Unconstrained Minimization

In this lecture, we will develop a generic framework for solving n-dimensional unconstrained minimization problems of the form

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}).$$

By "solving", we imply "finding a local minimizer", remembering that for convex functions it will coincide with a global minimizer.

1 General framework

A very broad family of the so-called line search-based iterative optimization methods works as follows: Starting with some *initial point*, we finding a direction a step in which decreases the value of the objective (*descent direction*), select a *step size* in that direction, update the point, and iterate the process *until convergence*. This general framework can be summarized as the following iterative procedure:

input : function f; initial point $\mathbf{x}_0$
output: (approximate) local minimizer $\mathbf{x}^*$ of f
Start with an initial guess $\mathbf{x}_0$
for $k = 1, 2, \ldots,$ *until convergence* **do**
  Find descent direction $\mathbf{d}_k$
  Find step size α_k
  Update $\mathbf{x}_k \leftarrow \mathbf{x}_{k-1} + \alpha_k \mathbf{d}_k$
end
Return $\mathbf{x}^* = \mathbf{x}^k$

Algorithm 1: Generic unconstrained minimization algorithm

Initialization. Selecting the initial point $\mathbf{x}_0$ is a very important ingredient in the solution of optimization problems, which is often overlooked. For non-convex problems, selecting a good initial guess that is sufficiently close to a good (hopefully, global) minimum (we often say that the initial point is in the *basin of attraction* of a global minimum) is crucial to obtain a useful result. For convex problems, while most algorithms will eventually converge to a global minimum, a good initialization has crucial impact on the convergence speed. Unfortunately, very little general facts can be said about initialization. It is very problem-specific, and is more art than science. In some problems, however, such as those using multi-resolution schemes a good initialization of a fine-resolution problem can naturally come from the solution of a coarser-level problem.

Stopping criteria. Different criteria or their combination can be checked in order to determine if convergence has occurred. For example:

1. *First-order optimality condition*: since the solution has to satisfy $\nabla f(x^*) = 0$ one can stop the algorithm when the gradient norm becomes sufficiently small, e.g., either $\|\nabla f(x^k)\|_2 \leq \epsilon$ or $\|\nabla f(x^k)\|_\infty \leq \epsilon$ (typically, $\epsilon \sim 10^{-6} \div 10^{-8}$).

2. *Step size* stops the algorithm when the step size becomes sufficiently small, $\|\mathbf{x}_k - \mathbf{x}_{k-1}\| \leq \delta$

3. *Function value change* stops the algorithm when the absolute or the relative change of the function value becomes insignificant, $f(\mathbf{x}_{k-1}) - f(\mathbf{x}_k) \leq \delta$ or $\dfrac{f(\mathbf{x}_{k-1}) - f(\mathbf{x}_k)}{|f(\mathbf{x}_k)|} \leq \delta$.

Step size. The step size α_k at each iteration can be selected using one of these methods:

1. *Exact line search* solving

$$\alpha_k = \arg\min_{\alpha} f(\mathbf{x}_{k-1} + \alpha \mathbf{d}_k).$$

2. *Inexact line search* like the Armijo rule that finds α_k such that the decrease $f(\mathbf{x}_{k-1}) - f(\mathbf{x}_{k-1} + \alpha_k \mathbf{d}_k)$ is sufficient

3. *Constant step size*: under certain conditions (for example, if the gradient is Lipschitz continuous with the constant $1/\alpha$ and the step size is smaller than α), constant step size yields a converging algorithm. However, this method is generally not recommended. If the step size is too small, the convergence will typically be very slow; if it is too big, the algorithm might end up oscillating around the minimum or, even worse, diverge. Furthermore, constant step size does not take into account the fact that the objective might have different curvature at different points, which requires local adaptation of the step size.

4. *Decaying step size*: under the technical condition that $\alpha_k \downarrow 0$ and $\displaystyle\sum_{k=1}^{\infty} \alpha_k = \infty$ (e.g., $\alpha_k = 1/k$ or $\alpha_k = 1/\sqrt{k}$), a pre-defined sequence of decaying steps produces a convergent algorithm. Decaying step size is often the choice in stochastic optimization algorithms.

In what follows, we discuss in depth the most important ingredient distinguishing between different algorithms: the selection of the descent direction.

2 Steepest descent

As the name suggests, a descent direction has to be such a direction that a sufficiently small step in it decreases the value of the function. Formally, a descent direction $\mathbf{d}$ at points $\mathbf{x}$ has to satisfy

$$0 > f'_{\mathbf{d}}(\mathbf{x}) = \mathbf{d}^{\mathrm{T}} \nabla f(\mathbf{x}).$$

Geometrically, a descent direction forms an obtuse angle with the direction of the gradient or an acute angle with $-\nabla f(\mathbf{x})$.

A natural question to ask is what is best descent direction at a given point $\mathbf{x}$. Since a function can be locally approximated by a linear function, $f(\mathbf{x}+\mathbf{d}) \approx f(\mathbf{x}) + \mathbf{d}^{\mathrm{T}} \nabla f(\mathbf{x})$, we can say that the best descent direction is the one that produces the biggest decrease in the value of the function,

$$\mathbf{d} = \arg\min_{\mathbf{d}} f(\mathbf{x}+\mathbf{d}) - f(\mathbf{x}) \approx \arg\min_{\mathbf{d}} \mathbf{d}^{\mathrm{T}} \nabla f(\mathbf{x}).$$

Such a direction is usually referred to as *steepest descent direction*.

However, note that the above minimization problem is unbounded; in fact, we can select any vector $\mathbf{d}$ forming an acute angle with $-\nabla f(\mathbf{x})$, and make it arbitrarily large. In order to avoid this, we restrict $\mathbf{d}$ to be of unit length. This give rise to a plethora of possibilities, since every norm will yield a different result! The following choices are typical:

1. *Gradient descent* (often referred to as simply *steepest descent*) corresponding to the ℓ_2 norm:

$$\mathbf{d} = \arg\min_{\|\mathbf{d}\|_2=1} \mathbf{d}^{\mathrm{T}} \nabla f(\mathbf{x}) = -\nabla f(\mathbf{x}).$$

2. *Coordinate descent* corresponding to the ℓ_1 norm:

$$\mathbf{d} = \arg\min_{\|\mathbf{d}\|_1=1} \mathbf{d}^{\mathrm{T}} \nabla f(\mathbf{x}) = -(\nabla f(\mathbf{x}))_i,$$

   where i is the coordinate of the gradient vector having the largest absolute value,

$$i = \arg\max_{i=1,\ldots,n} |(\nabla f(\mathbf{x}))_i|.$$

   Some problems might have an arithmetically complex objective and gradient; however, once all but one variables are fixed, it might become significantly cheaper to compute. In such cases, coordinate (or, more generally, block-coordinate) descent methods are a popular choice.

Sometimes, the descent direction $\mathbf{d}$ is normalized; the resulting family of algorithms are known as *normalized steepest descent*.

3 Convergence rate of gradient descent

Steepest descent algorithms with exact (and, under certain technical conditions, inexact) line search are known to have linear asymptotic convergence rate. Reminder: this means that for $k \to \infty$, there exists $c > 0$ such that

$$\frac{f(\mathbf{x}_{k+1}) - f(\mathbf{x}^*)}{f(\mathbf{x}_k) - f(\mathbf{x}^*)} \leq c.$$

The constant c is called the convergence rate of the algorithm. It depends on the *condition number* of the Hessian at the solution point, defined as

$$\beta = \frac{\lambda_{\max}}{\lambda_{\min}},$$

where $\lambda_{\min}$ and $\lambda_{\max}$ are, respectively, the smallest and the largest eigenvalues of $\nabla^2 f(\mathbf{x}^*)$.

To see this dependence, let us consider a simple particular example of a quadratic function $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{H}\mathbf{x}$ with a symmetric positive definite $\mathbf{H}$. It is easy to see that $\mathbf{x}^* = \mathbf{0}$, and $f(\mathbf{x}^*) = 0$. For a current point $\mathbf{x}$, the steepest descent direction (in the ℓ_2 sense) is given by $\mathbf{d} = -\nabla f(\mathbf{x}) = -\mathbf{H}\mathbf{x}$. Making an α step in this direction yields

$$\begin{aligned}
\varphi(\alpha) &= f(\mathbf{x} + \alpha\mathbf{d}) = \frac{1}{2}(\mathbf{x} + \alpha\mathbf{d})^{\mathrm{T}}\mathbf{H}(\mathbf{x} + \alpha\mathbf{d}) \\
&= \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{H}\mathbf{x} + \alpha\mathbf{d}^{\mathrm{T}}\mathbf{H}\mathbf{x} + \frac{\alpha^2}{2}\mathbf{d}^{\mathrm{T}}\mathbf{H}\mathbf{d} \\
&= f(\mathbf{x}) + \alpha\mathbf{d}^{\mathrm{T}}\mathbf{H}\mathbf{x} + \frac{\alpha^2}{2}\mathbf{d}^{\mathrm{T}}\mathbf{H}\mathbf{d} \\
&= f(\mathbf{x}) - \alpha\mathbf{d}^{\mathrm{T}}\mathbf{d} + \frac{\alpha^2}{2}\mathbf{d}^{\mathrm{T}}\mathbf{H}\mathbf{d}.
\end{aligned}$$

The minimum of $\varphi(\alpha)$ is obtained by demanding

$$0 = \varphi'(\alpha) = \mathbf{d}^{\mathrm{T}}\mathbf{d} + \alpha\mathbf{d}^{\mathrm{T}}\mathbf{H}\mathbf{d},$$

from where

$$\alpha = \frac{\mathbf{d}^{\mathrm{T}}\mathbf{d}}{\mathbf{d}^{\mathrm{T}}\mathbf{H}\mathbf{d}}.$$

The next point is therefore given by $\mathbf{x}' = \mathbf{x} + \alpha\mathbf{d}$, and the value of f at is by

$$\begin{aligned}
f(\mathbf{x}') &= f(\mathbf{x}) - \alpha\mathbf{d}^{\mathrm{T}}\mathbf{d} + \frac{\alpha^2}{2}\mathbf{d}^{\mathrm{T}}\mathbf{H}\mathbf{d} \\
&= f(\mathbf{x}) - \frac{1}{2}\frac{(\mathbf{d}^{\mathrm{T}}\mathbf{d})^2}{\mathbf{d}^{\mathrm{T}}\mathbf{H}\mathbf{d}}.
\end{aligned}$$

Therefore,

$$\begin{aligned}\frac{f(\mathbf{x}')-f(\mathbf{x}^*)}{f(\mathbf{x})-f(\mathbf{x}^*)} &= \frac{f(\mathbf{x})-\frac{1}{2}\frac{(\mathbf{d}^{\mathrm{T}}\mathbf{d})^2}{\mathbf{d}^{\mathrm{T}}\mathbf{H}\mathbf{d}}}{f(\mathbf{x})} \\ &= 1-\frac{\frac{1}{2}\frac{(\mathbf{d}^{\mathrm{T}}\mathbf{d})^2}{\mathbf{d}^{\mathrm{T}}\mathbf{H}\mathbf{d}}}{\frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{H}\mathbf{x}} = 1-\frac{\frac{(\mathbf{d}^{\mathrm{T}}\mathbf{d})^2}{\mathbf{d}^{\mathrm{T}}\mathbf{H}\mathbf{d}}}{(\mathbf{H}\mathbf{x})^{\mathrm{T}}\mathbf{H}^{-1}\mathbf{H}\mathbf{x}} = 1-\frac{\mathbf{d}^{\mathrm{T}}\mathbf{d}}{\mathbf{d}^{\mathrm{T}}\mathbf{H}\mathbf{d}}\frac{\mathbf{d}^{\mathrm{T}}\mathbf{d}}{\mathbf{d}^{\mathrm{T}}\mathbf{H}^{-1}\mathbf{d}}.\end{aligned}$$

Observing the latter ratios, we note that for every $\mathbf{d}$,

$$\frac{\mathbf{d}^{\mathrm{T}}\mathbf{H}\mathbf{d}}{\mathbf{d}^{\mathrm{T}}\mathbf{d}} \leq \lambda_{\max}(\mathbf{H})$$

and

$$\frac{\mathbf{d}^{\mathrm{T}}\mathbf{H}^{-1}\mathbf{d}}{\mathbf{d}^{\mathrm{T}}\mathbf{d}} \leq \lambda_{\max}(\mathbf{H}^{-1}) = \frac{1}{\lambda_{\min}(\mathbf{H})},$$

implying

$$\frac{\mathbf{d}^{\mathrm{T}}\mathbf{d}}{\mathbf{d}^{\mathrm{T}}\mathbf{H}\mathbf{d}}\frac{\mathbf{d}^{\mathrm{T}}\mathbf{d}}{\mathbf{d}^{\mathrm{T}}\mathbf{H}^{-1}\mathbf{d}} \geq \frac{\lambda_{\min}(\mathbf{H})}{\lambda_{\max}(\mathbf{H})} = \frac{1}{\beta}.$$

The result of this example can be generalized into

Property. *Steepest descent algorithms have linear asymptotic convergence rate* $c = 1 - \dfrac{1}{\beta}$, *where* β *is the condition number of the Hessian at the solution point.*

Matrices with high condition number are called *ill-conditioned*. Steepest descent converges very slowly for minimization problems with an ill-conditioned Hessian at the solution point. For example, if $\beta = 10^3$, $c = 0.999$. This means that over 2000 iterations will be required to decrease the optimality gap $f(\mathbf{x}_k) - f(\mathbf{x}^*)$ ten times! The algorithm will be zigzaging very slowly toward the minimum. On the other hand, if $\beta = 1$, the gradient always points toward the minimum and the algorithm will converge in one iteration.

4 Preconditioning

The dependence of the convergence rate on the condition number of the Hessian brings the question whether there is a way to modify the problem to improve its condition number. One of such changes is the change of the coordinate system. Let use define a new optimization variable $\mathbf{x} = \mathbf{P}\mathbf{y}$, with $\mathbf{P}$ being a regular matrix. We define a new objective $\overline{f}(\mathbf{y}) = f(\mathbf{P}\mathbf{x})$. Then, using the chain rule, we have

$$\begin{aligned}\nabla\overline{f}(\mathbf{y}) &= \mathbf{P}^{\mathrm{T}}\nabla f(\mathbf{P}\mathbf{y}) = \mathbf{P}^{\mathrm{T}}\nabla f(\mathbf{x}) \\ \nabla^2\overline{f}(\mathbf{y}) &= \mathbf{P}^{\mathrm{T}}\nabla^2 f(\mathbf{P}\mathbf{y})\mathbf{P} = \mathbf{P}^{\mathrm{T}}\nabla^2 f(\mathbf{x})\mathbf{P}.\end{aligned}$$

Note that the Hessian at the solution point $\mathbf{x}^*$, $\overline{\mathbf{H}}(\mathbf{x}^*) = \mathbf{P}^{\mathrm{T}}\mathbf{H}(\mathbf{x}^*)\mathbf{P}$ depends on the choice of the coordinate system! We can choose such a $\mathbf{P}$ that improves it condition number, ideally making $\mathbf{P}^{\mathrm{T}}\mathbf{H}(\mathbf{x}^*)\mathbf{P} = \mathbf{I}$. The selection of such a transformation is called *preconditioning*. In some problems, a good preconditioning transformation can be inferred from the structure of the problem.

A step of the gradient descent method in the new coordinate system will look like

$$\mathbf{y}_k = \mathbf{y}_{k-1} - \alpha\nabla\overline{f}(\mathbf{y}_{k-1}),$$

which can be translated back to the original system as

$$\mathbf{x}_k = \mathbf{P}\mathbf{y}_k = \mathbf{x}_{k-1} - \alpha\mathbf{P}\nabla\overline{f}(\mathbf{y}_{k-1}) = \mathbf{x}_{k-1} - \alpha\mathbf{P}\mathbf{P}^{\mathrm{T}}\nabla f(\mathbf{x}_{k-1}).$$

5 Newton's method

Note that for every C^2 function, a positive-definite Hessian $\mathbf{H}(\mathbf{x}^*)$ can be written as $\mathbf{H}(\mathbf{x}^*) = \mathbf{R}^{\mathrm{T}}\mathbf{R}$ (the matrix $\mathbf{R}$ is usually called the symmetric *square root* of $\mathbf{H}$). Substituting this decomposition and demanding $\overline{\mathbf{H}} = \mathbf{P}^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}\mathbf{R}\mathbf{P} = \mathbf{I}$, we conclude that $\mathbf{P} = \mathbf{R}^{-1} = \mathbf{H}^{-1/2}(\mathbf{x}^*)$. In other words, there exists an optimal coordinate system in which the gradient descent step

$$\mathbf{x}_k = \mathbf{x}_{k-1} - \alpha\mathbf{P}\mathbf{P}^{\mathrm{T}}\nabla f(\mathbf{x}_{k-1}) = \mathbf{x}_{k-1} - \alpha\mathbf{H}^{-1}(\mathbf{x}^*)\nabla f(\mathbf{x}_{k-1})$$

has the fastest asymptotic convergence.

Note, however, that this "ideal" preconditioning requires the knowledge of the Hessian at the solution point, which is often impractical. Hence, the best we can do is to substitute $\mathbf{H}^{-1}(\mathbf{x}^*)$ with $\mathbf{H}^{-1}(\mathbf{x}_{k-1})$; sufficiently close to the minimum, the two matrices will be very similar, making the condition number of the problem close to 1. The resulting steepest descent algorithm is called *Newton's method*; at every iteration, the descent direction is selected according to

$$\mathbf{d} = -\mathbf{H}^{-1}\mathbf{g},$$

where $\mathbf{g}$ and $\mathbf{H}$ are the gradient and the Hessian at the current point. In other words, the descent direction is the solution of the linear system

$$\mathbf{H}\mathbf{d} = -\mathbf{g},$$

often called the *Newton system*.

Another way to conceive Newton's method is through quadratic interpolation of the objective. From second-order Taylor expansion around the current point $\mathbf{x}$,

$$f(\mathbf{x}+\mathbf{d}) \approx q(\mathbf{d}) = f(\mathbf{x}) + \mathbf{g}^{\mathrm{T}}\mathbf{d} + \frac{1}{2}\mathbf{d}^{\mathrm{T}}\mathbf{H}\mathbf{d}.$$

Note that the approximation $q(\mathbf{d})$ is quadratic in $\mathbf{d}$, and is convex iff $\mathbf{H} \succeq 0$. We can obtain a closed-form expression for its minimizer by demanding

$$0 = \nabla q(\mathbf{d}) = \mathbf{g} + \mathbf{H}\mathbf{d},$$

which yields precisely the Newton system.

Yet another way to conceive Newton's method is by casting the minimization of $f(\mathbf{x})$ as finding the root of the vector-valued function $\nabla f(\mathbf{x})$ or, in other words, solving the non-linear system of equations

$$\mathbf{g}(\mathbf{x}) = \mathbf{0}.$$

An iterative algorithm for solving (a one dimensional version of) this system was originally proposed by Isaac Newton himself (hence, the name). The system is linearized around a current point $\mathbf{x}$, and the linear system is solved instead. The solution of the linear system servers as the new point, around which the nonlinear system is linearized again, and the process is repeated until convergence. The linearization can be obtained from the first-order Taylor expansion of $\mathbf{g}(\mathbf{x})$,

$$\mathbf{g}(\mathbf{x} + \mathbf{d}) = \mathbf{g}(\mathbf{x}) + \mathbf{H}\mathbf{d};$$

its root is given again precisely by the solution of the Newton system.

Several important facts are important to mention in relation to Newton's method. First, for a quadratic objective, it converges in one iteration. Second, there is no way to better precondition it – it is, unlike regular steepest descent, invariant to any affine transformation of the coordinate system (prove it!).

Exercise 1. *Derive the Newton update step under an affine transformation of the coordinate system, and show that it is invariant to such transformations.*

Also, special attention is required when the objective is non-convex. Note that in order for the Newton direction to be a descent direction, we have to verify that $\mathbf{d}$ forms an acute angle with $-\mathbf{g}$,

$$0 > f'_{\mathbf{d}}(\mathbf{x}) = -\mathbf{g}^{\mathrm{T}}\mathbf{d} = \mathbf{g}^{\mathrm{T}}\mathbf{H}\mathbf{g}.$$

A sufficient condition for this to hold for any $\mathbf{g}$ is $\mathbf{H} \succ 0$, which is satisfied by a strictly convex function.

In case the latter condition is violated, the common practice is to *modify* the Hessian by shifting it spectrum by a positive diagonal matrix $\mathbf{\Delta}$, such that $\mathbf{H} + \mathbf{\Delta} \succ 0$. The new matrix is used in the *modified Newton system* $(\mathbf{H} + \mathbf{\Delta})\mathbf{d} = -\mathbf{g}$, the solution of which is guaranteed to be a descent direction.

Exercise 2. *Show that for* $\epsilon > \lambda_{\min}(\mathbf{H})$, $\mathbf{H} + \epsilon\mathbf{I} \succ 0$.

6 Numerical considerations

The solution of the Newton system is usually the most computationally demanding part of Newton's method, and it quickly becomes infeasible to directly invert the Hessian for problems bigger than $n \sim 10^3$. One of the most efficient algebraic techniques used to solve the Newton system (or, any system of equations with a positive-definite matrix) is based on *Cholesky factorization*.

Cholesky factorization of a positive-define matrix $\mathbf{H}$ is the decomposition $\mathbf{H} = \mathbf{L}\mathbf{L}^{\mathrm{T}}$, where $\mathbf{L}$ is a lower-triangular matrix. The factorization of an $n \times n$ matrix takes $\frac{n^3}{6}$ arithmetic operations. Once $\mathbf{H}$ has been decomposed into the product of lower triangular factors, the Newton system can be written as

$$\mathbf{L}\mathbf{L}^{\mathrm{T}}\mathbf{d} = -\mathbf{g}.$$

Denoting $\mathbf{y} = \mathbf{L}^{\mathrm{T}}\mathbf{d}$, we end up with the triangular system

$$\begin{pmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} -g_1 \\ \vdots \\ -g_n \end{pmatrix}.$$

The solution of such a system is straighforward through the so-called *forward substitution*: we first solve for $y_1 = -\frac{g_1}{l_{11}}$; then, substituting y_1 into the second row, we solve for y_2, and so on. Once $\mathbf{y}$ has been found, we solve the system $\mathbf{L}^{\mathrm{T}}\mathbf{d} = \mathbf{y}$ by running the process in the reverse order (*backward substitution*). Both procedures take $O(n^2)$ arithmetic operations.

In cases where $\mathbf{H}$ is not positive-definite, there exists a *modified Cholesky factorization* procedure representing $\mathbf{H}$ as $\mathbf{H} = \mathbf{L}\mathbf{L}^{\mathrm{T}} - \boldsymbol{\Delta}$, where $\boldsymbol{\Delta}$ is the smallest positive diagonal matrix guaranteeing that $\mathbf{H} + \boldsymbol{\Delta} \succ 0$.