Lecture 3
Algorithms for unconstrained nonlinear optimisation. First- and second-order methods

Analysis and Development of Algorithms

УНИВЕРСИТЕТ ИТМО

Overview

Terms

Problem

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

Remark. We use numerical methods with allowed error $\varepsilon > 0$. In the iterative algorithms below, we stop if $\|\mathbf{a}_n - \mathbf{a}_{n-1}\| < \varepsilon$, supposing $\mathbf{x}^* \approx \mathbf{a}_n$ with error ε.

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 $\log x$), $|x|$?

Gradient

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

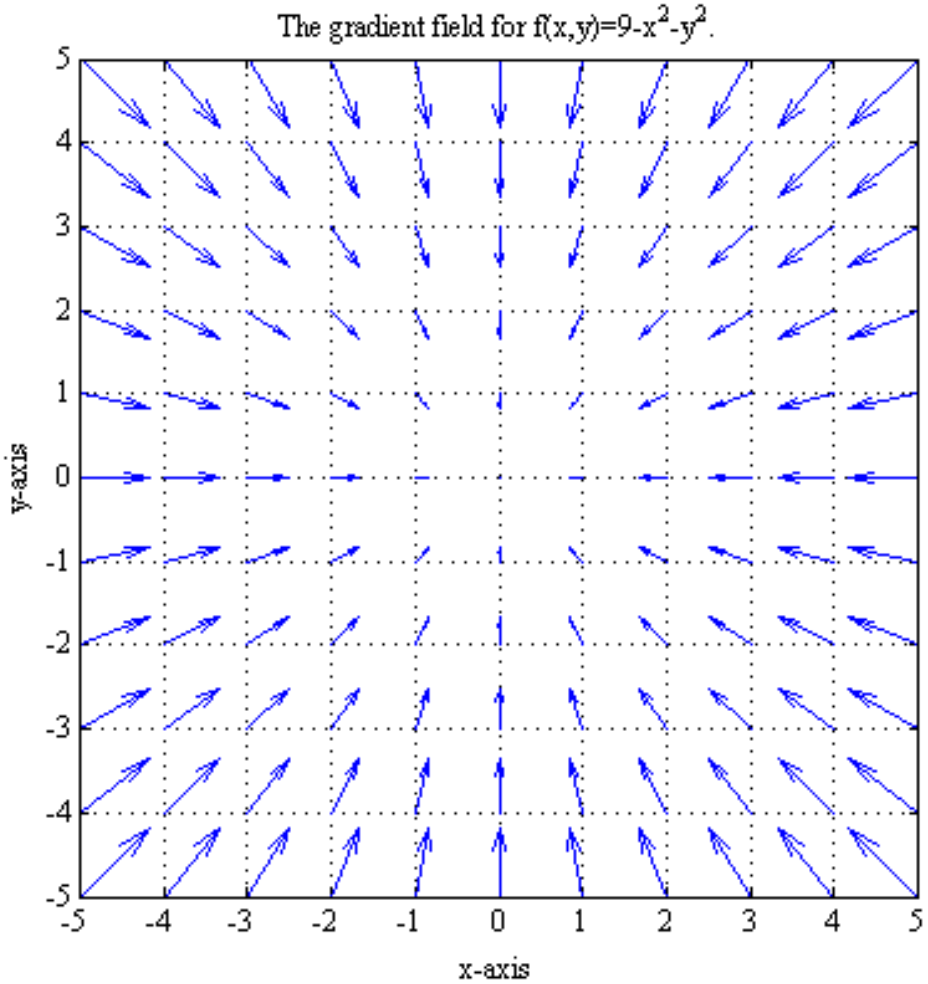
The gradient of a differentiable function $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ at a point $\mathbf{a}$ is the (row- or **column**-) vector whose components are the partial derivatives of f at $\mathbf{a}$:

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

Example: $f(\mathbf{x}) = 2x_1 + 3x_2^2$, $\nabla f = (2 \quad 6x_2)^{\mathsf{T}}$, $\nabla_{\mathbf{a}} f = (2 \quad 6)^{\mathsf{T}}$, 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


The gradient field for f(x,y)=9-x^2-y^2.
y-axis
x-axis
5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
-4
-3
-2
-1
0
1
2
3
4
5

Hessian and Taylor expansion

The **Hessian matrix** or **Hessian** is a square matrix of second-order partial derivatives of f that describes the local curvature of f (the generalisation of the second derivative).

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:

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

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

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

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

$$T_f(\mathbf{x}) = f(\mathbf{a}) + (\mathbf{x}-\mathbf{a})^\mathsf{T} \nabla_\mathbf{a} f + \frac{1}{2!}(\mathbf{x}-\mathbf{a})^\mathsf{T} \mathbf{H}_\mathbf{a} f \, (\mathbf{x}-\mathbf{a}) + \ldots.$$

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

Gradient descent (Steepest descent)

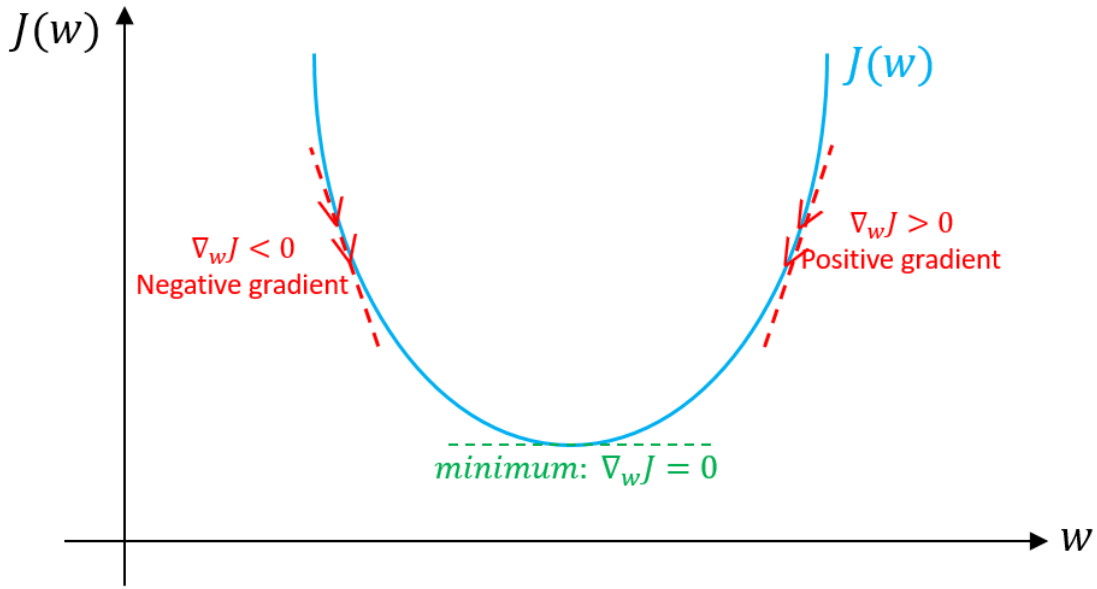
Gradient descent is based on the observation that if $f(\mathbf{x})$ is defined and differentiable in a neighborhood of a point $\mathbf{a}$, then $f(\mathbf{x})$ decreases fastest, i.e. one gets closer to $\mathbf{x}^*$, if one goes from $\mathbf{a}$ in the direction of $-\nabla f(\mathbf{a})$. It follows that, if

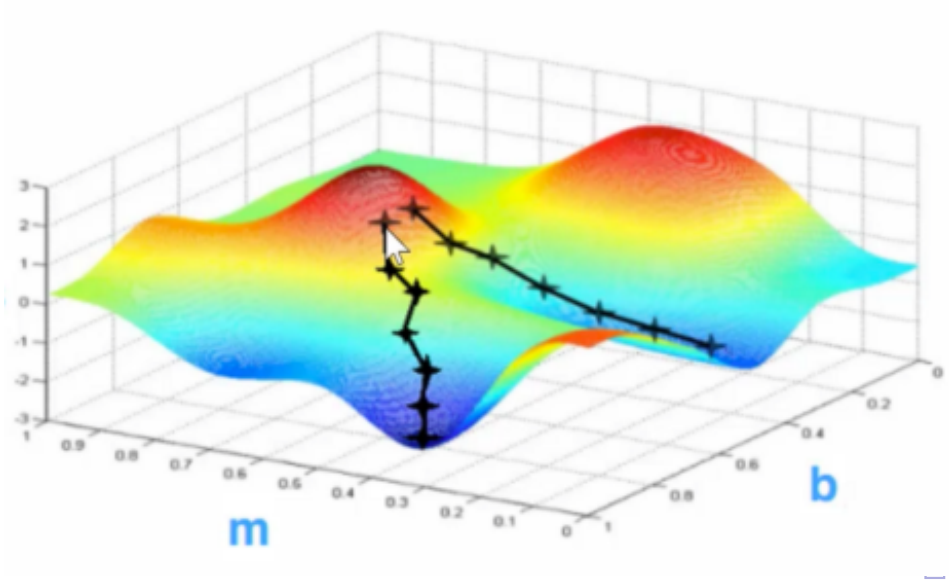
$$\mathbf{a}_{n+1} = \mathbf{a}_n - \gamma \nabla f(\mathbf{a}_n)$$

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

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

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


$J(w)$
$J(w)$
$\nabla_w J < 0$
Negative gradient
$\nabla_w J > 0$
Positive gradient
$minimum$: $\nabla_w J = 0$
w


m
b

(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 $s_0 = \Delta \mathbf{a}_0$:

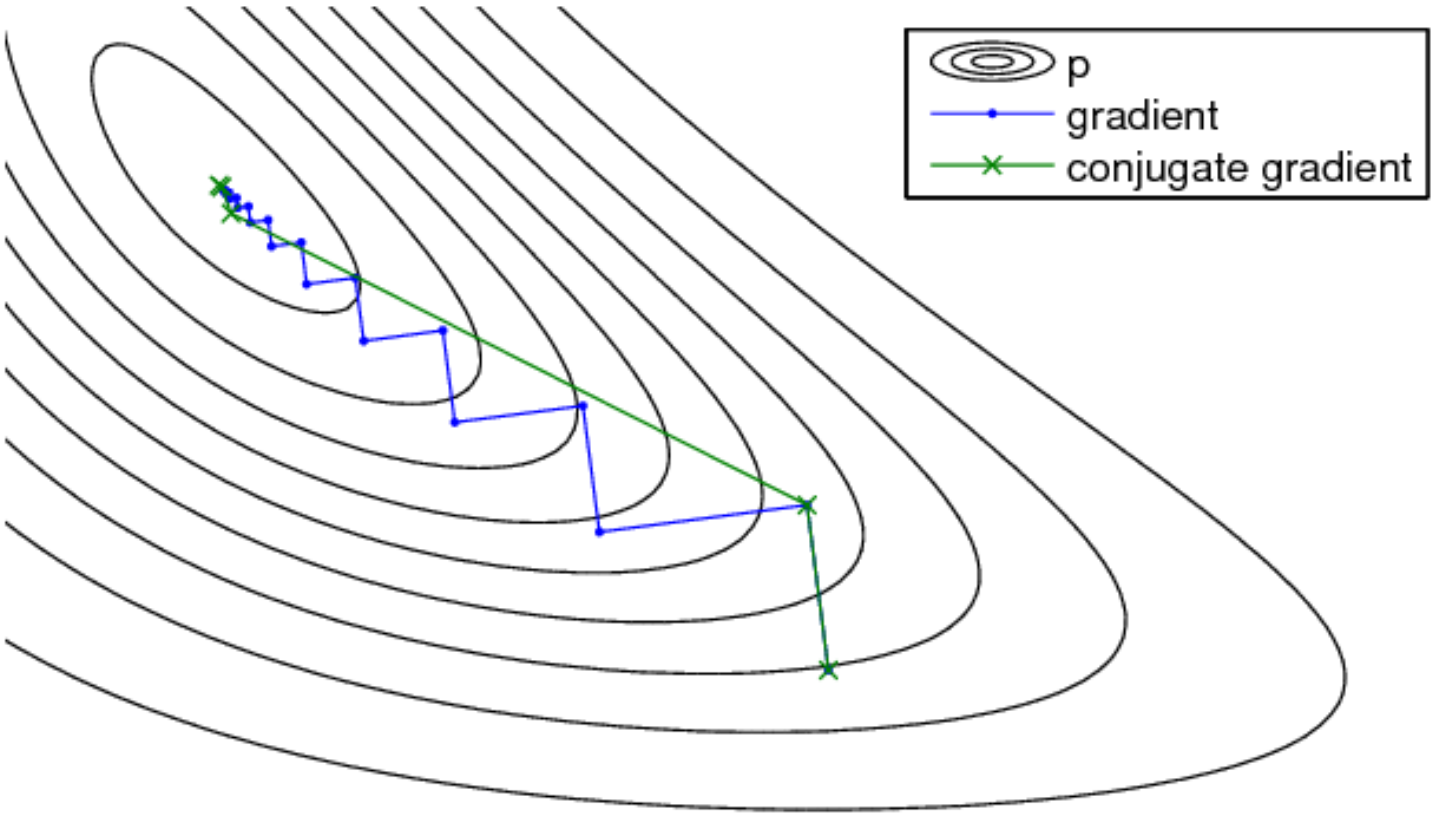
- Calculate the steepest direction $\Delta \mathbf{a}_n = -\nabla_{\mathbf{a}_n} f$.
- Compute β_n according to certain formulas.
- Update the conjugate direction $s_n = \Delta \mathbf{a}_n + \beta_n s_{n-1}$.
- Find $\alpha_n = \arg\min_\alpha f(\mathbf{a}_n + \alpha s_n)$.
- Update the position: $\mathbf{a}_{n+1}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

The choice of β_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 β_n due to Polak-Ribiere:

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


p
gradient
conjugate gradient

Newton's method. One-dimensional case

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

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

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

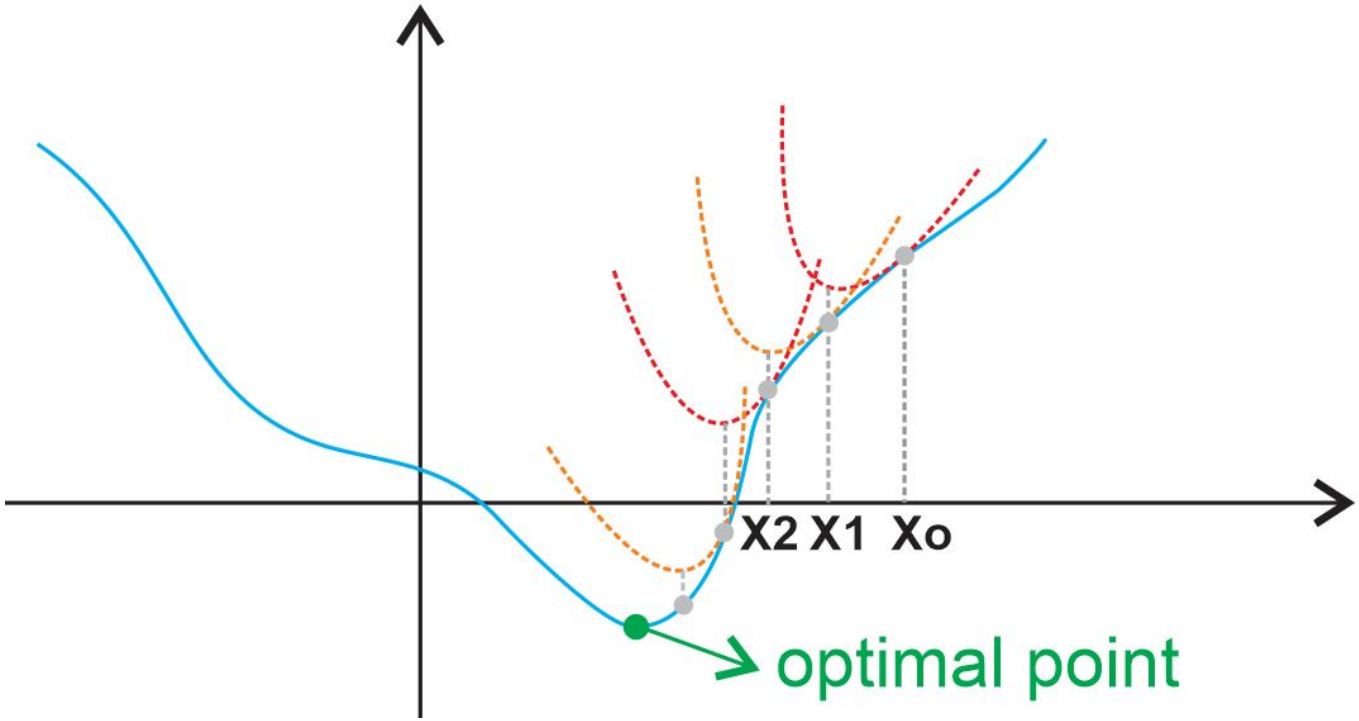
Use this quadratic functions as approximants to f in a neighbourhood of a_n and find their minimum points (take into account that $f''(x) \geqslant 0$):

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

Incrementing a_n by this Δa 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, $a_n \to x^*$ as $n \to \infty$.



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

Newton's method. Multidimensional case

Let $f : \mathbb{R}^n \to \mathbb{R}$ 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, ∇f, and the reciprocal of the second derivative with the inverse of the Hessian matrix, $\mathbf{H} f$:

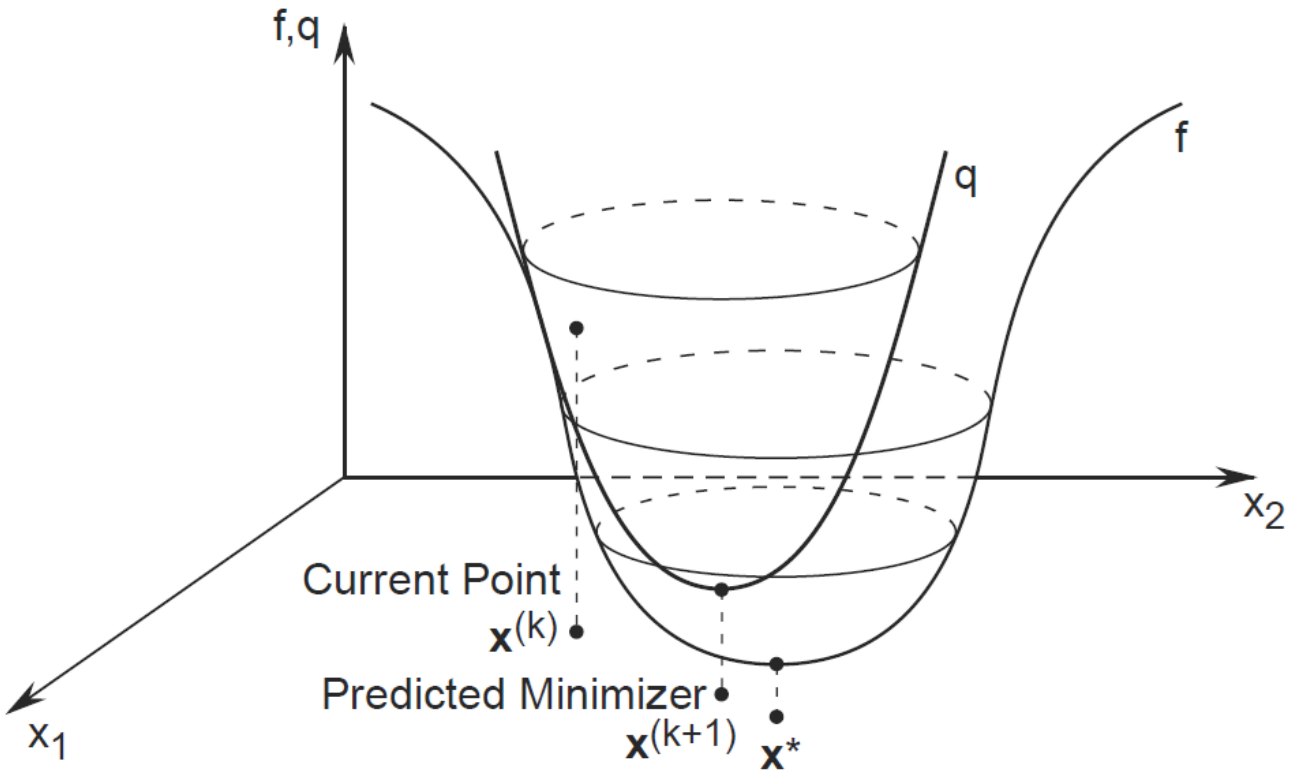
$$\mathbf{a}_{n+1} = \mathbf{a}_n - [\mathbf{H}_{\mathbf{a}_n} f]^{-1} \nabla_{\mathbf{a}_n} f, \ \ n \geq 0.$$

Often Newton's method is modified to include a small step size $\gamma \in (0, 1)$:

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

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 $\to$ 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 β (column vector) of the model curve $f(x, \beta)$ so that the sum of the squares of the deviations $S(\beta)$ is minimized:

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

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

$$f(x_i, \beta + \Delta\beta) \approx f(x_i, \beta) + J_i \Delta\beta, \qquad \mathbf{J}_i = \left(\nabla f(x_i, \beta)\right)^T .$$

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

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

or in vector notation,

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

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

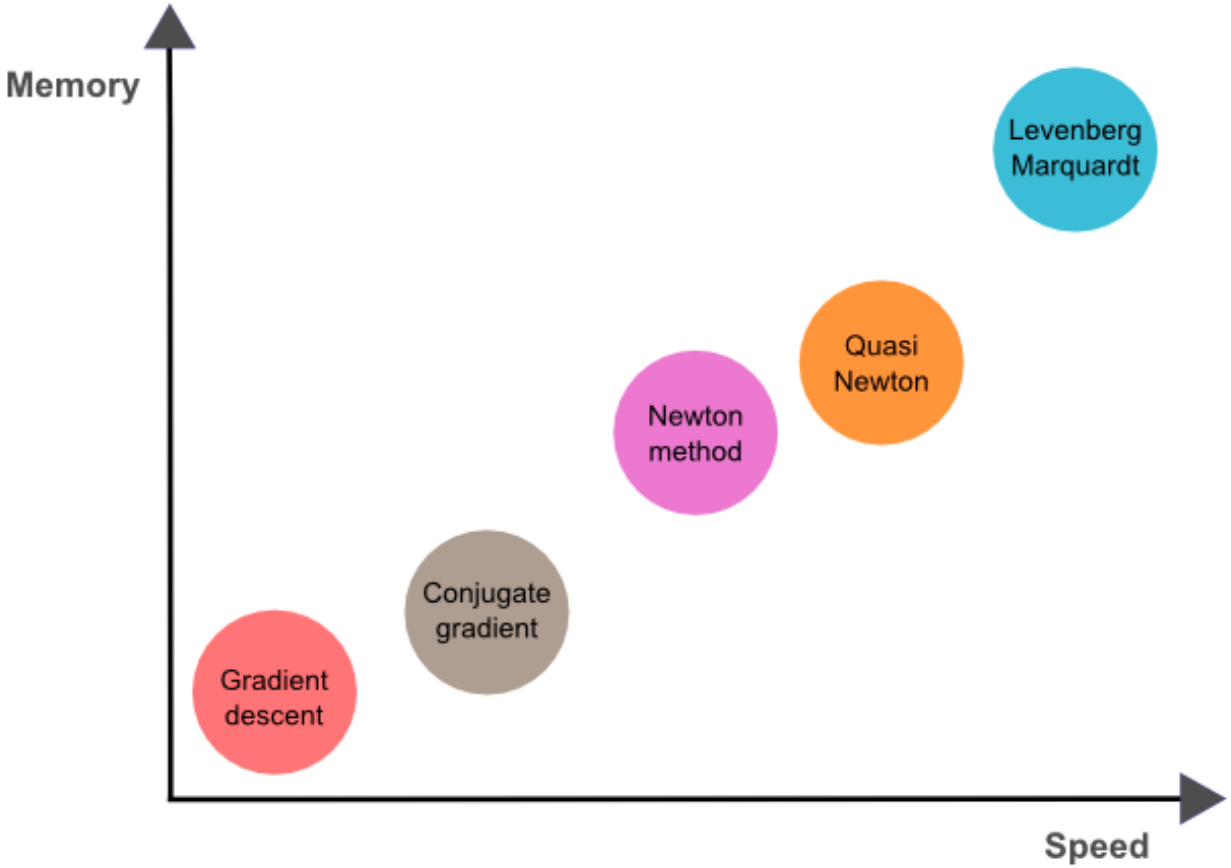
$$\left(\mathbf{J}^{\mathrm{T}} \mathbf{J}\right) \Delta\beta = \mathbf{J}^{\mathrm{T}} [\mathbf{y} - \mathbf{f}(\beta)].$$

This is a set of linear equations, which can be solved for $\Delta\beta$.
This equation may be replaced by

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

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

Demonstration



Machine Learning blog: 5 algorithms to train a neural network

Thank you for your attention!