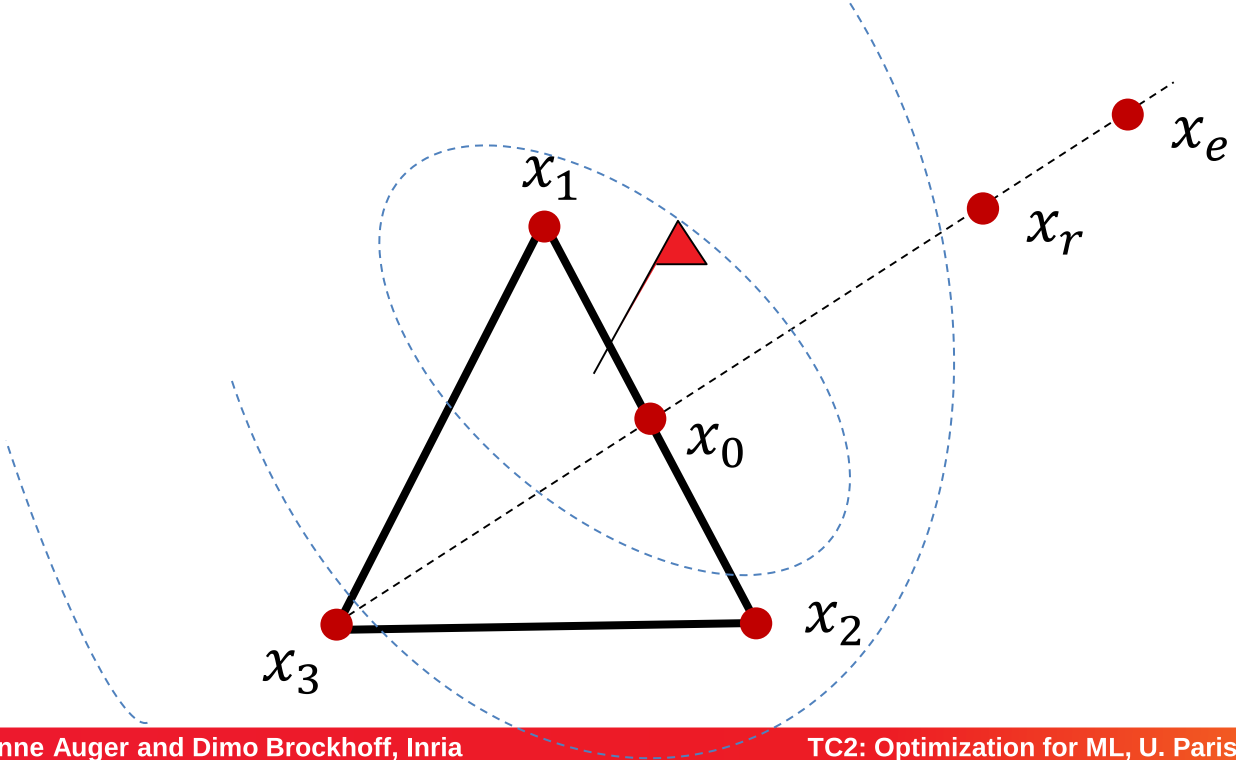
If x_r is the best point so far: compute the expanded point

$$x_e \;= x_o \;+\gamma\,(x_r \;-\; x_o)(\gamma \;>\; 0)$$

If x_e better than x_r then $x_{n+1} \coloneqq x_e$ and go to 1)

Else $x_{n+1} \coloneqq x_r$ and go to 1)

Else (i.e. reflected point is not better than second worst) continue with 5)



Nelder-Mead: Expansion

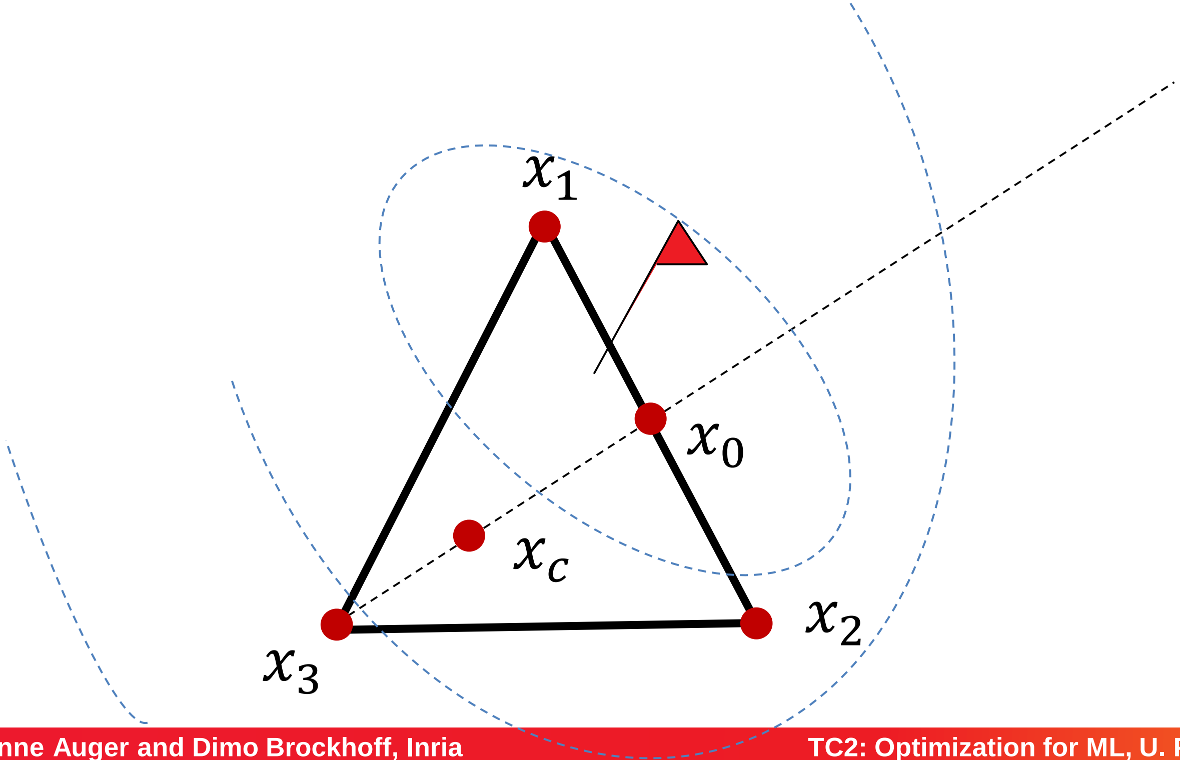
2) Calculate x_o, the centroid of all points except x_{n+1}.

5) Contraction (here: $f(x_r) \geq f(x_n)$)

Compute contracted point $x_c = x_o + \rho(x_{n+1} - x_o)$ $(0 < \rho \leq 0.5)$

If $f(x_c) < f(x_{n+1})$: $x_{n+1} := x_c$ and go to 1)

Else go to 6)



Nelder-Mead: Expansion

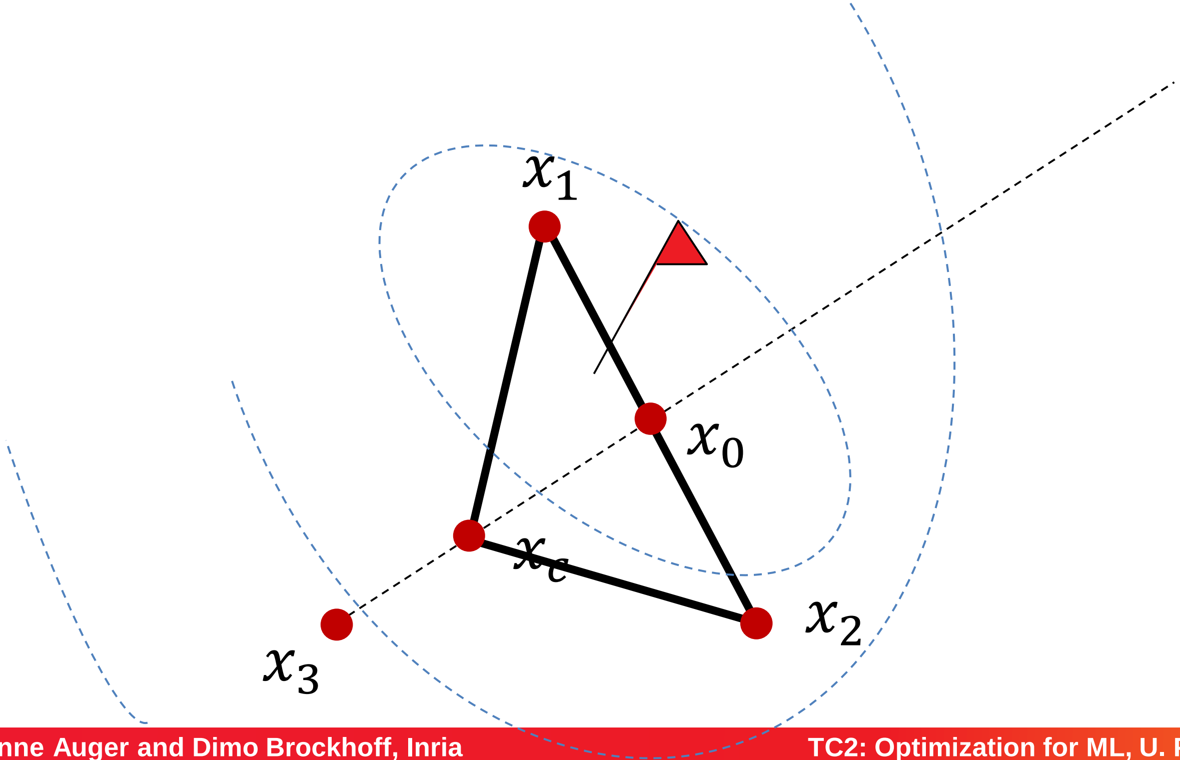
2) Calculate x_o, the centroid of all points except x_{n+1}.

5) Contraction (here: $f(x_r) \geq f(x_n)$)

Compute contracted point $x_c = x_o + \rho(x_{n+1} - x_o)$ $(0 < \rho \leq 0.5)$

If $f(x_c) < f(x_{n+1})$: $x_{n+1} := x_c$ and go to 1)

Else go to 6)

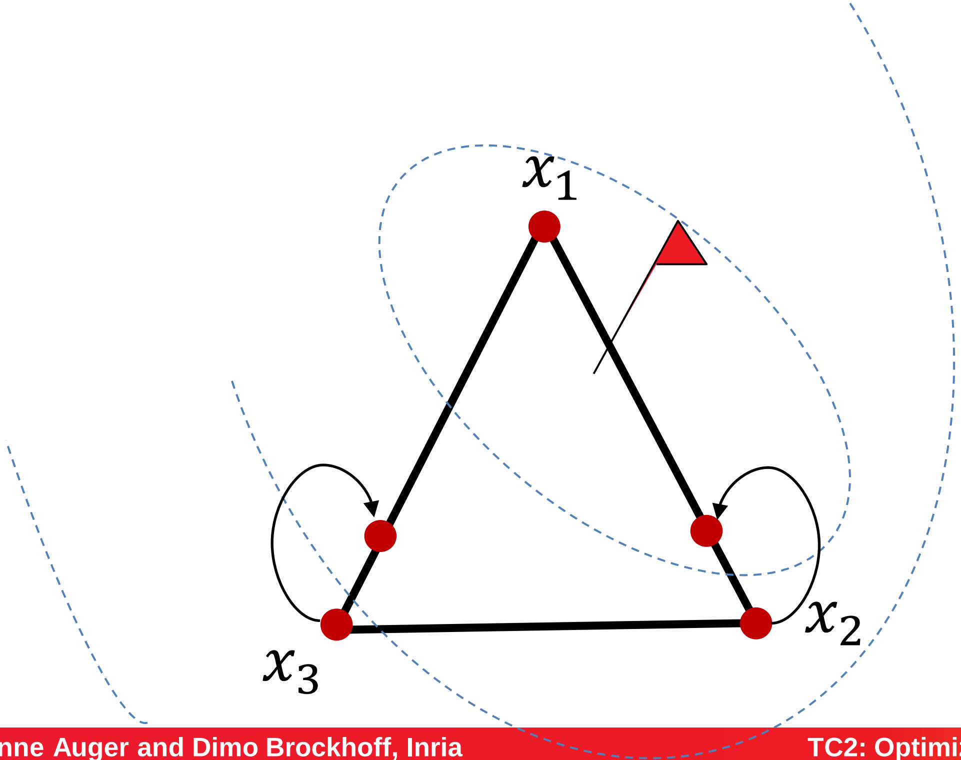


Nelder-Mead: Expansion

2) Calculate x_o, the centroid of all points except x_{n+1}.

6) Shrink

$x_i = x_1 + \sigma(x_i - x_1)$ for all $i \in \{2, \ldots, n+1\}$ and go to 1)

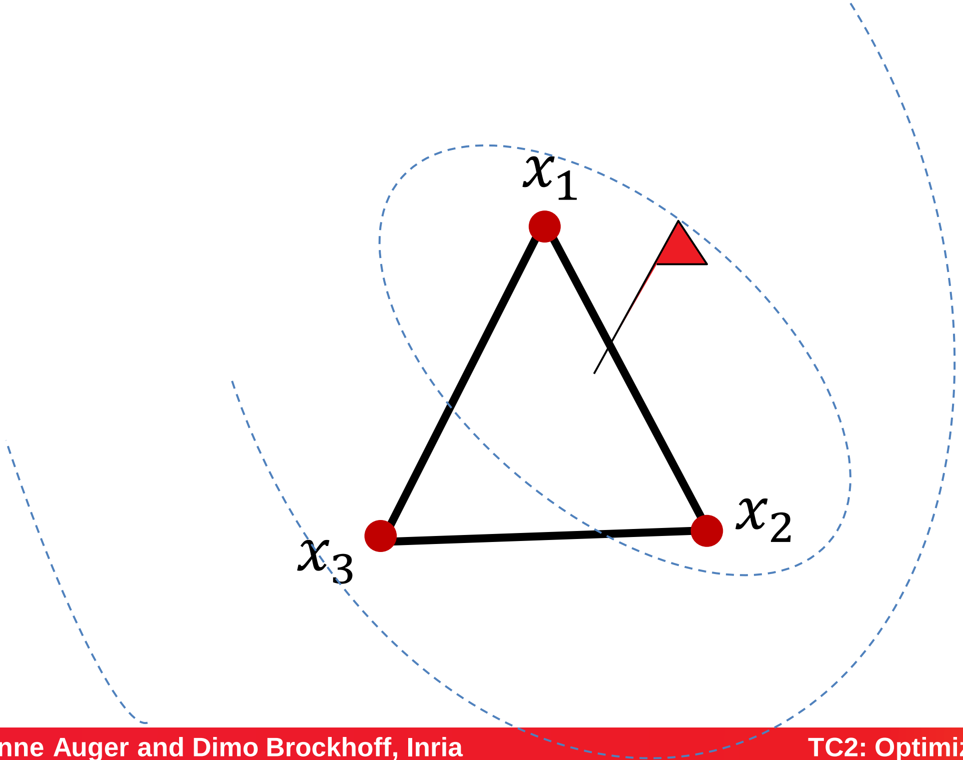


Nelder-Mead: Expansion

2) Calculate x_o, the centroid of all points except x_{n+1}.

6) Shrink

$x_i = x_1 + \sigma(x_i - x_1)$ for all $i \in \{2, \dots, n+1\}$ and go to 1)



Nelder-Mead: Standard Parameters

- reflection parameter : $\alpha = 1$
- expansion parameter: $\gamma = 2$
- contraction parameter: $\rho = \frac{1}{2}$
- shrink parameter: $\sigma = \frac{1}{2}$

some visualizations of example runs can be found here:
https://en.wikipedia.org/wiki/Nelder%E2%80%93Mead_method

stochastic algorithms

Stochastic Search Template

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

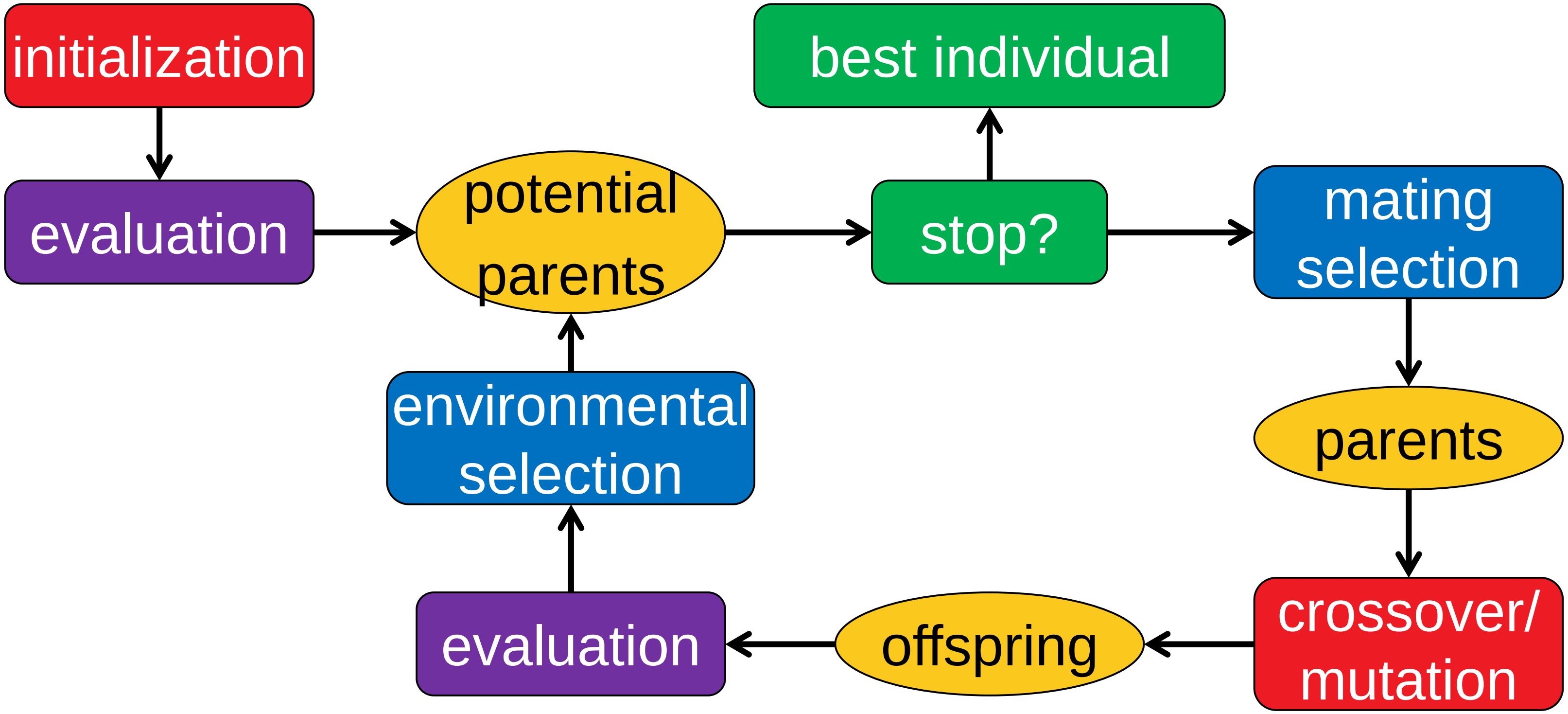
```
Initialize distribution parameters $\theta$, set population size $\lambda \in \mathbb{N}$
While happy do:
    ▪ Sample distribution $P(\boldsymbol{x}|\theta) \to \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda \in \mathbb{R}^n$
    ▪ Evaluate $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda$ on $f$
    ▪ Update parameters $\theta \leftarrow F_\theta(\theta, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_\lambda, f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_\lambda))$
```

- All depends on the choice of P and F_θ

deterministic algorithms are covered as well

- In Evolutionary Algorithms, P and F_θ are often defined implicitly via their operators.

Generic Framework of an Evolutionary Algorithm



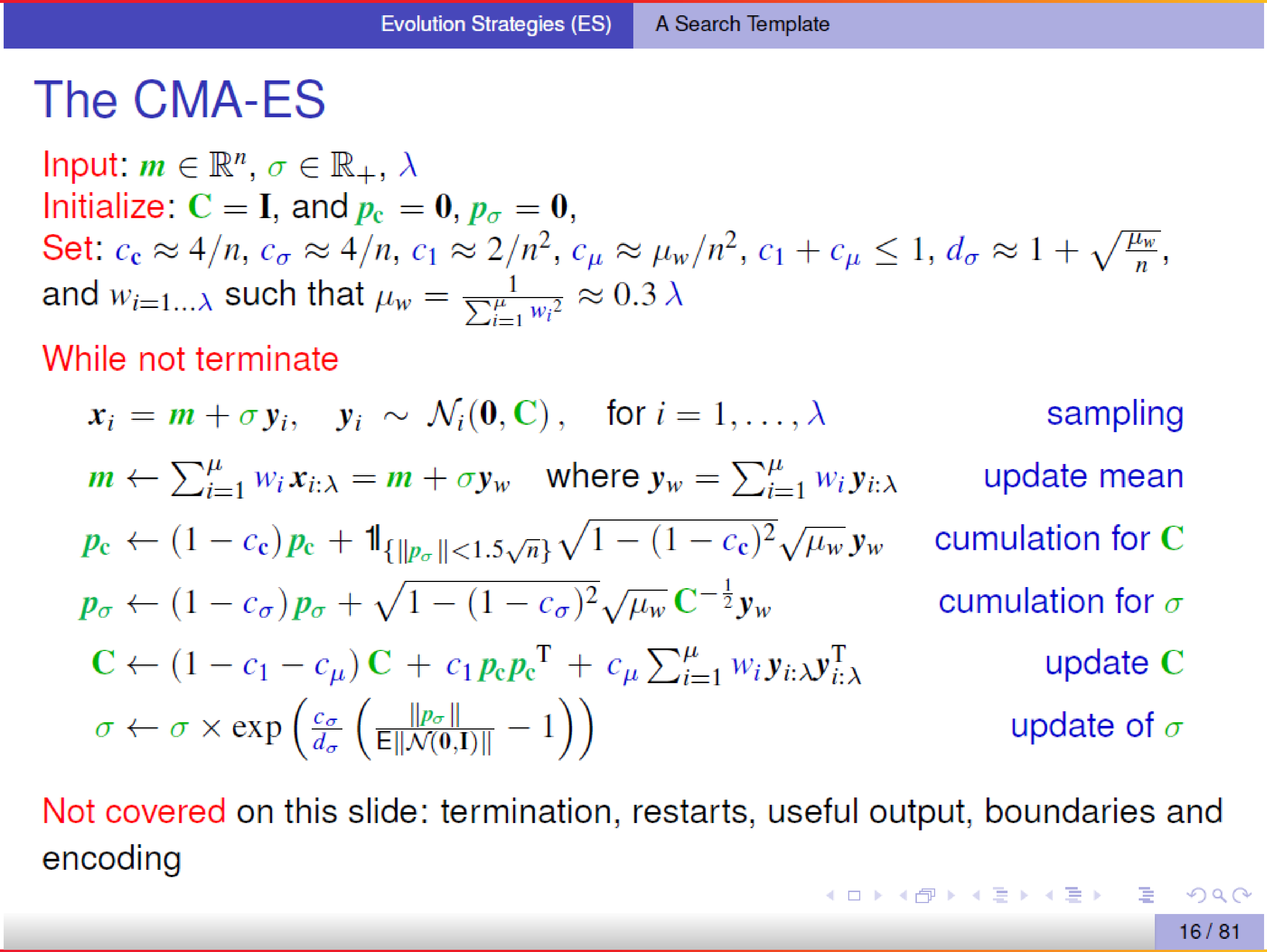
🟥 stochastic operators

🟦 “Darwinism”

🟩 stopping criteria

Nothing else: just interpretation change

CMA-ES in a Nutshell



The CMA-ES

Input: $m \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, λ
Initialize: $\mathbf{C} = \mathbf{I}$, and $p_c = \mathbf{0}$, $p_\sigma = \mathbf{0}$,
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

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

$$m \leftarrow \sum_{i=1}^{\mu} w_i x_{i:\lambda} = m + \sigma y_w \quad \text{where } y_w = \sum_{i=1}^{\mu} w_i y_{i:\lambda} \qquad \text{update mean}$$

$$p_c \leftarrow (1 - c_c)\, p_c + \mathbb{1}_{\{\|p_\sigma\| < 1.5\sqrt{n}\}} \sqrt{1 - (1 - c_c)^2} \sqrt{\mu_w}\, y_w \qquad \text{cumulation for } \mathbf{C}$$

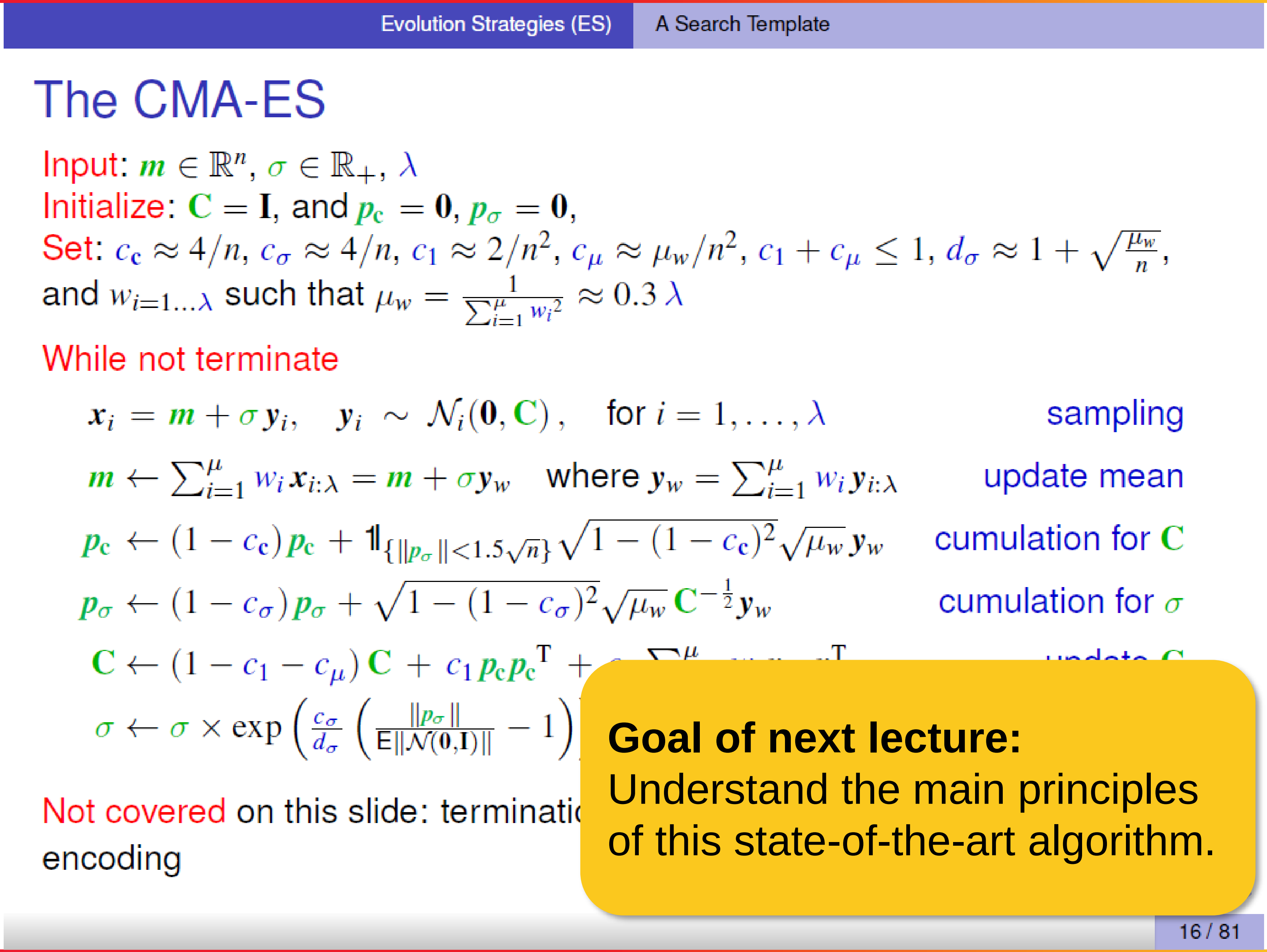
$$p_\sigma \leftarrow (1 - c_\sigma)\, p_\sigma + \sqrt{1 - (1 - c_\sigma)^2} \sqrt{\mu_w}\, \mathbf{C}^{-\frac{1}{2}} y_w \qquad \text{cumulation for } \sigma$$

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

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

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

CMA-ES in a Nutshell



The CMA-ES

Input: $\boldsymbol{m} \in \mathbb{R}^n$, $\sigma \in \mathbb{R}_+$, λ
Initialize: $\mathbf{C} = \mathbf{I}$, and $\boldsymbol{p}_\mathbf{c} = \mathbf{0}$, $\boldsymbol{p}_\sigma = \mathbf{0}$,
Set: $c_\mathbf{c} \approx 4/n$, $c_\sigma \approx 4/n$, $c_1 \approx 2/n^2$, $c_\mu \approx \mu_w/n^2$, $c_1 + c_\mu \leq 1$, $d_\sigma \approx 1 + \sqrt{\frac{\mu_w}{n}}$,
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

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

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

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

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

$\mathbf{C} \leftarrow (1 - c_1 - c_\mu)\,\mathbf{C} + c_1 \boldsymbol{p}_\mathbf{c} {\boldsymbol{p}_\mathbf{c}}^{\mathrm{T}} +$

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

Not covered on this slide: terminati
encoding

Goal of next lecture:
Understand the main principles of this state-of-the-art algorithm.