MSc Data Science for Business

Introduction to Machine Learning

MAP 534

Julie Josse, Erwan Scornet

OPTIMIZATION

Anne Auger

Credit for some of the slides to Robert M. Gower

Overview

1. Optimization and Machine Learning
2. Gradients, Hessian, optimality conditions, convexity
3. Descent Algorithms
4. Stochastic Gradient

Supervised Learning

Given a set of examples $\{(x^1, y^1), \ldots, (x^n, y^n)\}$ with x^i the features and y^i labels/targets

Find a mapping $h : x \in X \rightarrow y \in \mathbb{R}$ that will assign the "correct" target to each input

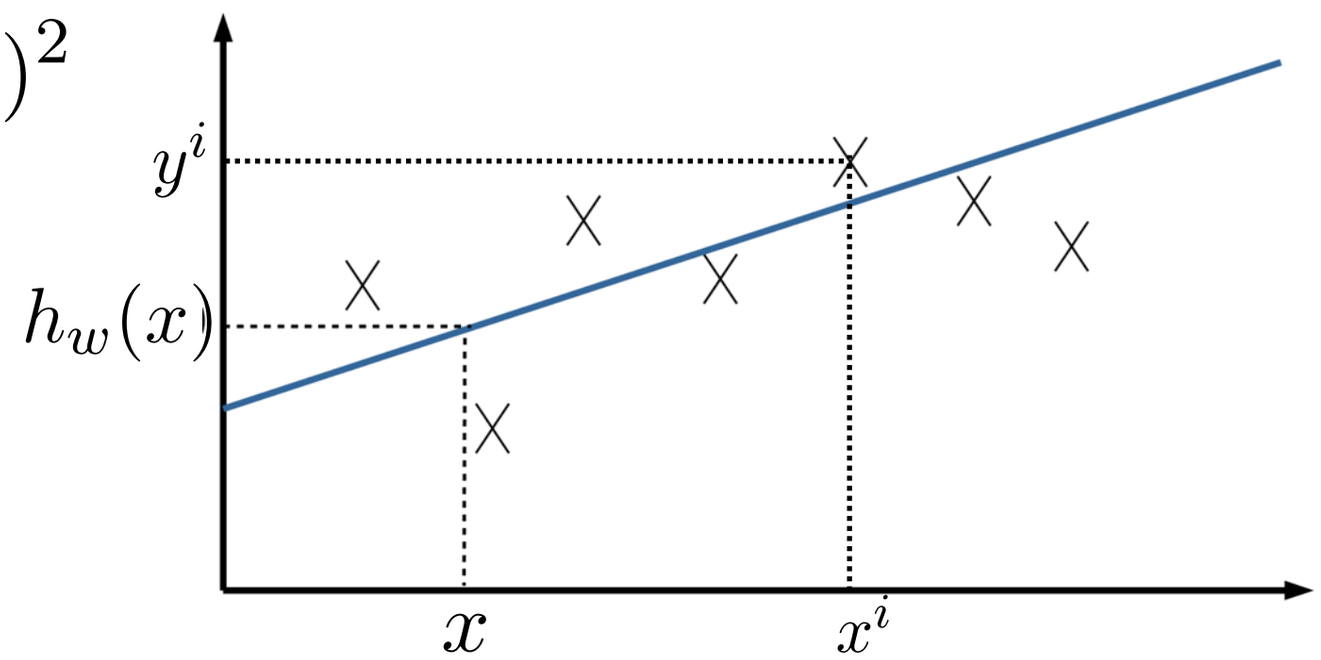
Example 1: Linear Regression

Hypothesis: linear model

$$h_w(x) = w_0 + w_1 x_1 + \ldots + w_{d-1} x_{d-1} \overset{x_0 = 1}{=} \langle w, x \rangle$$

Find $h_w(x)$ via solving the minimization problem

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (h_w(x^i) - y^i)^2$$



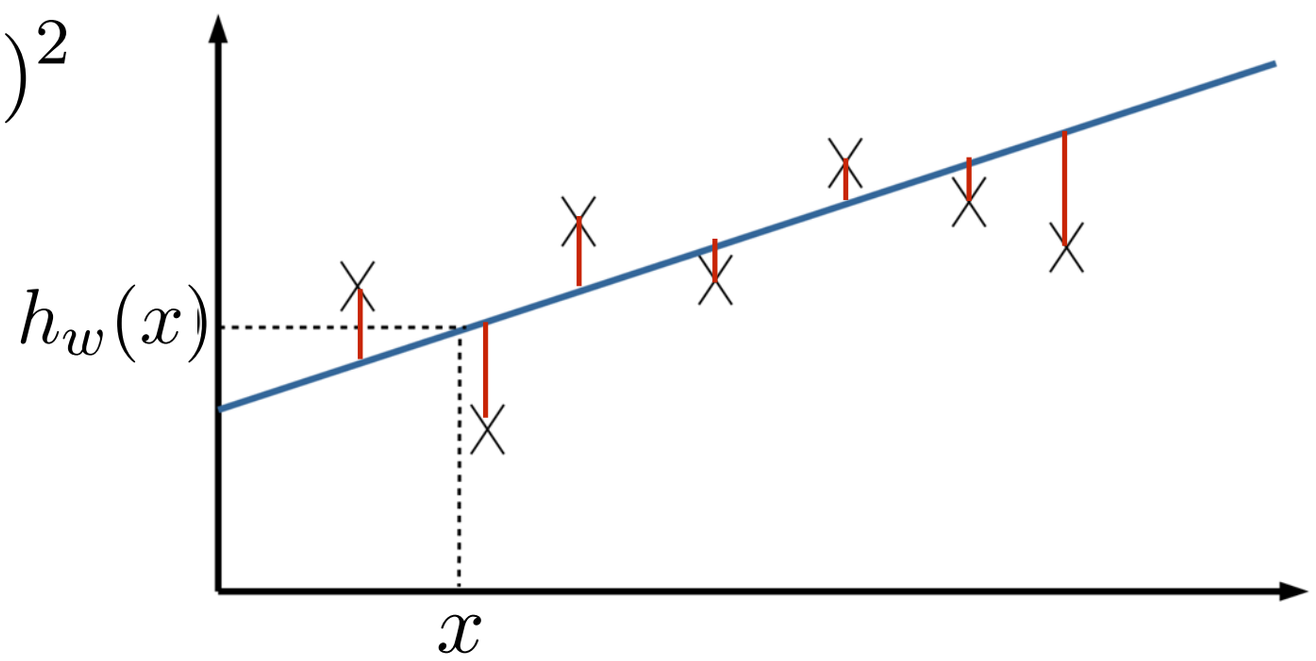
Example 1: Linear Regression

Hypothesis: linear model

$$h_w(x) = w_0 + w_1 x_1 + \ldots + w_{d-1} x_{d-1} \overset{x_0 = 1}{=} \langle w, x \rangle$$

Find $h_w(x)$ via solving the minimization problem

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (h_w(x^i) - y^i)^2$$



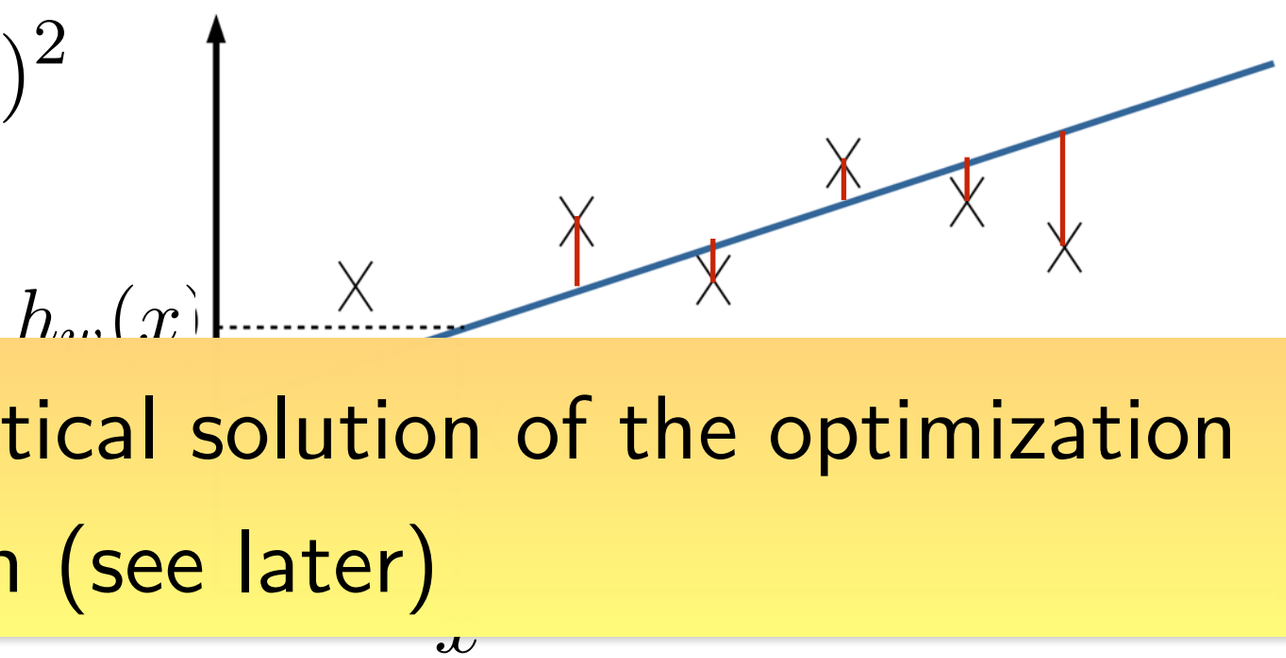
Example 1: Linear Regression

Hypothesis: linear model

$$h_w(x) = w_0 + w_1 x_1 + \ldots + w_{d-1} x_{d-1} \overset{x_0 = 1}{=} \langle w, x \rangle$$

Find $h_w(x)$ via solving the minimization problem

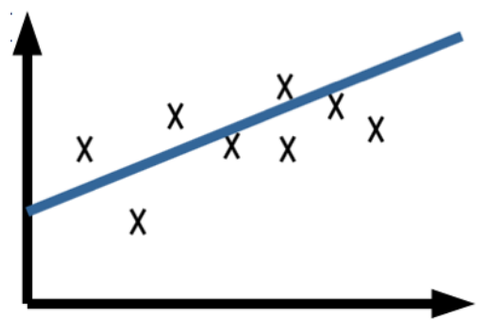
$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (h_w(x^i) - y^i)^2$$



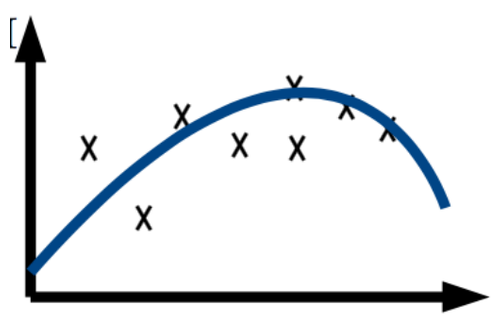
In this case we find an analytical solution of the optimization problem (see later)

Generalization: Parametrization of the Hypothesis

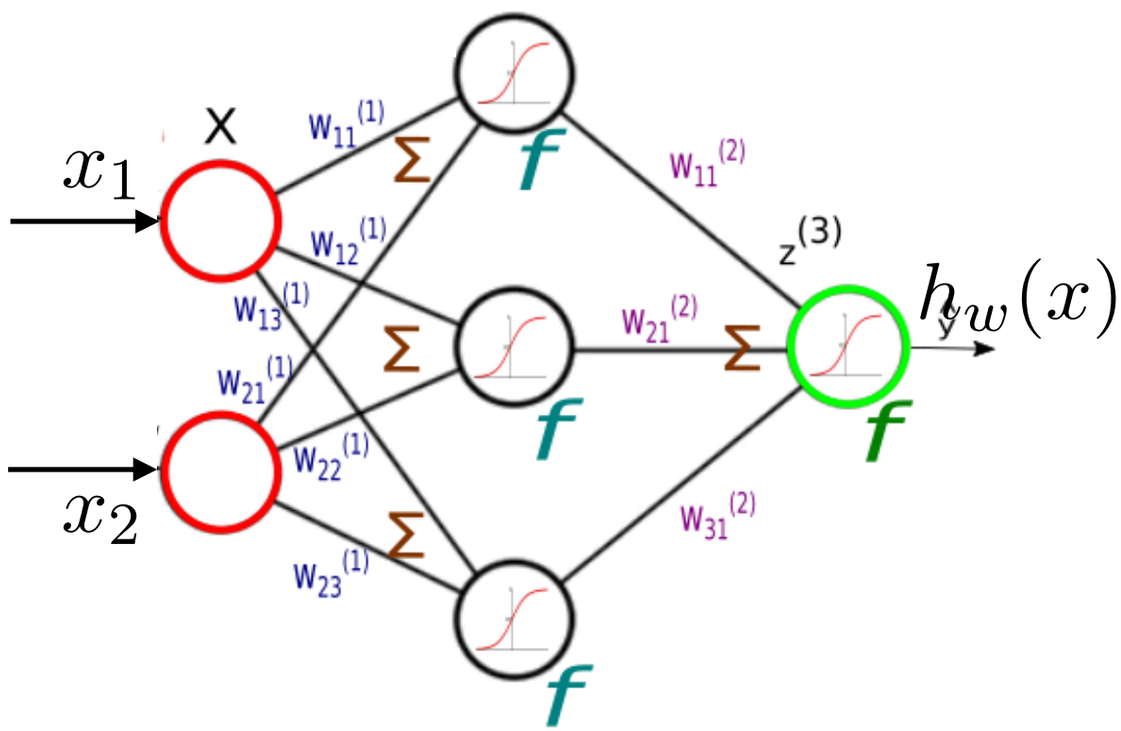
Linear: $h_w(x) = \langle w, x \rangle = \sum_{i=0}^{d-1} w_i x_i$



Polynomial: $h_w(x) = \sum_{i,j=0}^{d-1} w_{i,j} x_i x_j$



Neural network:



Generalization: Different Loss Functions

Start from the linear regression problem:

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (h_w(x^i) - y^i)^2$$

Let $y_h := h_w(x)$

Loss function: $l : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$

$$(y_h, y) \to l(y_h, y)$$

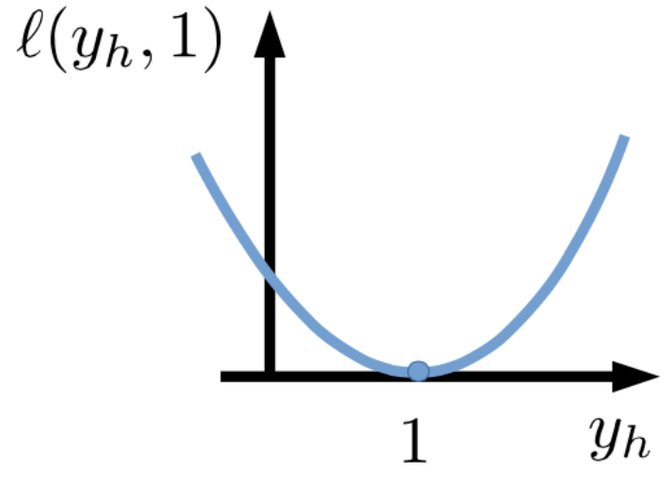
For linear regression
$l(y_h, y) = (y_h - y)^2$

Training (optimization) problem:

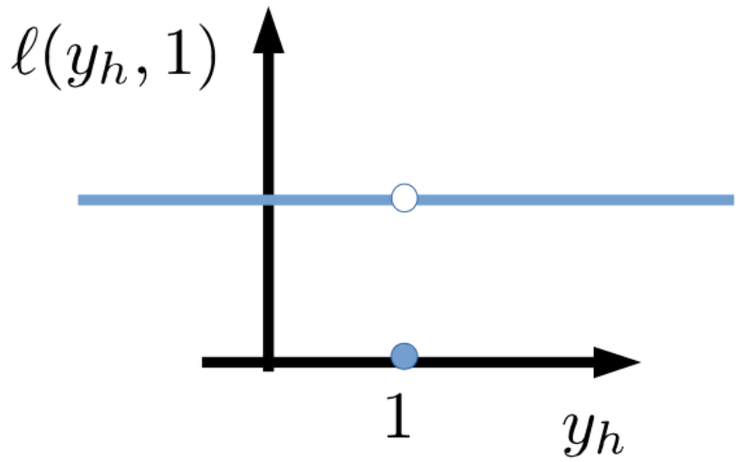
$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} l(h_w(x^i), y^i)$$

Generalization: Different loss functions

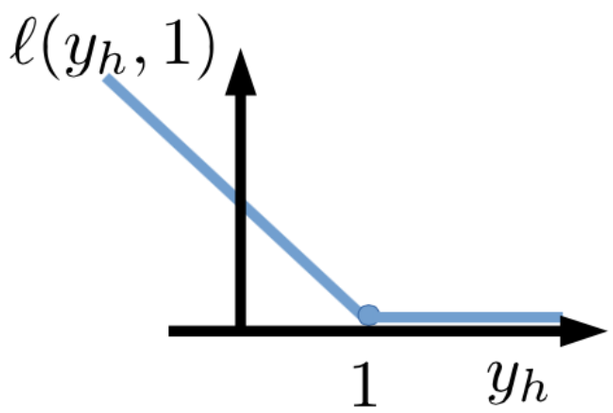
Quadratic loss: $l(y_h, y) = (y_h - y)^2$



Binary loss: $l(y_h, y) = \begin{cases} 0 \text{ if } y_h = y \\ 1 \text{ if } y_h \neq y \end{cases}$



Hinge loss: $l(y_h, y) = \max\{0, 1 - y_h y\}$



Example 2: Logistic Regression

Classification: $y^i = 0 \text{ or } 1$

Hypothesis: $h_w(x) = g(\langle w, x \rangle) = \dfrac{1}{1 + e^{-\langle w,x \rangle}} = \dfrac{e^{\langle w,x \rangle}}{e^{\langle w,x \rangle} + 1}$

$$g(z) = \frac{1}{1 + e^{-z}} \quad \text{sigmoid/logistic function}$$

$h_w(x)$ models $\Pr(y = 1|x, w)$ that is

$$\Pr(y = 1|x, w) = h_w(x)$$
$$\Pr(y = 0|x, w) = 1 - h_w(x)$$

or

$$\Pr(y|x, w) = (h_w(x))^y (1 - h_w(x))^{1-y}$$

Example 2: Logistic Regression

Classification: $y^i = 0 \text{ or } 1$

Hypothesis: $h_w(x) = g(\langle w, x \rangle) = \dfrac{1}{1 + e^{-\langle w,x \rangle}}$

How do we find w?

$h_w(x)$ models $\Pr(y = 1|x, w)$ that is

$$\Pr(y = 1|x, w) = h_w(x)$$
$$\Pr(y = 0|x, w) = 1 - h_w(x)$$

or

$$\Pr(y|x, w) = (h_w(x))^y (1 - h_w(x))^{1-y}$$

Example 2: Logistic Regression

Maximizing the log-likelihood of the data:

$$\max_{w \in \mathbb{R}^d} l(w) = \sum_{i=1}^{n} y^i \log(h_w(x^i)) + (1 - y^i) \log(1 - h_w(x^i))$$

In this case, no analytical solution. We need to use a **numerical optimization algorithm** already presented in the logistic regression class that is a particular case of a more general optimization algorithm that we will see later on today.

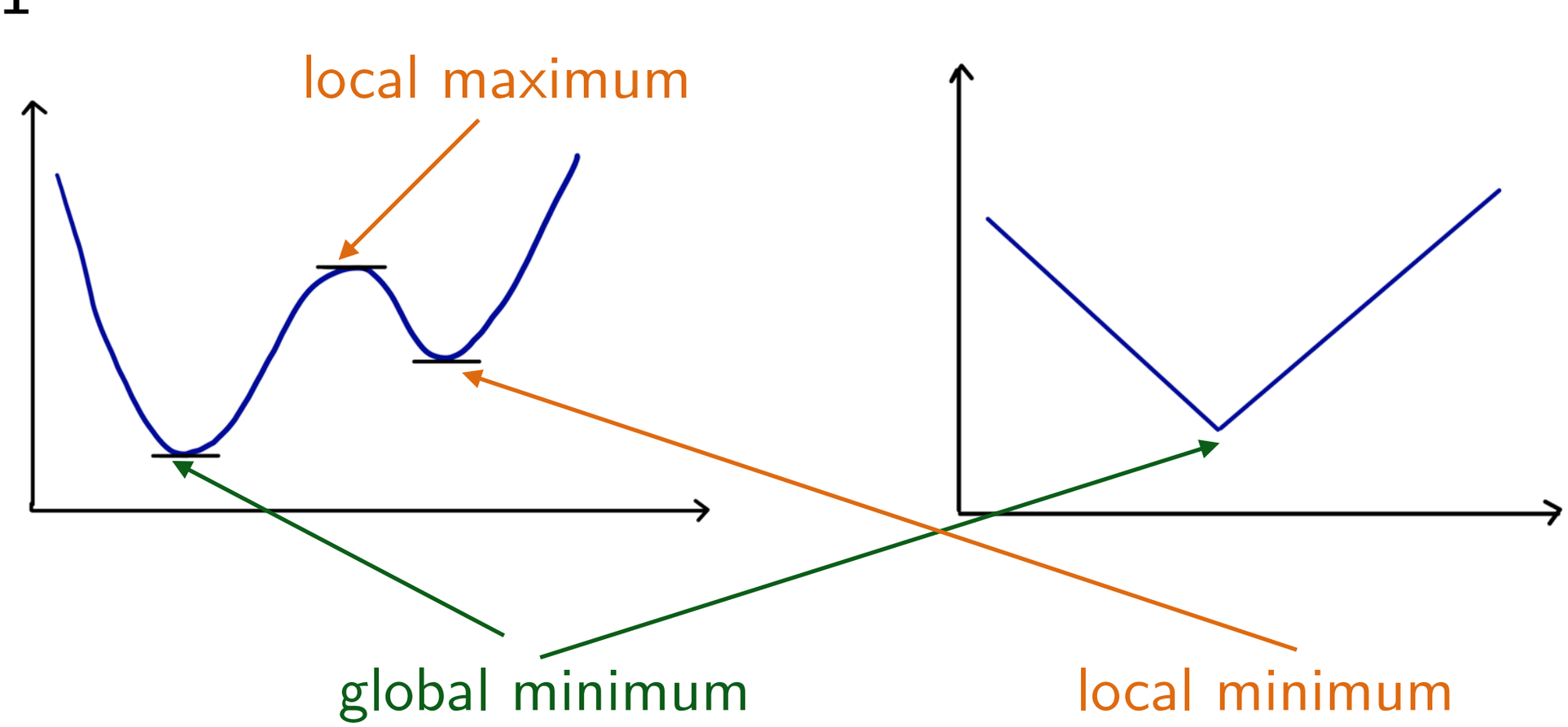
Numerical Optimization: General Framework

Unconstrained optimization: general setting

$$\text{minimize } f : x \in \Omega \subset \mathbb{R}^d \to f(x) \in \mathbb{R}$$

d: dimension of the search space

d=1

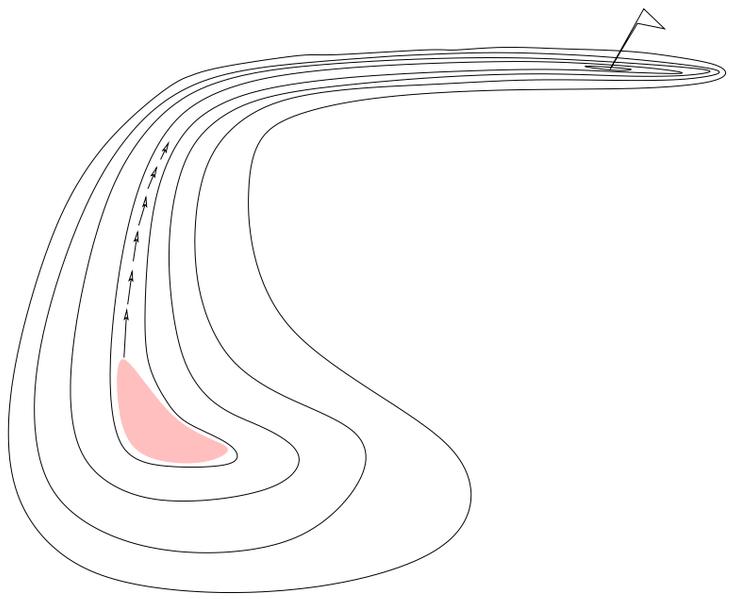


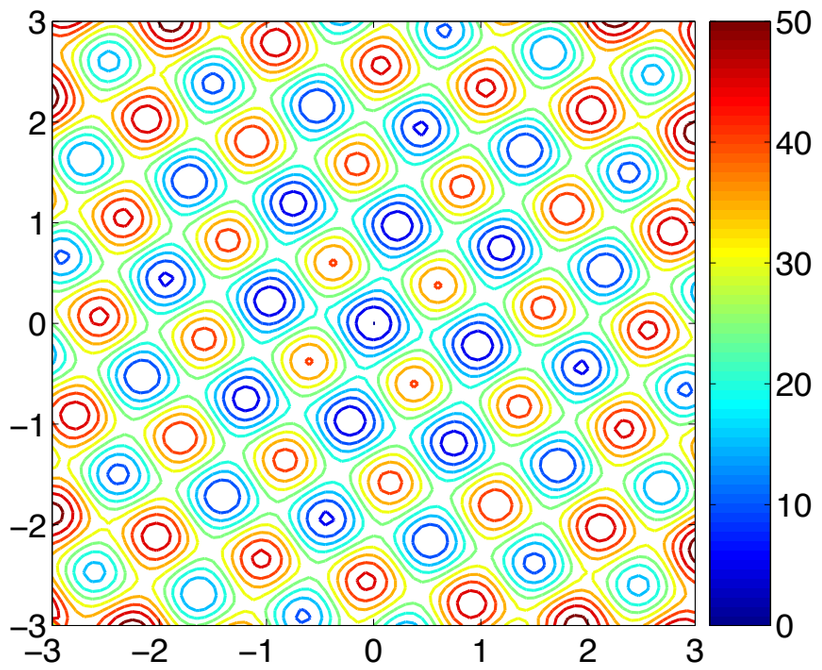
Level Sets: Visualization of a Function

One-dimensional (1-D) representations are often misleading (as 1-D optimization is "trivial", see slides related to curse of dimensionality), we therefore often represent level-sets of functions

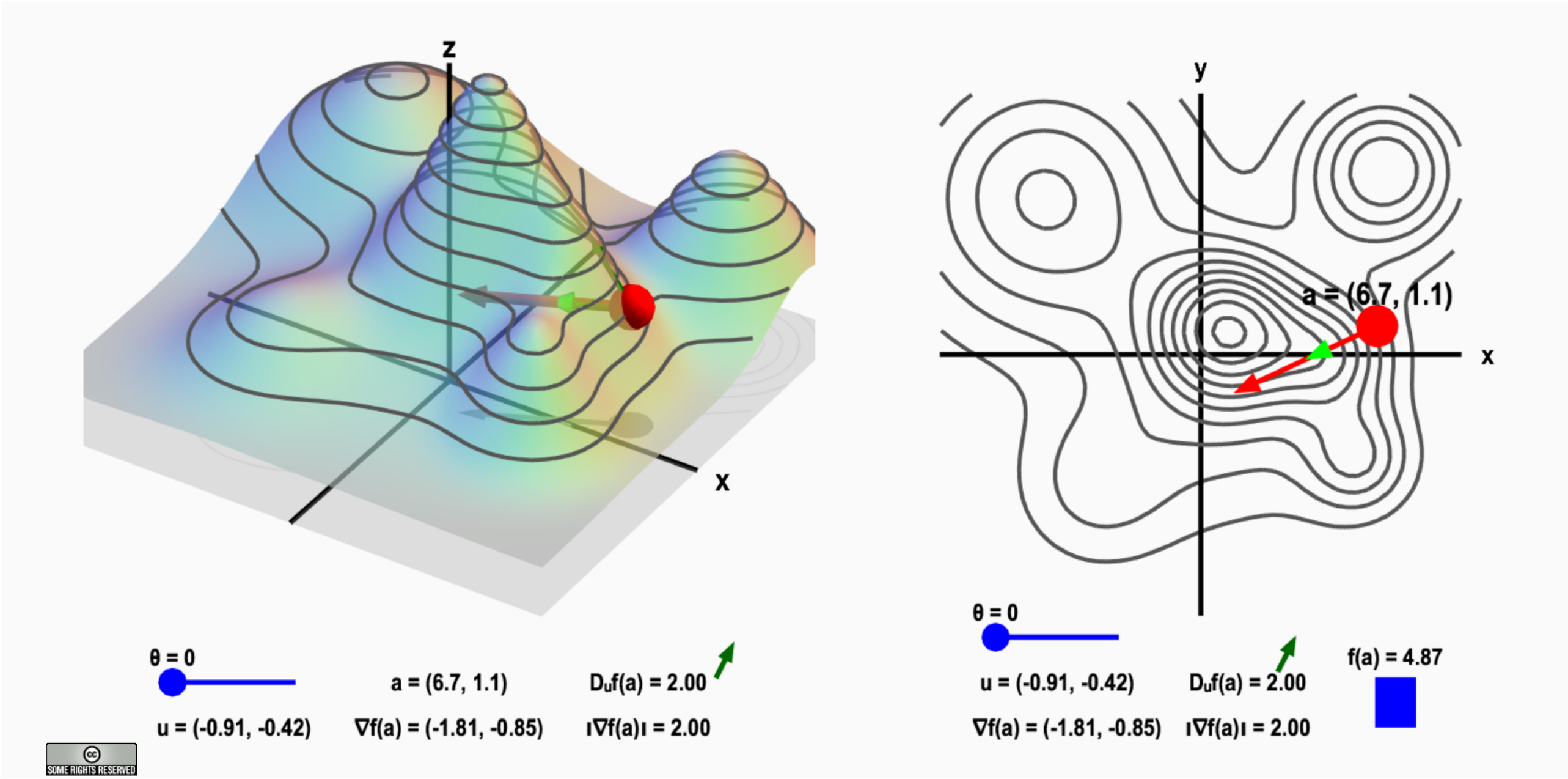
$$\mathcal{L}_c = \{x \in \mathbb{R}^d | f(x) = c\} \text{ for } c \in \mathbb{R}$$

Examples of level sets in 2D





Level Sets: Visualization of a Function

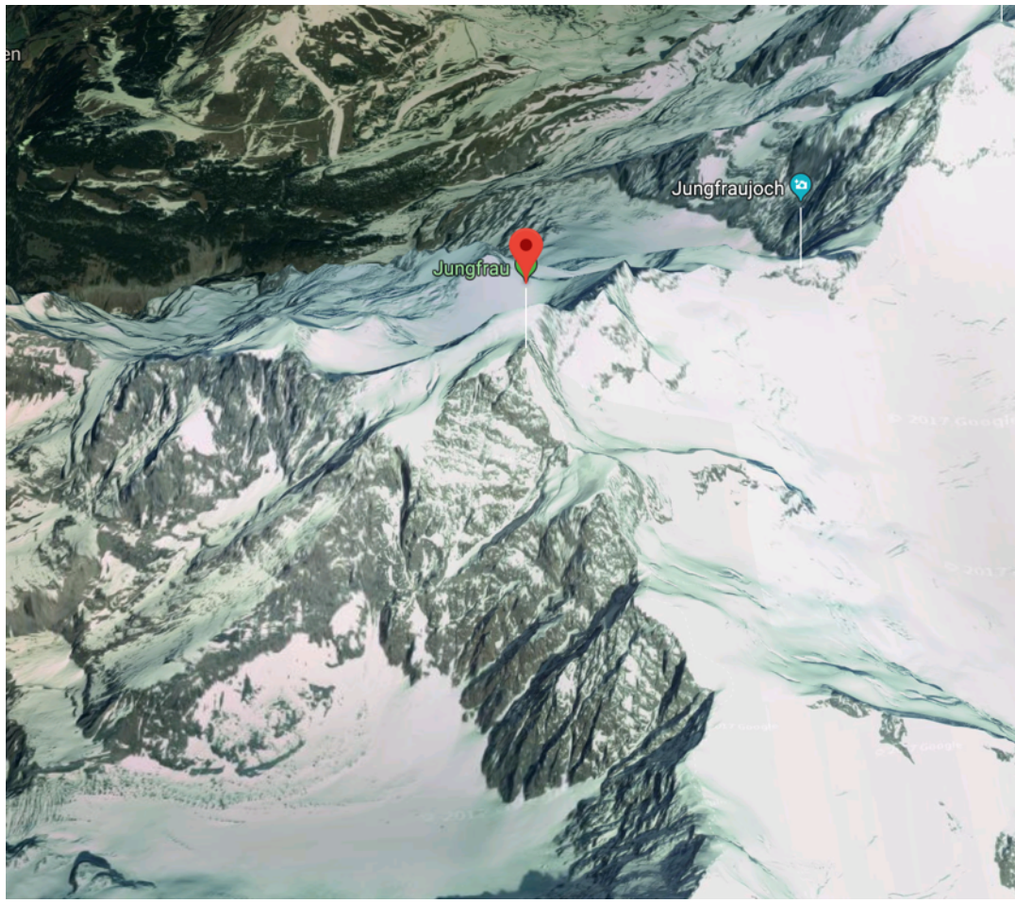


SOME RIGHTS RESERVED

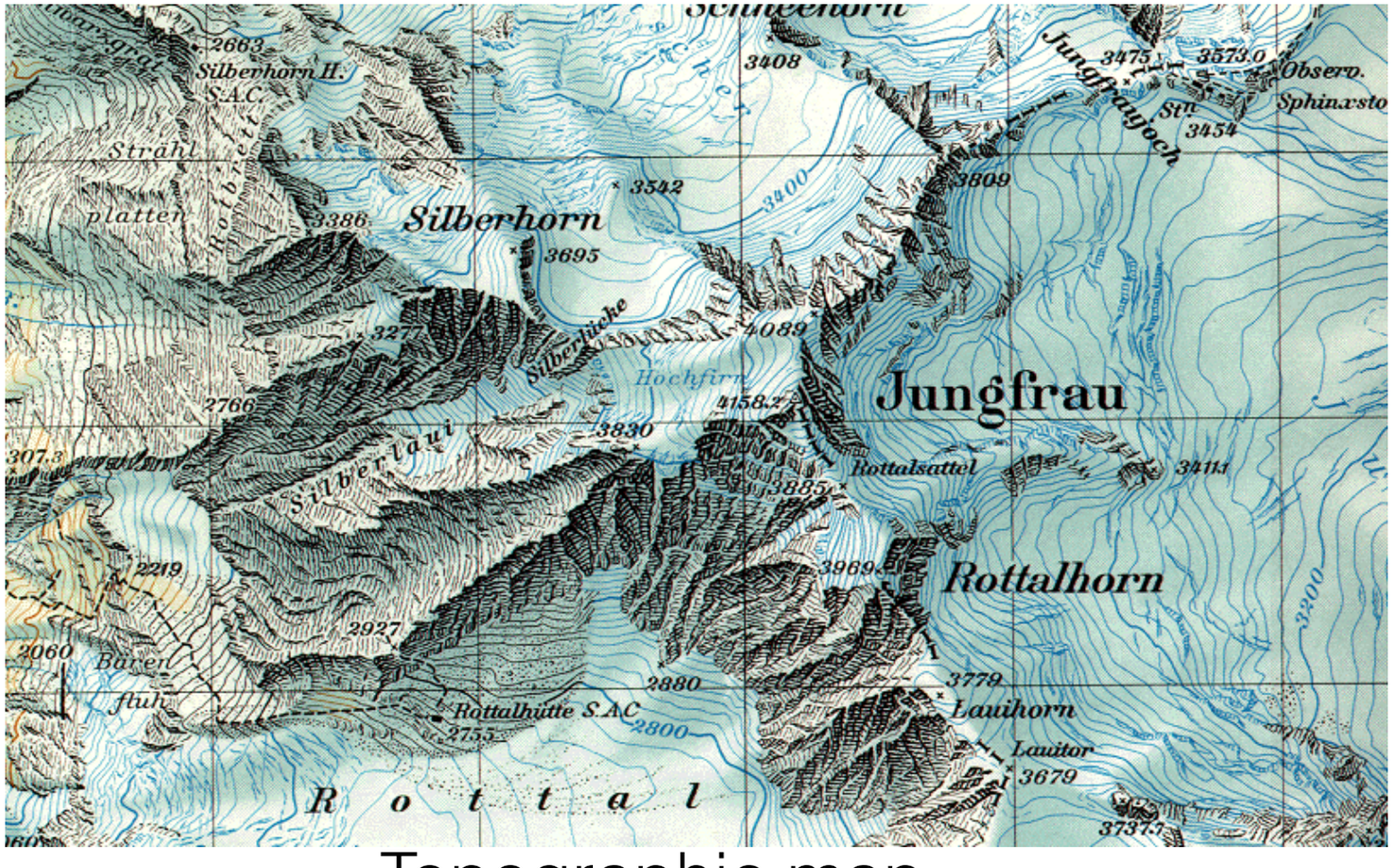
Source: Nykamp DQ, “Directional derivative on a mountain.” From *Math Insight*. http://mathinsight.org/applet/directional_derivative_mountain

Level Sets: Topographic Map

The function is the altitude



3-D picture



Topographic map

Level Set: Exercice

Consider a convex-quadratic function

$$f : x \mapsto \frac{1}{2}(x - x^*)^T H(x - x^*) = \frac{1}{2}\sum_i h_{i,i}\,(x_i - x_i^*)^2 + \frac{1}{2}\sum_{i \neq j} h_{i,j}\,(x_i - x_i^*)(x_j - x_j^*)$$

with H a symmetric, positive, definite matrix

1. Assume d=2, $H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ plot the level sets of f

2. Same question with $H = \begin{bmatrix} 1 & 0 \\ 0 & 9 \end{bmatrix}$

3. Same question with $H = P \begin{bmatrix} 1 & 0 \\ 0 & 9 \end{bmatrix} P^T$ with $P \in \mathbb{R}^{2\times 2}$

   P orthogonal

Numerical Optimization: General Framework

Unconstrained optimization: general setting

$$\text{minimize } f : x \in \Omega \subset \mathbb{R}^d \to f(x) \in \mathbb{R}$$

d: dimension of the search space

Remarks:

- we can assume minimization without loss of generality as maximizing f boils down to minimizing -f
- if $f : \Omega \to \mathbb{R}$ where Ω is discrete (e.g. $\Omega = \{0, 1\}^n$) and thus $|\Omega|$ is finite, we talk about discrete / combinatorial optimization

not covered in this class

Why is Optimization a non-trivial Problem?

Curse of dimensionality

if d=1, which simple approach could you use to minimize:

$$f : [0, 1] \to \mathbb{R} \quad ?$$

Why is Optimization a non-trivial Problem?

Curse of dimensionality

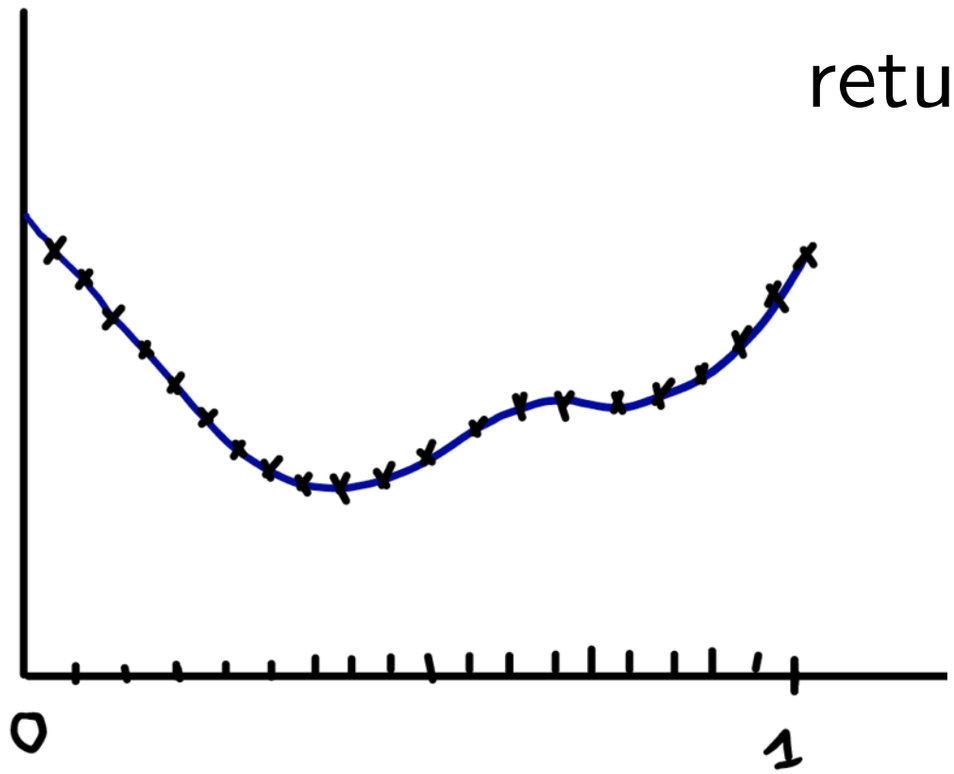
if d=1, which simple approach could you use to minimize:

$$f : [0, 1] \to \mathbb{R} \quad ?$$

set a regular grid on [0,1]

evaluate on f all the points of the grid

return the lowest function value



Why is Optimization a non-trivial Problem?

Curse of dimensionality

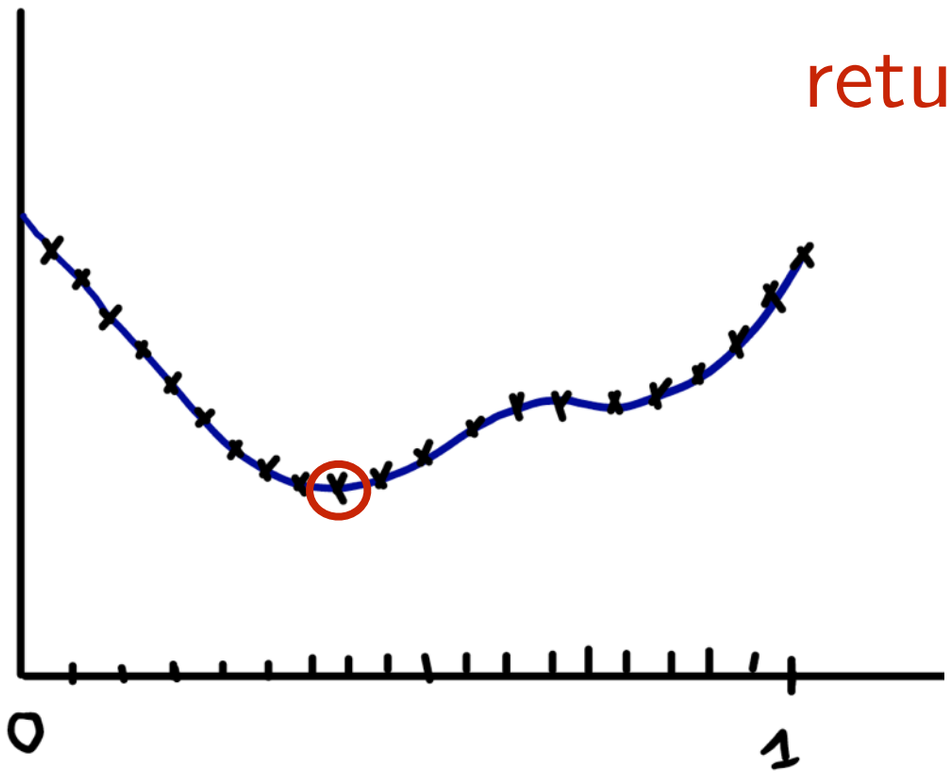
if d=1, which simple approach could you use to minimize:

$$f : [0, 1] \to \mathbb{R} \quad ?$$

set a regular grid on [0,1]

evaluate on f all the points of the grid

return the lowest function value



Why is Optimization a non-trivial Problem?

Curse of dimensionality

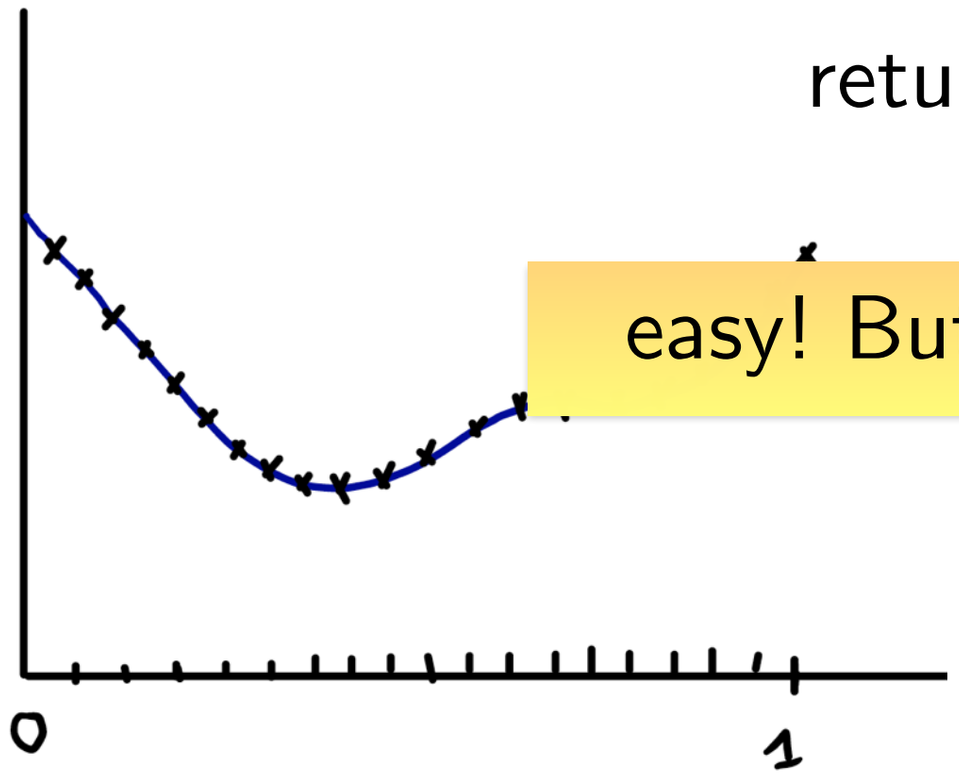
if d=1, which simple approach could you use to minimize:

$$f : [0, 1] \rightarrow \mathbb{R} \quad ?$$

set a regular grid on [0,1]

evaluate on f all the points of the grid

return the lowest function value



easy! But how does it scale when d increases?

1-D optimization is trivial

Curse of Dimensionality

The term curse of dimensionality (Richard Bellman) refers to problems caused by the rapid increase in volume associated with adding extra dimensions to a (mathematical) space.

Example: Consider placing 100 points onto a real interval, say [0,1].

How many points would you need to get a similar coverage (in terms of distance between adjacent points) in dimension 10?

Curse of Dimensionality

The term curse of dimensionality (Richard Bellman) refers to problems caused by the rapid increase in volume associated with adding extra dimensions to a (mathematical) space.

Example: Consider placing 100 points onto a real interval, say [0,1]. To get similar coverage, in terms of distance between adjacent points, of the 10-dimensional space $[0,1]^{10}$ would require $100^{10} = 10^{20}$ points. A 100 points appear now as isolated points in a vast empty space.

Consequence: a search policy (e.g. exhaustive search) that is valuable in small dimensions might be useless in moderate or large dimensional search spaces.

Curse of Dimensionality

How long would it take to evaluate 10^{20} points ?

```
import timeit
timeit.timeit('import numpy as np ;
np.sum(np.ones(10)*np.ones(10))', number=1000000)
> 7.0521080493927
```

7 seconds for 10^6 evaluations of $f(x) = \sum_{i=1}^{10} x_i^2$

We would need more than 10^8 days for evaluating 10^{20} points

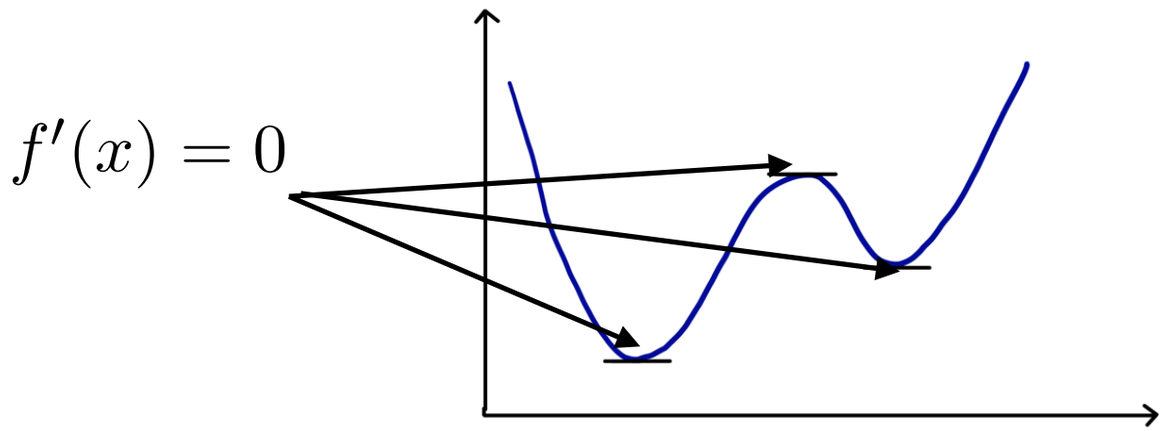
[As a reference: origin of human species: roughly 6 x 10^8 days]

Basics: Optimality Conditions

First order **necessary** conditions

Start with d=1 (what you should already know)

Let $f : \mathbb{R} \to \mathbb{R}$ be a derivable function, if x^* is a local extremum then $f'(x) = 0$



Generalization for d>1

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a derivable/differentiable function, if x^* is a local extremum of f, then $\nabla f(x^*) = 0$

Remarks: Points where $\nabla f(x^*) = 0$ are called critical points those are necessary but not sufficient conditions (consider $f(x) = x^3$)

Gradient: Reminder

The gradient of a function $f : \mathbb{R}^d \to \mathbb{R}$ in x^* denoted as $\nabla f(x^*)$ is the vector of partial derivatives

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_d} \end{pmatrix}$$

Exercice:

in $\mathbb{R}$, $\nabla f(x) = f'(x)$

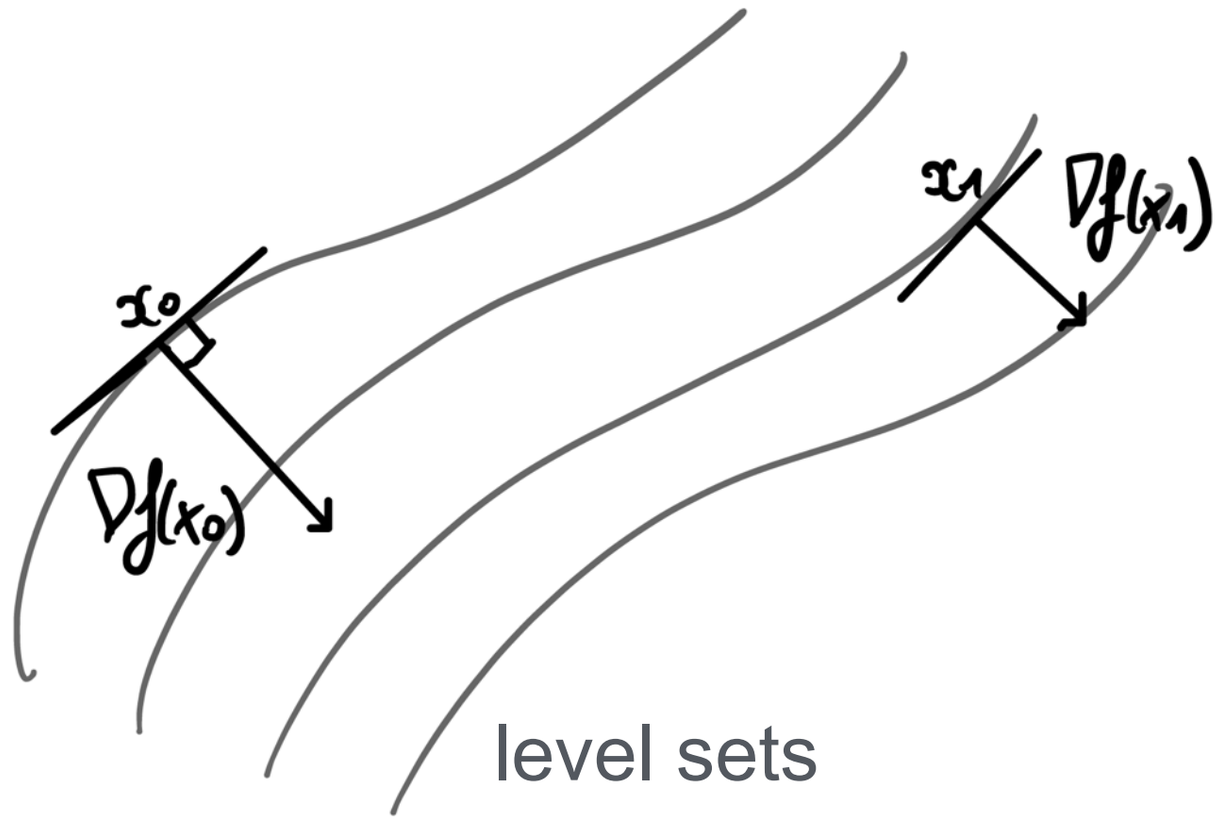
$f(x) = \langle a, x \rangle$: $\nabla f(x) = a$

$f(x) = x^T A x$: $\nabla f(x) = (A + A^T)x$

particular case $f(x) = \|x\|^2$: $\nabla f(x) = 2x$

Gradient: Geometric Interpretation

The gradient is orthogonal to the level sets

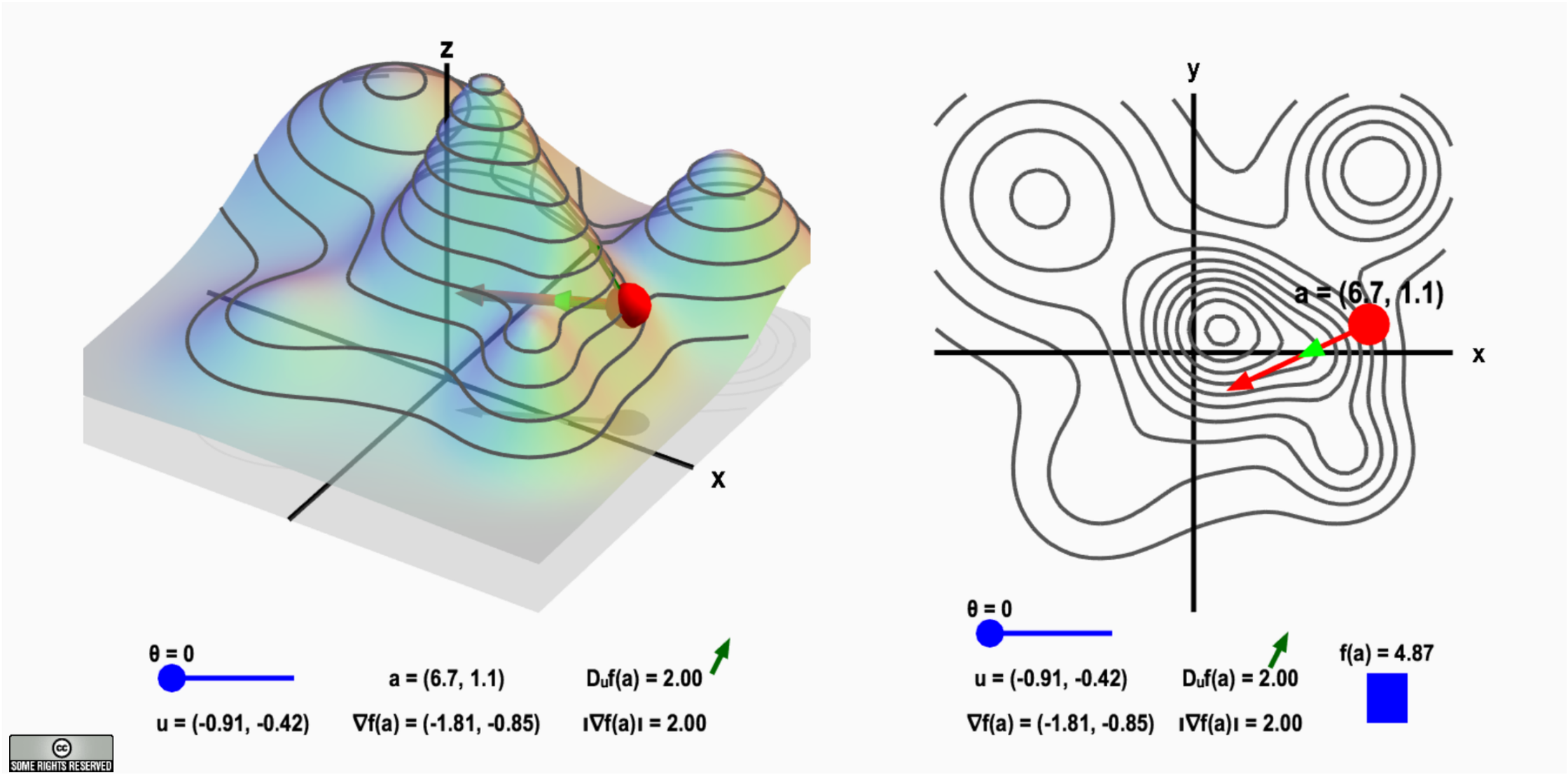


Exercice:

1. Compute and plot the gradient on the convex quadratic functions of the level-set exercice.
2. Verify the property that the gradient is orthogonal to the level sets

Gradient: Geometric Interpretation

The gradient is orthogonal to the level sets



SOME RIGHTS RESERVED

Source: Nykamp DQ, “Directional derivative on a mountain.” From *Math Insight*. http://mathinsight.org/applet/directional_derivative_mountain

Optimality Conditions: Second Order

Second order necessary and sufficient conditions

Assume f is twice continuously differentiable (C^2)

Necessary condition

if x^* is a local minumum, then $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive semi-definite

Sufficient condition

if $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite, then x^* is a strict local optimum

Remark: For d=1 this condition boils down to $f(x^*) = 0$ and $f''(x^*) > 0$

Hessian Matrix : Reminder

If f is twice differentiable, the Hessian matrix denoted $\nabla^2 f$ is given by

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

The Hessian matrix is symmetric if f is twice continuously differentiable (C^2)

First and Second Order Approximations

First order approximation of f

If f is derivable, a first order approximation of f in x_0 given by

$$f(x_0 + h) = f(x_0) + \nabla f(x_0)^T h + o(\|h\|)$$

first order Taylor expansion

If h is small enough $f(x_0 + h) \approx f(x_0) + \nabla f(x_0)^T h$

linear function in h

First and Second Order Approximations

Second order approximation of f

If f is twice continuously differentiable, a second order approximation of f in x_0 given by

$$f(x_0 + h) = f(x_0) + \nabla f(x_0)^T h + \frac{1}{2} h^T \nabla^2 f(x_0) h + o(\|h\|^2)$$

second order Taylor expansion

If h is small enough $f(x_0 + h) \approx f(x_0) + \nabla f(x_0)^T h + \frac{1}{2} h^T \nabla^2 f(x_0) h$

quadratic function in h

First and Second Order Approximations

Second order approximation of f

If f is twice continuously differentiable, a second order approximation of f in x_0 given by

$$f(x_0 + h) = f(x_0) + \nabla f(x_0)^T h + \frac{1}{2} h^T \nabla^2 f(x_0) h + o(\|h\|^2)$$

second order Taylor expansion

If h is small enough $f(x_0 + h) \approx f(x_0) + \nabla f(x_0)^T h + \frac{1}{2} h^T \nabla^2 f(x_0) h$

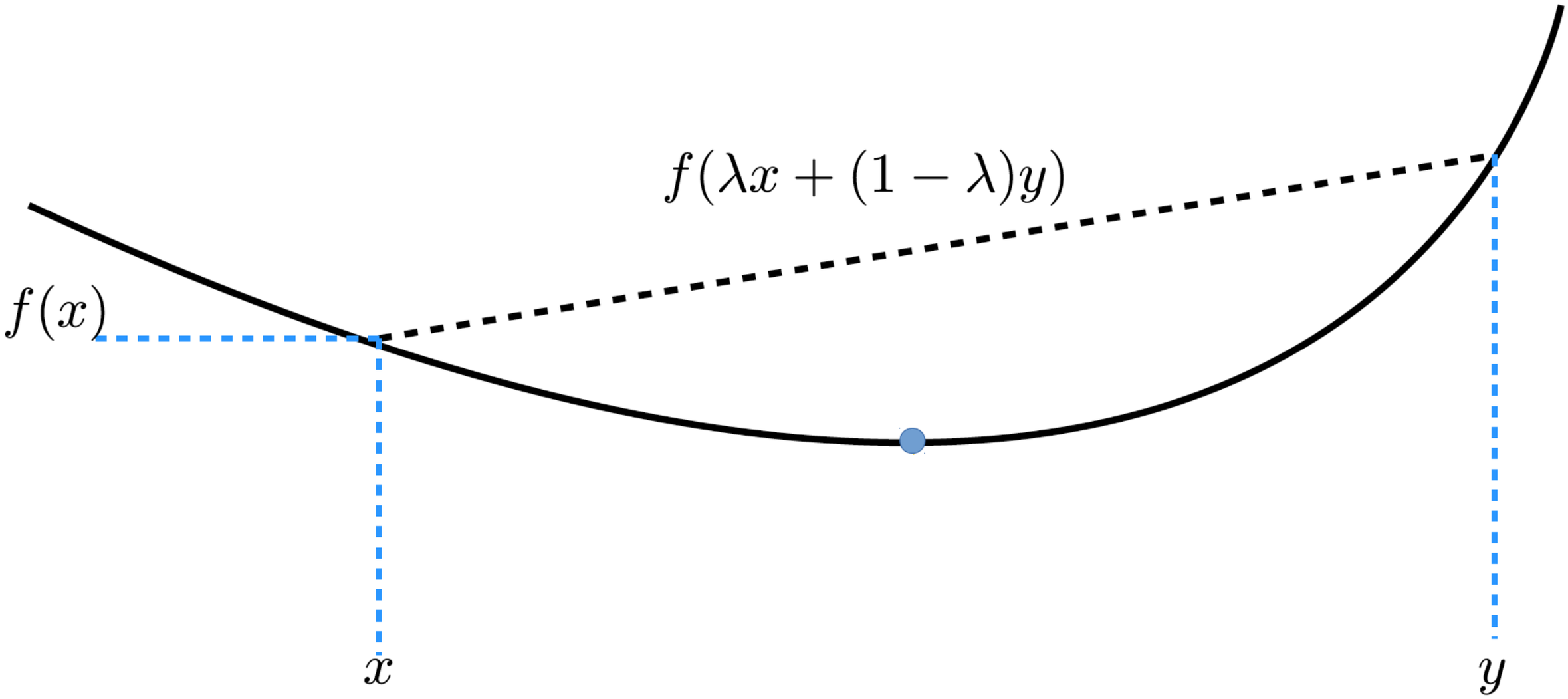
quadratic function in h

will serve as motivation for Newton's method

Convexity

We say that $f : \mathrm{dom}(f) \subset \mathbb{R}^n \to \mathbb{R}$ is convex if $\mathrm{dom}(f)$ is convex and

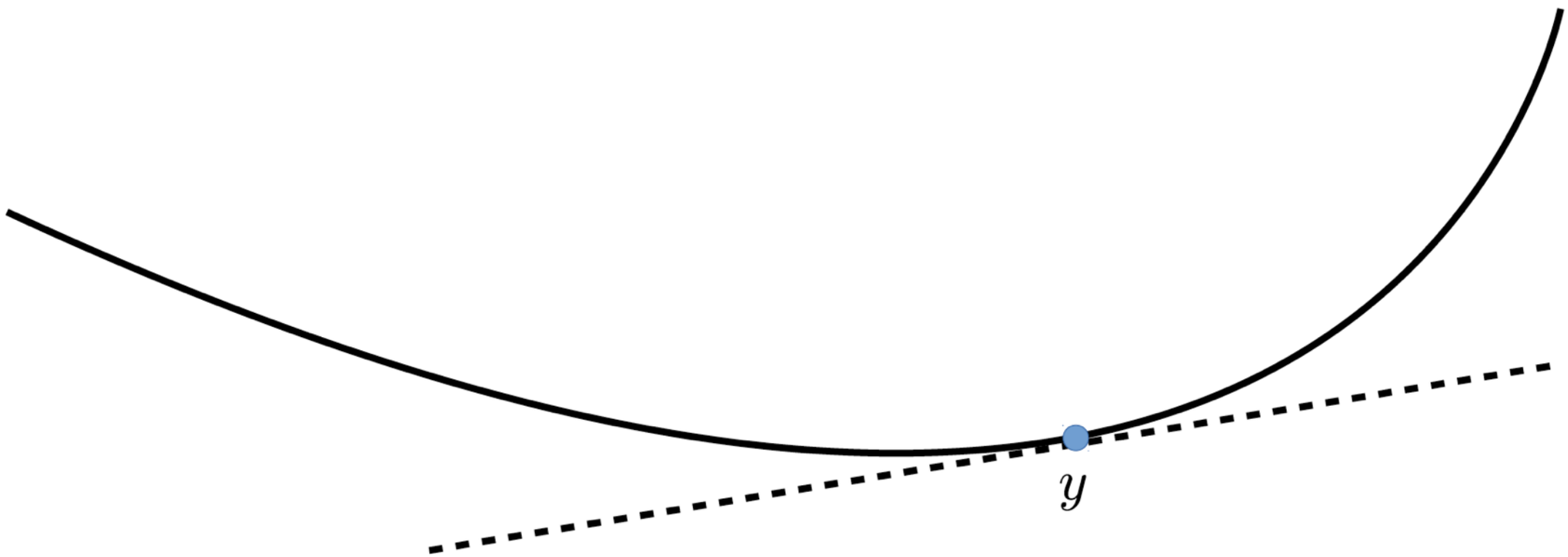
$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y), \ \forall x, y \in \mathrm{dom}(f), \lambda \in [0,1]$$



Convexity: First Derivative

A differentiable function $f : \mathrm{dom}(f) \subset \mathbb{R}^n \to \mathbb{R}$ is convex iff

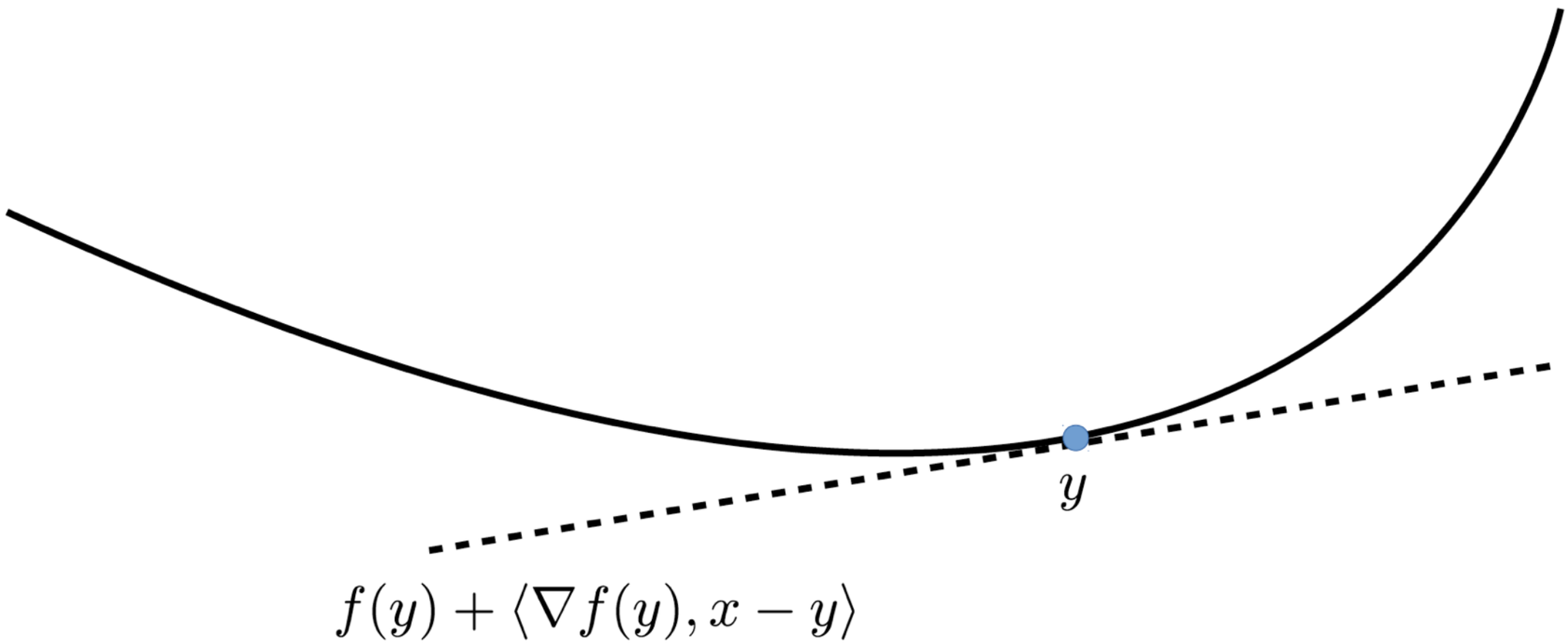
$$f(x) \geq \lambda f(y) + \langle \nabla f(y), x - y \rangle, \ \forall x, y \in \mathrm{dom}(f)$$



Convexity: First Derivative

A differentiable function $f : \mathrm{dom}(f) \subset \mathbb{R}^n \to \mathbb{R}$ is convex iff

$$f(x) \geq \lambda f(y) + \langle \nabla f(y), x - y \rangle, \ \forall x, y \in \mathrm{dom}(f)$$



Consequence:

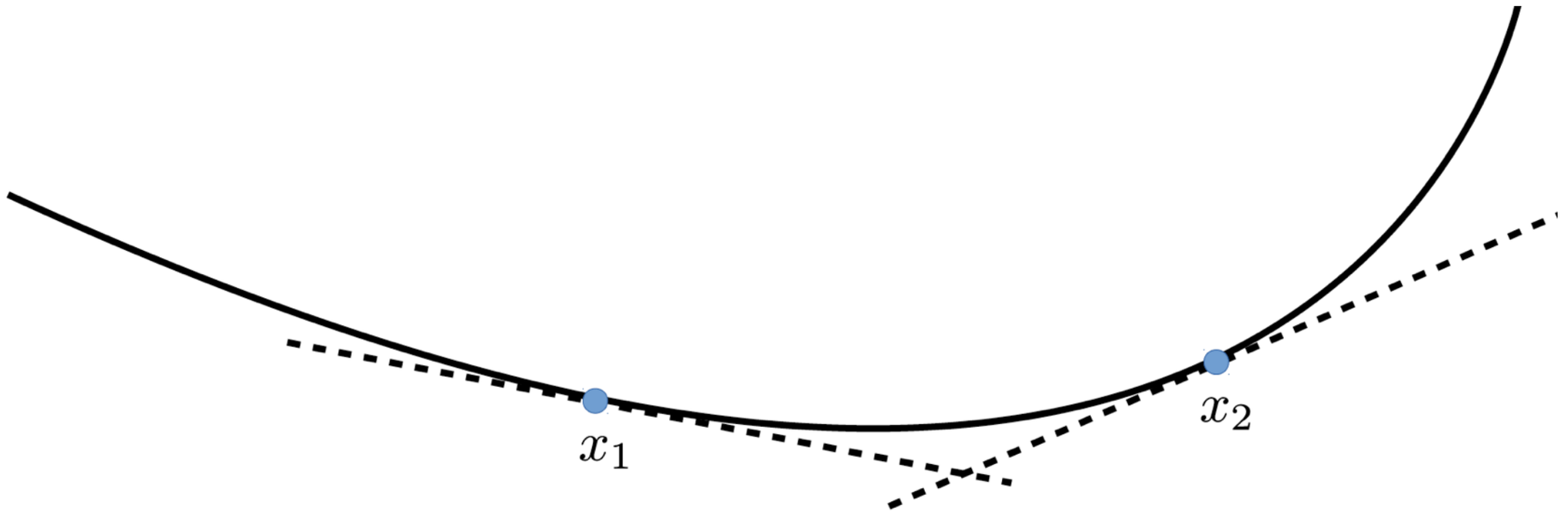
Assume that $f : \mathrm{dom}(f) \subset \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable.

If $\nabla f(x^*) = 0$, then x^* is a global minimum of f.

Convexity: Second Derivative

A twice differentiable function $f : \mathrm{dom}(f) \subset \mathbb{R}^n \to \mathbb{R}$ is convex iff

$$\nabla^2 f(x) \succeq 0 \text{ for all } x \qquad (\text{that is } h^T \nabla^2 f(x) h \geq 0 \text{ for all h})$$



$$x_1 \leq x_2 \quad \Rightarrow f'(x_1) \leq f'(x_2)$$

Convexity: Examples

Norms and squared norms: $x \mapsto ||x||$

$x \mapsto ||x||^2$

Negative log and logisitc: $x \mapsto -\log(x)$

$x \mapsto \log\left(1 + e^{-y\langle a, x\rangle}\right)$

Hinge loss $x \mapsto \max\{0, 1 - yx\}$

Application - Linear Regression

$$f(w) := \|Xw - y\|^2 \qquad X \in \mathbb{R}^{n \times d}$$

Minimizing f:

$$f(w) \text{ is convex}$$

$$\nabla f(w_{\text{opt}}) = 0 \Leftrightarrow \; w_{\text{opt}} = (X^\top X)^{-1} X^\top y$$

if $X^\top X$ invertible

Numerical Optimization

Very often it is not possible to solve analytically the equation $\nabla f(x) = 0$ and we have to resort to an iterative algorithm (or numerical optimization algorithm) that will generate a sequence of points $\{x_k : k \geq 0\}$ that should converge to $\operatorname{argmin}_x f(x)$

Optimization algorithm:

input $f, \nabla f, (\nabla^2 f)$

initialize $k = 0,\ x_0$ [other state variables]

while not happy do

  update x_k

  $k = k + 1$

end-do

return $x_k,\ k$

$f(x_{k+1}) \leq f(x_k)$ (typically)

Goal:

$\lim_{k\to\infty} f(x_k) = \min_x f(x)$

$\lim_{k\to\infty} \|x_k - x^*\| = 0$

Algorithm Classes

Depending on the information the algorithm is using to create a new point (or iterate) we distinguish

Zero-order's algorithms: only use f (no gradients, ...). Those methods are also called derivative-free optimization algorithms. Used when gradient or Hessian are difficult to compute, or when the functions are not differentiable.

First-order algorithms: use f and ∇f. Standard algorithms when f is differentiable, convex.

Second-order algorithms: use $f, \nabla f$ and $\nabla^2 f$. When we can have an "easy" access to the Hessian matrix.

Algorithm Classes

Depending on the information the algorithm is using to create a new point (or iterate) we distinguish

Zero-order's algorithms: only use f (no gradients, ...). Those methods are also called derivative-free optimization algorithms. Used when gradient or Hessian are difficult to compute, or when the functions

Today's lecture

First-order algorithms: use f and ∇f. Standard algorithms when f is differentiable, convex.

Second-order algorithms: use f, ∇f and $\nabla^2 f$. When we can have an "easy" access to the Hessian matrix.

Descent Algorithm

descent direction

Illustration idea from “Alexander & Michael Bronstein” Numerical Optimization slides

Descent Algorithm

Generic algorithm:

choose an initial point x_0, $k = 0$

while not happy

choose a descent direction d_k

line-search: choose a step-size σ_k

$$x_{k+1} = x_k + \sigma_k d_k$$

$$k = k + 1$$

Line search: 1-d minimization along the descent direction

$$\sigma \to f(x_k + \sigma d_k)$$

Descent direction: direction such that for σ small enough

$$f(x_k + \sigma d_k) < f(x_k)$$

Stopping criteria

When are we “happy”, i.e. when do we stop the algorithm?

- when gradient norm becomes small

$$\|\nabla f(x_k)\| \leq \epsilon$$

- when step-size becomes small

$$\|x_{k+1} - x_k\| \leq \epsilon$$

- when progress in f becomes small

$$\frac{|f(x_{k+1}) - f(x_k)|}{|f(x_k)|} \leq \epsilon$$

Gradient Descent

We take as descent direction minus the gradient

$$d_k = -\nabla f(x_k)$$

Why is it a descent direction? for f differentiable according to Taylor's expansion:

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

Gradient descent: follows the direction orthogonal to the level sets

Line Search

We need a large enough step-size to converge quickly

Armijo-Goldstein Rule or Backtracking Line Search

starts with a (too) large estimate of σ and reduces it (i.e. backtracks) until f is reduced enough

what is enough?

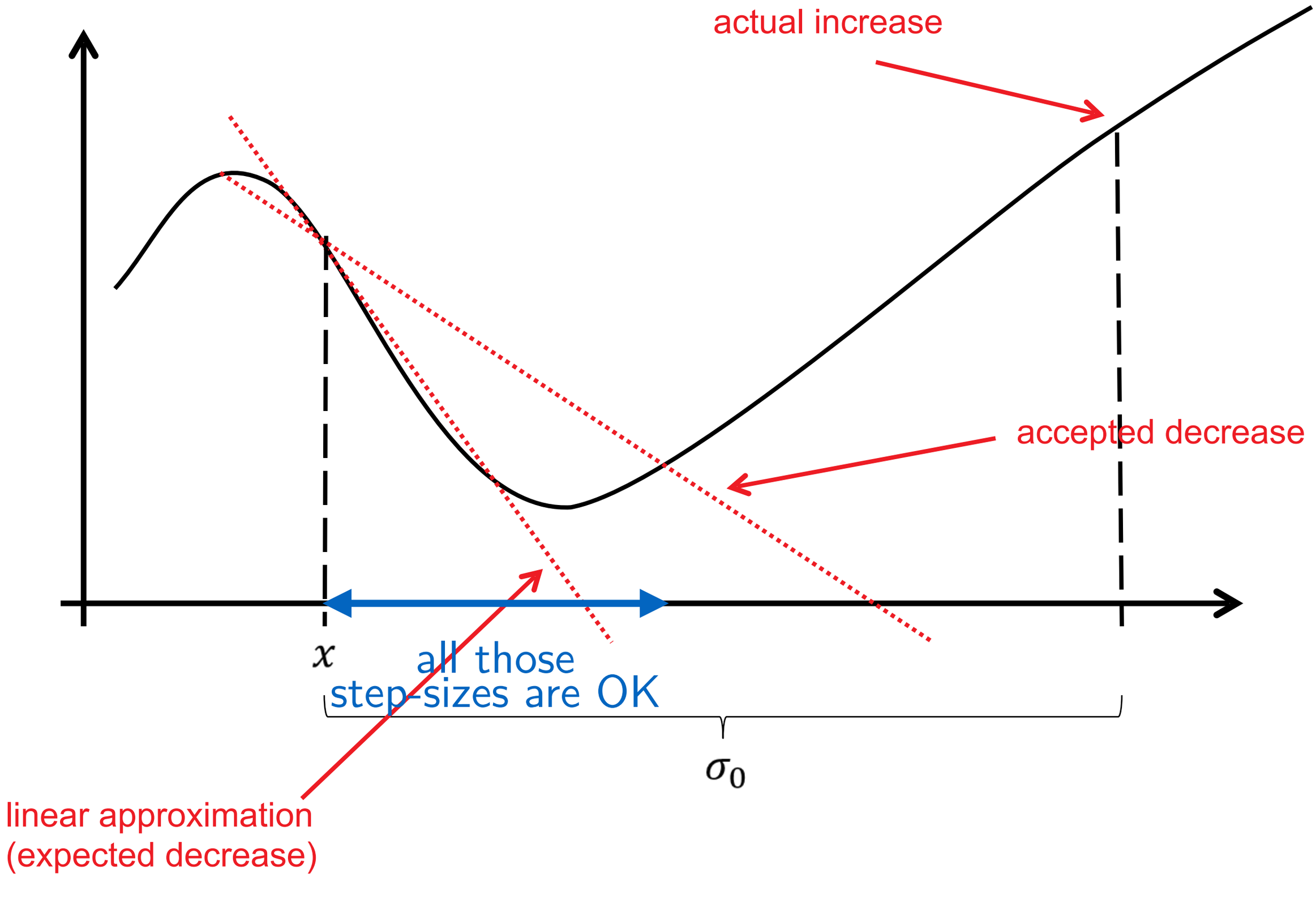
assume a linear f, e.g. $m_k(x) = f(x_k) + \nabla f(x_k)^\top (x - x_k)$

expected decrease if step of σ_k is done in direction d: $\sigma_k \nabla f(x_k)^\top d$

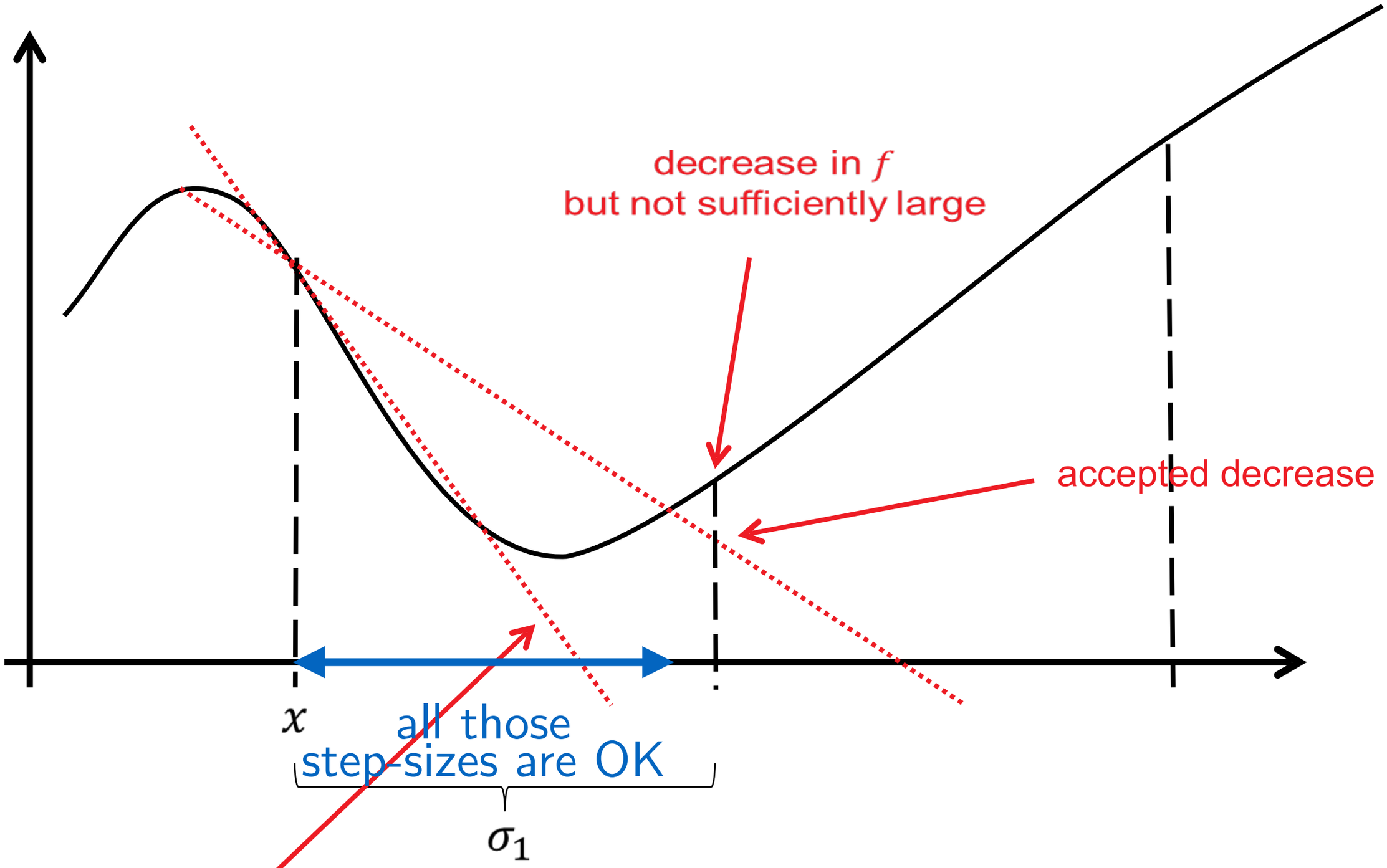
actual decrease: $f(x_k) - f(x_k + \sigma_k d)$

stop if actual decrease is at least constant times expected decrease

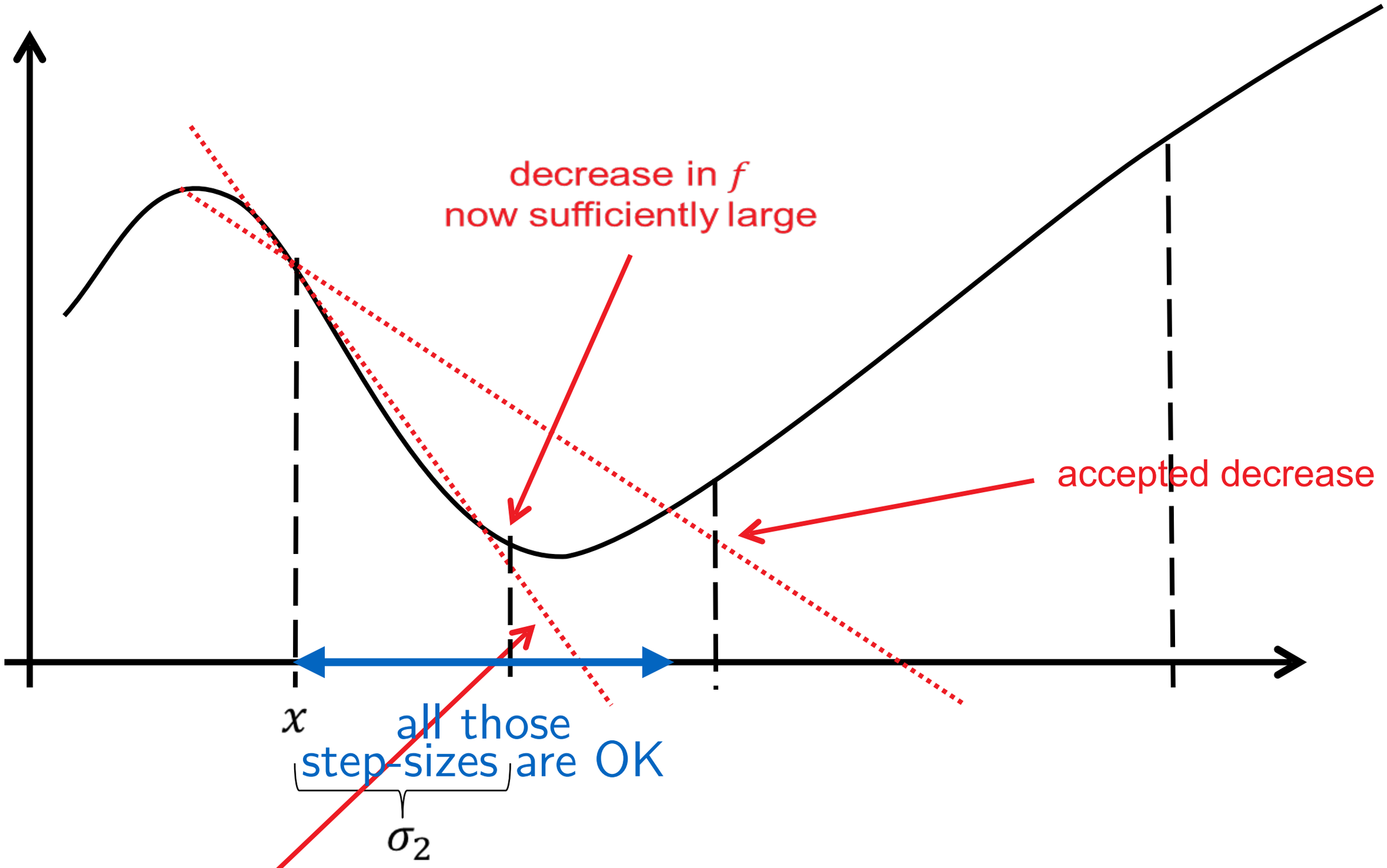
The Armijo-Goldstein Rule



The Armijo-Goldstein Rule



The Armijo-Goldstein Rule



The Armijo-Goldstein Rule

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 σ: $\sigma \leftarrow \sigma_0$
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

Is minus gradient direction a good direction to follow?

Ill-conditioned Problems

We often encounter ill-conditioned problems where different combinations of variables have completely different scalings.

Ill-conditioned convex-quadratic problems:

If f is convex-quadratic, i.e.

$$f : x \mapsto \frac{1}{2} x^\top H x$$

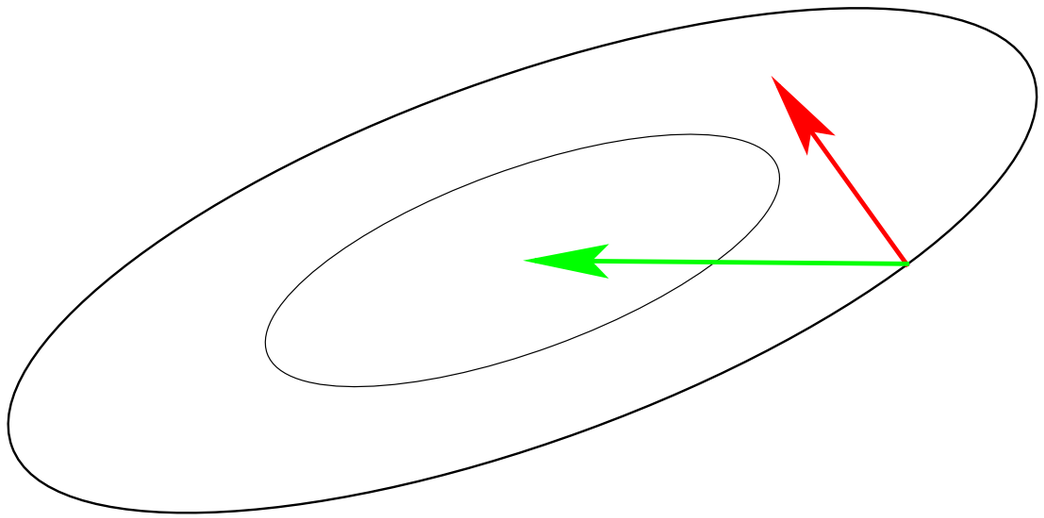
with H a symmetric, definite positive matrix (the Hessian of f)

ill-conditioned means a high condition number of the Hessian matrix *H*

$$\mathrm{cond}(H) = \frac{\lambda_{\max}(H)}{\lambda_{\min}(H)}$$

Ill-conditioned Problems (II)

Geometric interpretation: An ill-conditioned convex-quadratic problem will have a large ratio between the largest and smallest axis of the ellipsoid level set:

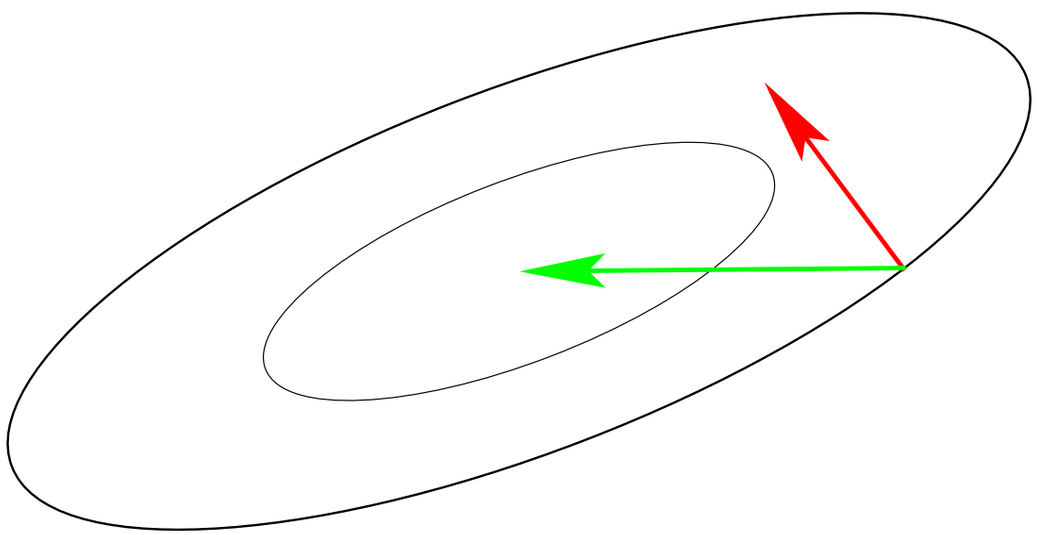


level-sets of a convex-quadratic function

Condition number equals nine here (so the problem is actually not ill-conditioned). Condition numbers **up to 10^{10}** are not unusual in real world problems.

For a function which is not convex-quadratic, we talk about an ill-conditioned problem if the level sets are "squeezed".

Gradient Descent on Ill-conditioned Problems



On ill-conditioned problem, the gradient descent algorithm is slow ...

A better descent direction is given by

$$-H^{-1}\nabla f(x)$$

Newton's Algorithm

Take as descent direction the Newton step:

$$d_k = -[\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$$

The Newton's direction minimizes the best locally quadratic approximation of f. Indeed, by Taylor's expansion we can approximate f locally in x by

$$\begin{aligned} g(h) &= f(x) + \nabla f(x)^\top h + \tfrac{1}{2} h^\top \nabla^2 f(x) h \\ &\approx f(x+h) \end{aligned}$$

Minimizing g with respect to h yields:

$$h = -[\nabla^2 f(x)]^{-1} \nabla f(x)$$

Newton's Algorithm for Logistic Regression

In the case of logistic regression, we can have an explicit expression for the Newton's step and Newton's algorithm is the Iterative Reweighted Least Squares (IRWLS) that was presented in the logistic regression class.

Yet, what can we do if we do not have access to the Hessian matrix?

Quasi-Newton's Methods

In quasi-Newton's methods, the Newton direction is approximated by using solely first order information (gradient)

Key idea: successive iterates x_k, x_{k+1} and gradients $\nabla f(x_k)$ yield second order information

$$q_k \approx \nabla^2 f(x_{k+1}) p_k$$

$$p_k = x_{k+1} - x_k,\ q_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm

BFGS algorithm:

B_k approximation of Hessian matrix

$$d_k = -B_k^{-1} \nabla f(x_k)$$
$$x_{k+1} = x_k + \sigma_k d_k \text{ (find } \sigma_k \text{ via line-search)}$$
$$y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$
$$B_{k+1} = B_k + \frac{y_k y_k^\top}{y_k^\top \sigma_k d_k} - \frac{B_k d_k d_k^\top B_k}{d_k^\top B_k d_k}$$

efficient update to compute the inverse of B_k

Considered as the state-of-the-art quasi-Newton's algorithm.
Implemented in all (good) optimization toolboxes

Gradient Descent - Simple Theoretical Analysis

Theorem[Linear convergence of gradient descent] Assume $f : \mathbb{R}^d \to \mathbb{R}$ is twice continuously differentiable, convex and for all x, $\mu I_d \preccurlyeq \nabla^2 f(x) \preccurlyeq L I_d$ with $\mu > 0$. Let x^* be the unique global minimum of f. The gradient descent algorithm with fixed step-size $\sigma_t = \frac{1}{L}$ satisfies

$$\|x_{k+1} - x^*\|^2 \leq \left(1 - \frac{\mu}{L}\right) \|x_k - x^*\|^2 \ .$$

That is the algorithm converges geometrically (also called linearly):

$$\|x_k - x^*\|^2 \leq \left(1 - \frac{\mu}{L}\right)^k \|x_0 - x^*\|^2$$

algorithm slower and slower with increasing condition number

In comparison, convergence of Newton's method is quadratic:

$$\|x_{k+1} - x^*\| \leq c \|x_k - x^*\|^2 \text{ with } c < 1$$

$$\|x_{k+1} - x^*\|^2 \leq c^2 \left(\|x_k - x^*\|^2\right)^2 \text{ with } c < 1$$

Gradient Descent - Simple Theoretical Analysis

Remarks:

$A \preccurlyeq B$ means $x^T Ax \leq x^T Bx$ for all x

For f twice continuous differentiable $\mu I_d \preccurlyeq \nabla^2 f(x)$ is called μ-strong convexity

a strongly convex function does not need to be twice continuously differentiable (it is assumed for the sake of simplicity)

$\mu I_d \preccurlyeq \nabla^2 f(x) \preccurlyeq L I_d$ is equivalent to the eigenvalues of the Hessian of f are in between mu and L

Stochastic Gradient - Motivation

We now come back to our training optimization problem

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \underbrace{l(h_w(x^i), y^i)}_{f_i(w)}$$

the f_i can include a regularization term

Gradient descent update:

$$w_{k+1} = w_k - \sigma_k \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w_k)$$

Problem: each iteration requires to compute a gradient $\nabla f_i(w)$ for each data point. We don't want to do that when n is large (quite typical).

Stochastic Gradient

The gradient of $f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$ is approximated by the gradient of a single data function $f_i(w)$ at each iteration

$$\nabla f(w) \approx \nabla f_i(w) \text{ for } j \text{ chosen at random}$$

Stochastic gradient descent update:

$$\text{sample } j \in \{1, \ldots, n\}$$
$$w_{k+1} = w_k - \sigma_k \nabla f_i(w_k)$$

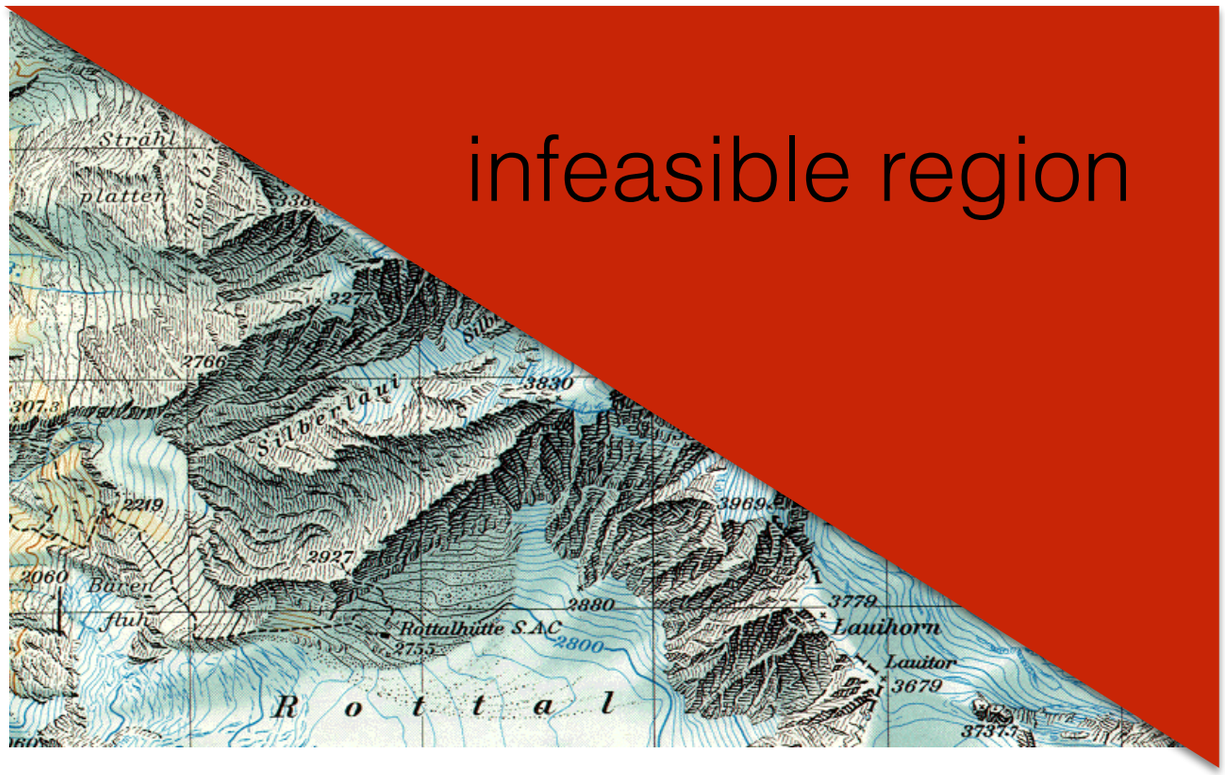
Constrained Optimization

Constrained optimization: general setting

$$\text{minimize } f : x \in \Omega \subset \mathbb{R}^d \to f(x) \in \mathbb{R}$$

$$g_i(x) \leq 0 \text{ with } g_i : \mathbb{R}^d \to \mathbb{R},\ i = 1, \ldots, m$$

m constraints



Not covered but important (SVM, ...)