Lecture 3
Algorithms for unconstrained nonlinear optimization. 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 $\mathbf{x} = (x_1 \quad x_2 \quad \cdots \quad x_n)$ is a **row**-vector
To solve the optimization problem $f(\mathbf{x}) \to \min_{\mathbf{x}\in Q}$ 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 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 generalization of the derivative

The gradient of a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ at $\mathbf{a}$ is the n-dimensional **column**-vector $\nabla f(\mathbf{a})$ whose elements are

$$\left.\frac{\partial f}{\partial x_i}\right|_{\mathbf{a}}, \quad i = 1, \ldots, n.$$

Example: $f(\mathbf{x}) = 2x_1 + 3x_2^2$

$$\nabla f(\mathbf{x}) = (2 \quad 6x_2)^{\mathsf{T}}$$

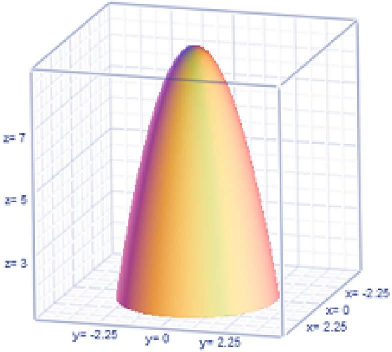
$$\text{If } \mathbf{a} = (0 \quad 1),\ \text{then } \nabla f(\mathbf{a}) = (2 \quad 6)^{\mathsf{T}}$$

If the gradient of our function is not the zero vector at **a**, it has the direction of **fastest increase** of the function at **a**.

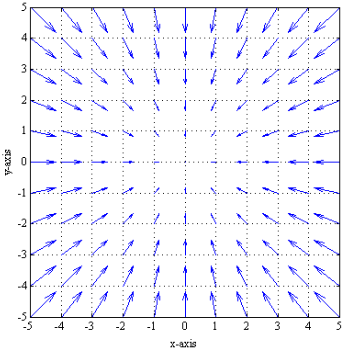
Gradient

$$z = f(x, y) = 9 - (x^2 + y^2)$$

3D plot



Gradient field



Hessian

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

The Hessian of a twice differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ at $\mathbf{a}$ is the $n \times n$ matrix $\mathbf{H}f(\mathbf{a})$ whose elements are

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

Example: $f(\mathbf{x}) = x_1^2 + 3x_1x_2^3$

$$H^{\mathbf{x}}_{1,1} = \frac{\partial^2 f}{\partial x_1^2} = 2,\ H^{\mathbf{x}}_{1,2} = \frac{\partial^2 f}{\partial x_1 \partial x_2} = H^{\mathbf{x}}_{2,1} = \frac{\partial^2 f}{\partial x_2 \partial x_1} = 9x_2^2,\ H^{\mathbf{x}}_{2,2} = \frac{\partial^2 f}{\partial x_2^2} = 18x_1x_2$$

$$\mathbf{H}f(\mathbf{x}) = \begin{pmatrix} 2 & 9x_2^2 \\ 9x_2^2 & 18x_1x_2 \end{pmatrix}$$

$$\text{If } \mathbf{a} = (1 \quad 2),\ \text{then } \mathbf{H}f(\mathbf{a}) = \begin{pmatrix} 2 & 36 \\ 36 & 36 \end{pmatrix}$$

Taylor expansion

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

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

Its generalization for $f : \mathbb{R}^n \to \mathbb{R}$ at $\mathbf{a}$ is

$$T_f(\mathbf{x}) = f(\mathbf{a}) + (\mathbf{x}-\mathbf{a})\frac{\nabla f(\mathbf{a})}{1!} + (\mathbf{x}-\mathbf{a})\frac{\mathbf{H}f(\mathbf{a})}{2!}(\mathbf{x}-\mathbf{a})^{\mathsf{T}} + \dots .$$

Recall that $\mathbf{x}$ and $\mathbf{a}$ are defined to be **row**-vectors and $\nabla f(\mathbf{a})$ to be a **column**-vector here.

First-order methods

Gradient descent (Steepest descent)

Gradient descent is based on the observation that if f is differentiable at $\mathbf{a}$, then $f(\mathbf{x})$ decreases **fastest** in a neighbourhood of $\mathbf{a}$ in the direction of $-\nabla f(\mathbf{a})$. One may write down the following formula:

$$\mathbf{a}_{n+1} = \mathbf{a}_n - \beta_n \nabla f(\mathbf{a}_n), \qquad \beta_n > 0, \qquad n = 0, 1, \ldots,$$

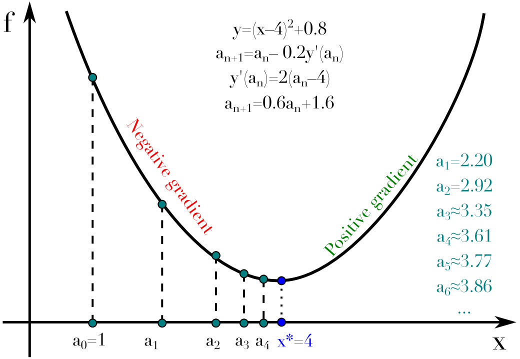
starting with some initial approximation $\mathbf{a}_0$.

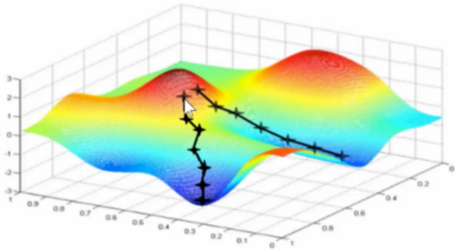
If the factor β_n is chosen properly, then $f(\mathbf{a}_n) \geq f(\mathbf{a}_{n+1}) \geqslant f(\mathbf{a}_{n+2}) \geqslant \ldots$ and furthermore $\mathbf{a}_n \to \mathbf{x}^*$ as $n \to \infty$, where $\mathbf{x}^*$ is a local minimum.

If f is convex, ∇f is Lipschitz and the choice of β_n is due to Barzilai-Borwein,

$$\beta_n^{BB} = \frac{\left|(\mathbf{a}_n - \mathbf{a}_{n-1})\left(\nabla f(\mathbf{a}_n) - \nabla f(\mathbf{a}_{n-1})\right)\right|}{\left\|\nabla f(\mathbf{a}_n) - \nabla f(\mathbf{a}_{n-1})\right\|^2},$$

then the uniform convergence is guaranteed.


f
y=(x–4)²+0.8
aₙ₊₁=aₙ– 0.2y'(aₙ)
y'(aₙ)=2(aₙ–4)
aₙ₊₁=0.6aₙ+1.6
Negative gradient
Positive gradient
a1=2.20
a2=2.92
a3≈3.35
a4≈3.61
a5≈3.77
a6≈3.86
...
a0=1
a1
a2
a3
a4
x*=4
x



(Nonlinear) Conjugate Gradient method

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

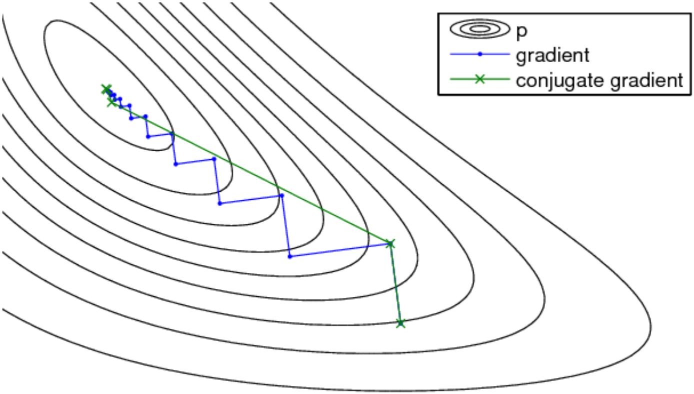
$$\Delta \mathbf{a}_0 = -\nabla f(\mathbf{a}_0).$$

Find the step size $\alpha_0 := \arg\min_\alpha f(\mathbf{a}_0 + \alpha \Delta \mathbf{a}_0)$ and $\mathbf{a}_1 = \mathbf{a}_0 + \alpha_0 \Delta \mathbf{a}_0$. After this iteration, the following steps with $n = 1, 2, \ldots$ constitute one iteration of moving along a subsequent conjugate direction s_n, where $s_0 = \Delta \mathbf{a}_0$:

- Calculate the steepest direction $\Delta \mathbf{a}_n = -\nabla f(\mathbf{a}_n)$.
- 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 (to guarantee the uniform convergence $\mathbf{a}_n \to \mathbf{x}^*$ as $n \to \infty$ for convex f and Lipschitz ∇f) is due to, say, Fletcher-Reeves or Polak-Ribiere:

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



Gradient Descent oscillates much and it slows down the convergence to $\mathbf{x}^*$, while the Non-linear Conjugate Gradient moves to $\mathbf{x}^*$ in a rather straightforward manner.

Second-order methods

Newton's method. One-dimensional case

Let $f : \mathbb{R} \to \mathbb{R}$ be convex and twice differentiable. Find a root of f' by constructing a sequence a_n from an initial approximation a_0 so that $a_n \to x^*$ as $n \to \infty$, where $f'(x^*) = 0$.

From the Taylor expansion of f near a_n,

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

We use this quadratic function (with respect to Δa) as an approximant to f in a neighbourhood of a_n. The vertex of the corresponding parabola gives us the next point a_{n+1}. To find the vertex x-coordinate, we write:

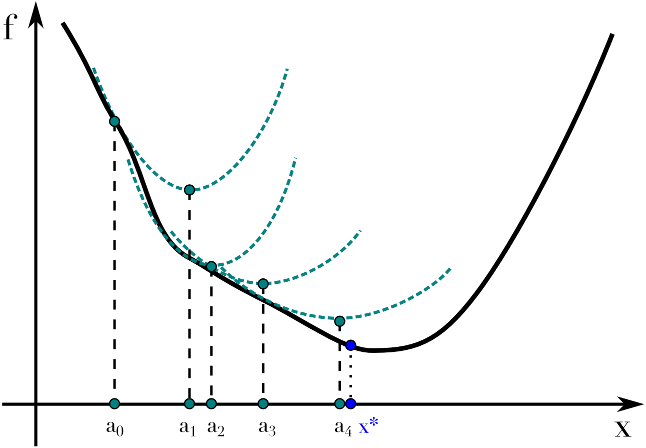
$$0 = \frac{dT_f(\Delta a)}{d\Delta a} = f'(a_n) + f''(a_n)\Delta a \quad \Rightarrow \quad \Delta a = -\frac{f'(a_n)}{f''(a_n)}.$$

Here $f''(x) \geqslant 0$ (*why?*). Incrementing a_n by this Δa gives us 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 one has $a_n \to x^*$ as $n \to \infty$.

Unimodal function f



Question: what happens if f is a quadratic function?

Newton's method. Multidimensional case

Let $f : \mathbb{R}^n \to \mathbb{R}$ be convex and $Hf(\mathbf{x})$ is invertible for $\mathbf{x} \in \mathbb{R}^n$.

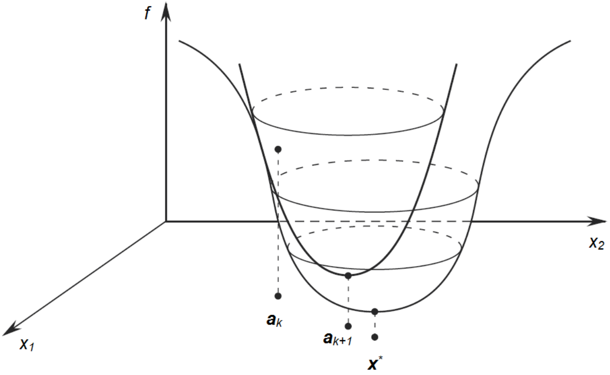
The one-dimensional scheme can be generalized to several dimensions by replacing the derivative with the gradient, ∇f, and the reciprocal of the second derivative with the inverse of the Hessian matrix, $\mathbf{H}f$:

$$\mathbf{a}_{n+1} = \mathbf{a}_n - [\mathbf{H}f(\mathbf{a}_n)]^{-1} \nabla f(\mathbf{a}_n), \qquad n = 0, 1, \ldots.$$

By decreasing the step size, one obtains the *relaxed* Newton's method:

$$\mathbf{a}_{n+1} = \mathbf{a}_n - \beta [\mathbf{H}f(\mathbf{a}_n)]^{-1} \nabla f(\mathbf{a}_n), \qquad \beta \in (0, 1), \qquad n = 0, 1, \ldots,$$

to guarantee the method's convergence. One often supposes then that f is strictly convex and $\mathbf{H}f$ is Lipschitz.



An example of comparison of Newton's and Gradient Decent methods

Active learning

What about the so-called **quasi-Newton** methods?
What are they for?
Which **quasi-Newton** methods are commonly used in ML?

Levenberg-Marquardt algorithm (LMA)

The LMA is a *pseudo*-second order method, its application is to solve non-linear least squares problems.

For a set $(x_i, y_i)_{i=1}^m$, find a **column**-vector β^* of the parameters $\beta = (\beta_1 \ \cdots \ \beta_n)^\mathsf{T}$ of the function $f(x, \beta)$ such that

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

Start with an initial guess for β. At each iteration, β is replaced $\beta + \Delta\beta$. To determine $\Delta\beta$, $f\left(x_i, \beta + \Delta\beta\right)$ is approximated by its linearization:

$$f(x_i, \beta + \Delta\beta) \approx f(x_i, \beta) + J_i \Delta\beta, \quad J_i = \left(\frac{\partial f(x_i, \beta_j)}{\partial \beta_j}\right)_{j=1}^n, \quad J_i \text{ is a } \textbf{row}\text{-vector.}$$

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

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

or in a vector notation,

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

where $\mathbf{J}$ is the Jacobian matrix, whose i-th row equals J_i, and where $\mathbf{f}(\beta)$, $\mathbf{y}$ and β are **column**-vectors with i-th component $f(x_i, \beta)$, y_i and β_i, respectively.
Taking the derivative of $S(\beta + \Delta\beta)$ with respect to $\Delta\beta$ and setting to zero gives

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

that is in fact a system of linear equations with respect to $\Delta\beta$.
The system may be replaced by the following *damped version*:

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

where $\mathbf{I}$ is the identity matrix, giving the increment $\Delta\beta$ to the estimated parameter vector β. The *damping factor* $\lambda > 0$ is adjusted at each iteration and should be chosen to guarantee the method's convergence.

Active learning. Why the LMA is thought to interpolate between the Gauss-Newton algorithm (a modification of Newton's method to solve non-linear least squares problems) and the Gradient Descent method (by means of λ)?

Demonstration

Thank you for your attention!