

## Lecture 3

# Algorithms for unconstrained nonlinear optimisation. First- and second-order methods

Analysis and Development of Algorithms



УНИВЕРСИТЕТ ИТМО

# Overview

- 1 Terms
- 2 Gradient descent
- 3 (Nonlinear) Conjugate Gradient method
- 4 Newton's method
- 5 Levenberg-Marquardt algorithm

## Problem

$f : \mathbb{R}^n \rightarrow \mathbb{R}$  is convex;  $f = f(\mathbf{x})$ , where  $\mathbf{x} = (x_1, \dots, x_n)^T$  is a column-vector  
To solve the optimisation problem  $f(\mathbf{x}) \rightarrow \min_{\mathbf{x} \in Q}$  means to find  $\mathbf{x}^* \in Q$ , where  $Q$  is the region of acceptability, such that  $f$  reaches a minimal value at  $\mathbf{x}^*$ .  
Notation:  $\mathbf{x}^* = \arg \min_{\mathbf{x} \in Q} f(\mathbf{x})$ .

**Remark.** The approximation error is  $\varepsilon > 0$ . In the iterative algorithms below, we stop if  $\|\mathbf{a}_n - \mathbf{a}_{n-1}\| < \varepsilon$ , supposing, say,  $\mathbf{x}^* \approx \frac{1}{2}(\mathbf{a}_n + \mathbf{a}_{n-1})$  with error  $\varepsilon$ .

## Recall:

- One-dimensional derivatives of first and second order
- Gradient
- Hessian
- Taylor expansion

**What are the first- and second-order derivatives of**  
 $x, x^3, \sin x, \ln x$  (or  $\log x$ ),  $|x|$ ?

The **gradient** is a multi-variable generalisation of the derivative

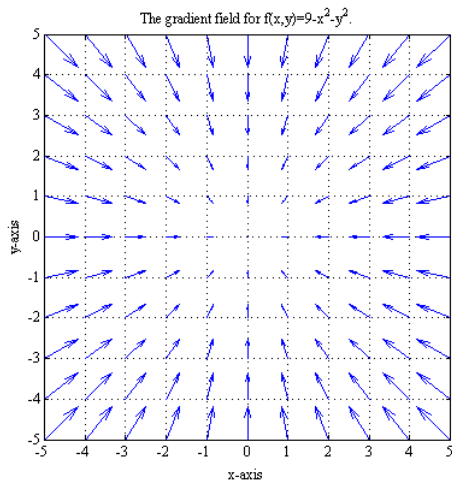
The gradient of a differentiable function  $f(\mathbf{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$  at a point  $\mathbf{a}$  is the (row- or **column**-) vector whose components are the partial derivatives of  $f$  at  $\mathbf{a}$ :

$$\nabla_{\mathbf{a}} f = \left( \frac{\partial f}{\partial x_i} \Big|_{\mathbf{a}} \right)_{i=1}^n.$$

**Example:**  $f(\mathbf{x}) = 2x_1 + 3x_2^2$ ,  $\nabla f = (2 \quad 6x_2)^\top$ ,  $\nabla_{\mathbf{a}} f = (2 \quad 6)^\top$ , if  $\mathbf{a} = (0, 1)$ .

If at  $\mathbf{a}$ , the gradient of a function is not the zero vector, it has the direction of **fastest increase** of the function at  $\mathbf{a}$ .

# Gradient



# Hessian and Taylor expansion

The **Hessian matrix** or **Hessian** is a square matrix of second-order partial derivatives that describes the local curvature and is the generalisation of the second derivative on multi-variable functions.

If all second partial derivatives of  $f = f(\mathbf{x})$  exist and are continuous, then the Hessian matrix  $\mathbf{H}_{\mathbf{a}}f$  of  $f$  at  $\mathbf{a}$  is a square  $n \times n$  matrix whose elements are defined as follows:

$$\mathbf{H}_{i,j} = \left. \frac{\partial^2 f}{\partial x_i \partial x_j} \right|_{\mathbf{a}}, \quad i, j = 1, \dots, n.$$

The **Taylor** expansion of  $f : \mathbb{R} \rightarrow \mathbb{R}$  that is infinitely differentiable at  $a$  is

$$T_f(x) = f(a) + \frac{f'(a)}{1!}(x - a) + \frac{f''(a)}{2!}(x - a)^2 + \frac{f'''(a)}{3!}(x - a)^3 + \dots$$

The generalisation on  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  in a neighbourhood of  $\mathbf{a}$  is

$$T_f(\mathbf{x}) = f(\mathbf{a}) + (\mathbf{x} - \mathbf{a})^\top \nabla_{\mathbf{a}} f + \frac{1}{2!}(\mathbf{x} - \mathbf{a})^\top \mathbf{H}_{\mathbf{a}}f (\mathbf{x} - \mathbf{a}) + \dots$$

Recall that  $\mathbf{x}$ ,  $\mathbf{a}$  and  $\nabla_{\mathbf{a}} f$  are defined to be column-vectors.

# Gradient descent (Steepest descent)

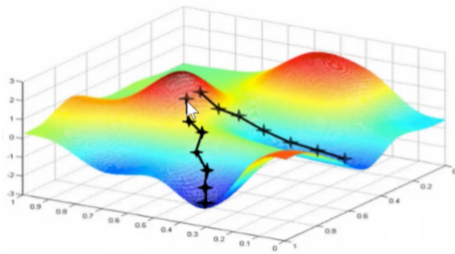
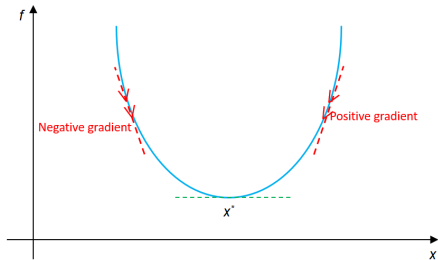
Gradient descent is based on the observation that if  $f(\mathbf{x})$  is defined and differentiable in a neighbourhood of a point  $\mathbf{a}$ , then  $f(\mathbf{x})$  decreases fastest in a neighbourhood of  $\mathbf{a}$  in the direction of  $-\nabla f(\mathbf{a})$ . One obtains the following formula:

$$\mathbf{a}_{n+1} = \mathbf{a}_n - \gamma \nabla f(\mathbf{a}_n)$$

for  $\gamma \in \mathbb{R}_+$  small enough, then  $f(\mathbf{a}_n) \geq f(\mathbf{a}_{n+1})$ . With this observation in mind, one starts with a guess  $\mathbf{a}_0$  for a local minimum of  $f$ , and considers the sequence  $\{\mathbf{a}_n\}$  such that

$$\mathbf{a}_{n+1} = \mathbf{a}_n - \gamma_n \nabla F(\mathbf{a}_n), \quad n \geq 0.$$

Here the value of the step size  $\gamma_n$  may be non-fixed and changed at every iteration (many possible ways to choose).





# (Nonlinear) Conjugate Gradient method

Given a function  $f(\mathbf{x})$ ,  $\mathbf{x} \in \mathbb{R}^n$  and an initial approximation  $\mathbf{a}_0$ , one starts in the steepest descent direction:

$$\Delta \mathbf{a}_0 = -\nabla_{\mathbf{a}_0} f.$$

Find the step length  $\alpha_0 := \arg \min_{\alpha} f(\mathbf{a}_0 + \alpha \Delta \mathbf{a}_0)$  and the next point  $\mathbf{a}_1 = \mathbf{a}_0 + \alpha_0 \Delta \mathbf{a}_0$ . After this iteration, the following steps constitute one iteration of moving along a subsequent conjugate direction  $s_n$ , where  $s_0 = \Delta \mathbf{a}_0$ :

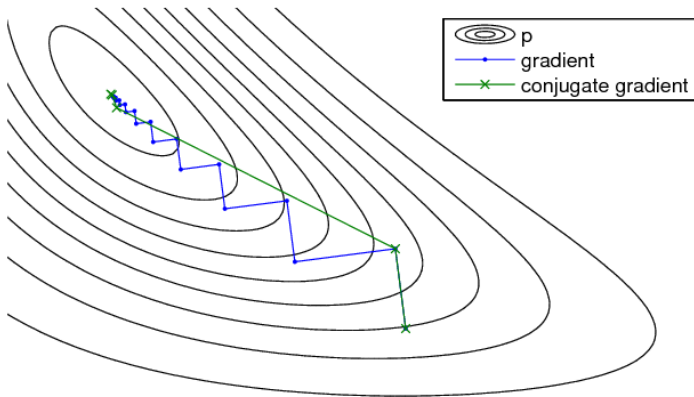
- Calculate the steepest direction  $\Delta \mathbf{a}_n = -\nabla_{\mathbf{a}_n} f$ .
- Compute  $\beta_n$  according to certain formulas.
- Update the conjugate direction  $s_n = \Delta \mathbf{a}_n + \beta_n s_{n-1}$ .
- Find  $\alpha_n = \arg \min_{\alpha} f(\mathbf{a}_n + \alpha s_n)$ .
- Update the position:  $\mathbf{a}_{n+1} = \mathbf{a}_n + \alpha_n s_n$ .

The choice of  $\beta_n$  due to Fletcher-Reeves:

$$\beta_n^{FR} = \frac{\Delta \mathbf{a}_n^T \Delta \mathbf{a}_n}{\Delta \mathbf{a}_{n-1}^T \Delta \mathbf{a}_{n-1}}.$$

The choice of  $\beta_n$  due to Polak-Ribiere:

$$\beta_n^{PR} = \frac{\Delta \mathbf{a}_n^T (\Delta \mathbf{a}_n - \Delta \mathbf{a}_{n-1})}{\Delta \mathbf{a}_{n-1}^T \Delta \mathbf{a}_{n-1}}.$$



# Newton's method. One-dimensional case

Let  $f : \mathbb{R} \rightarrow \mathbb{R}$  be convex and twice-differentiable. Find the roots of  $f'$  by constructing a sequence  $a_n$  from an initial guess  $a_0$  s.t.  $a_n \rightarrow x^*$  as  $n \rightarrow \infty$ , where  $f'(x^*) = 0$ , i.e.  $x^*$  is a stationary point of  $f$ .

From the Taylor expansion of  $f$  near  $a_n$  (think that  $x^* \approx a_n + \Delta a$ ),

$$f(a_n + \Delta a) \approx T_f(\Delta a) := f(a_n) + f'(a_n)\Delta a + \frac{1}{2}f''(a_n)(\Delta a)^2.$$

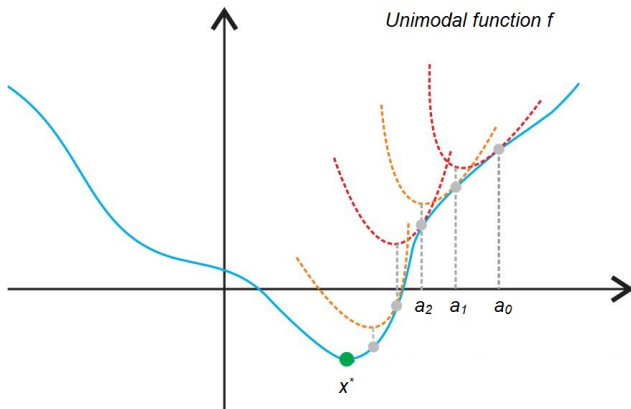
Use this quadratic functions as approximants to  $f$  in a neighbourhood of  $a_n$  and find their minimum points (take into account that  $f''(x) \geq 0$  (why?)):

$$0 = \frac{dT_f(\Delta a)}{d\Delta a} = f'(a_n) + f''(a_n)\Delta a \quad \Rightarrow \quad \Delta a = -\frac{f'(a_n)}{f''(a_n)}.$$

Incrementing  $a_n$  by this  $\Delta a$  yields a point closer to  $x^*$ :

$$a_{n+1} = a_n + \Delta a = a_n - \frac{f'(a_n)}{f''(a_n)}.$$

It is proved that for the chosen class of  $f$ ,  $a_n \rightarrow x^*$  as  $n \rightarrow \infty$ .



**Question:** what is the problem with this pic from the Internet?

# Newton's method. Multidimensional case

Let  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  be convex and  $H_{\mathbf{x}}f$  is invertible for  $\mathbf{x} \in \mathbb{R}^n$ . The one-dimensional scheme can be generalized to several dimensions by replacing the derivative with the gradient,  $\nabla f$ , and the reciprocal of the second derivative with the inverse of the Hessian matrix,  $\mathbf{H}f$ :

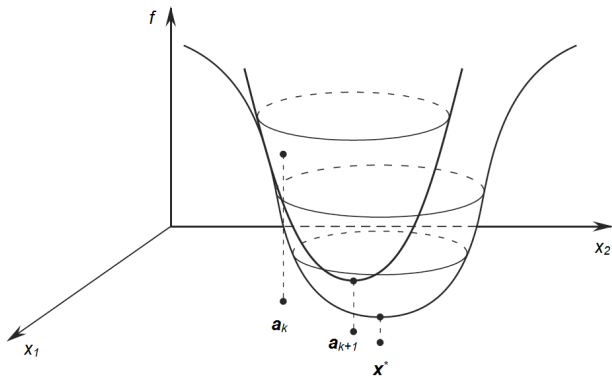
$$\mathbf{a}_{n+1} = \mathbf{a}_n - [\mathbf{H}_{\mathbf{a}_n}f]^{-1} \nabla_{\mathbf{a}_n} f, \quad n \geq 0.$$

Often Newton's method is modified by including the variable step size  $\gamma_n \in (0, 1)$ :

$$\mathbf{a}_{n+1} = \mathbf{a}_n - \gamma_n [\mathbf{H}_{\mathbf{a}_n}f]^{-1} \nabla_{\mathbf{a}_n} f, \quad n \geq 0,$$

to satisfy the convergence conditions.

**Remark.** It is difficult to compute the Hessian. Other methods called **Quasi-Newton** propose to find an approximation to the Hessian to simplify the scheme  $\rightarrow$  We consider some of them within the course projects.



Newton's and Gradient Decent methods demonstration

# Levenberg-Marquardt algorithm (LMA)

The application of LMA is the **least-squares curve fitting problem**: given a set  $(x_i, y_i)_{i=1}^m$ , find the parameters  $\beta$  (column vector) of the model curve  $f(x, \beta)$  so that the sum of the squares of the deviations  $S(\beta)$  is minimized:

$$\arg \min_{\beta} S(\beta) \equiv \arg \min_{\beta} \sum_{i=1}^m [y_i - f(x_i, \beta)]^2.$$

Start with an initial guess for  $\beta$ . In each iteration step, the parameter vector *beta* is replaced by a new estimate  $\beta + \Delta\beta$ . To determine  $\Delta\beta$ , the function  $f(x_i, \beta + \Delta\beta)$  is approximated by its linearization:

$$f(x_i, \beta + \Delta\beta) \approx f(x_i, \beta) + J_i \Delta\beta, \quad J_i = (\nabla f(x_i, \beta))^T.$$

The sum  $S(\beta)$  has its minimum at a zero gradient with respect to  $\beta$ . The above first-order approximation of  $f(x_i, \beta + \Delta\beta)$  gives

$$S(\beta + \Delta\beta) \approx \sum_{i=1}^m [y_i - f(x_i, \beta) - J_i \Delta\beta]^2,$$

or in vector notation,

$$S(\beta + \Delta\beta) \approx [\mathbf{y} - \mathbf{f}(\beta)]^T [\mathbf{y} - \mathbf{f}(\beta)] - 2 [\mathbf{y} - \mathbf{f}(\beta)]^T \mathbf{J} \Delta\beta + \Delta\beta^T \mathbf{J}^T \mathbf{J} \Delta\beta,$$

where  $\mathbf{J}$  is the Jacobian matrix, whose  $i$ -th row equals  $\mathbf{J}_i$ , and where  $\mathbf{f}(\beta)$  and  $\mathbf{y}$  are vectors with  $i$ -th component  $f(x_i, \beta)$  and  $y_i$ , respectively.

Taking the derivative of  $S(\beta + \Delta\beta)$  with respect to  $\Delta\beta$  and setting to zero gives

$$(\mathbf{J}^T \mathbf{J}) \Delta\beta = \mathbf{J}^T [\mathbf{y} - \mathbf{f}(\beta)],$$

that is in fact a system of linear equations with respect to  $\Delta\beta$ .

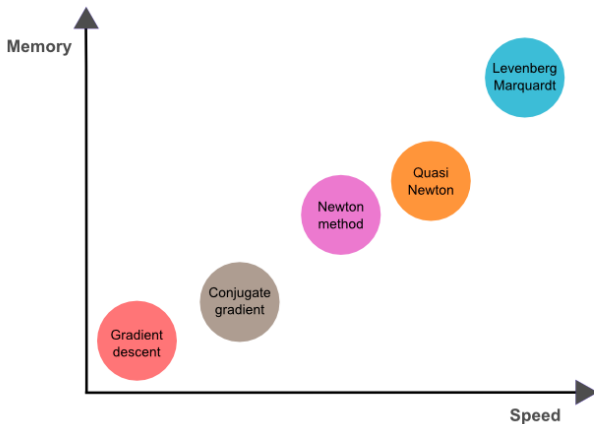
The system may be replaced by

$$(\mathbf{J}^T \mathbf{J} + \lambda \mathbf{I}) \Delta\beta = \mathbf{J}^T [\mathbf{y} - \mathbf{f}(\beta)],$$

where  $\mathbf{I}$  is the identity matrix, giving the increment  $\Delta\beta$  to the estimated parameter vector  $\beta$ .

## Demonstration





Machine Learning blog: 5 algorithms to train a neural network

Thank you for your attention!