Unconstrained optimization

©Marina Meilă
mmp@stat.washington.edu

Reading: For mathematical background, Appendix A of Boyd & Vandenberghe (BV). The notes follow mostly Bertsekas (B) Chapter 1. For the line minimization methods, look at B (ch 1) or at “Numerical recipes” (NR) chapter 10.

1 Overview

Problem Find \( \min_x f(x) \) for \( x \in \mathbb{R}^n \) or \( x \in D \) the domain of \( f \). We will assume also that \( f \) is a twice differentiable function with continuous second derivatives.

Notation The gradient of \( f \) is the column vector
\[
\nabla f(x) = \left[ \frac{\partial f}{\partial x_i}(x) \right]_{i=1}^n
\]
and the Hessian of \( f \) is the square symmetric matrix of second partial derivatives of \( f 
\]
\[
\nabla^2 f(x) = \left[ \frac{\partial^2 f}{\partial x_i \partial x_j}(x) \right]_{i,j=1}^n
\]

A local minimum for \( f \) is point \( x^* \) for which
\[ f(x^*) \leq f(x) \text{ whenever } ||x - x^*|| < \epsilon \]

A global minimum for \( f \) is point \( x^* \) for which
\[ f(x^*) \leq f(x) \text{ for all } x \text{ in the domain of } f \]
We say \( x^* \) is a **strict local/global minimum** when the above inequalities are strict for \( x \neq x^* \).

A **stationary point** for \( f \) is a point \( x^* \) for which \( \nabla f(x^*) = 0 \). A (non-singular) **local minimum** for \( f \) is a point \( x^* \) for which \( \nabla f(x^*) = 0 \) and \( \nabla^2 f(x^*) \geq 0 \) (\( \nabla^2 f(x^*) > 0 \)). In what follows, we will deal only with non-singular local minima. A non-singular local minimum is **strict** and **isolated**. A minimum is called isolated if it is the only local minimum in an \( \epsilon \)-ball around itself.

In the analysis of algorithms and practically, it is important to know if \( f \) has a **finite global minimum**; this is equivalent with \( f \) being bounded below by a constant. Otherwise, the global minimum of \( f \) is \( -\infty \) and the optimization algorithms will not converge on this problem (or will converge to other local minima).

Unconstrained optimization methods for finding a local minimum are of the form:

\[
x^{k+1} = x^k + \alpha^k d^k
\]

(3)

where \( d^k \in \mathbb{R}^n \) represents an (unnormalized) **direction** and \( \alpha^k > 0 \) is a scalar called the **step size**.

**Direction choice**

- **gradient based** \( d^k = -D^k \nabla f(x^k) \) with \( D^k \in \mathbb{R}^{n \times n} \)
  - steepest descent \( D^k = I \)
  - stochastic gradient (more about it later)
  - Newton-Raphson \( D^k = \nabla^2 f(x^k)^{-1} \)
  - conjugate gradient – implicit multistep rescaling of the axes “equivalent” to \( D^k = \nabla^2 f(x^k)^{-1} \)
  - quasi-Newton – implicit multistep approximation of \( D^k = \nabla^2 f(x^k)^{-1} \)

- **non-gradient based**
  - coordinate descent \( d^k = \) one of the basis vectors in \( \mathbb{R}^n \)
Step size choice

- line minimization $\alpha^k = \min_\alpha f(x^k + \alpha d^k)$
- Armijo rule = search but not minimization
- constant step size $\alpha^k = s$
- diminishing step size $\alpha^k \to 0; \sum_k \alpha^k = \infty$

2 How to evaluate an optimization method?

- Does it converge to a minimum?
- How fast?
- Practical issues: Is it easy to implement or tune? Available software?

As we shall see, all the methods described here converge to a minimum, but some of the require the function $f$ to have additional “good” properties.

For the second question, the answer is usually given in terms of rates of convergence, because of the general assumption that we’ll use an iterative algorithm to find the minimum.

Let $e^k = x^k - x^*$ or $e^k = f(x^k) - f(x^*)$ denote the “error” at step $k$. Then, an algorithm has a rate of convergence of order $p$ if

$$||e^{k+1}|| \leq \beta(||e^k||)^p$$

for some $0 < \beta < 1$ (4)

In the above, $p > 0$ but not necessarily an integer. However, the most common cases are $p = 1$ (linear) and $p = 2$ (quadratic). A rate of $p < 1$ is possible but is considered too slow in practice. Superlinear scales are desirable – and often achievable.

Note that the use of the term “linear” here is inconsistent with its use in e.g complexity theory. If an optimization algorithm is linear, that means that the error decreases exponentially with $k$, as $||e^k|| \leq \beta^k ||e^0||$.

In optimization problems, there are various ways of expressing the computational complexity of an algorithm:

- number of flops (floating point operations) per iteration, usually as a function of $n$ the dimension of the problem
• number of function or gradient evaluations per iteration
• number of iterations; this latter quantity is given implicitly, by the rate of convergence.
• memory requirements

With the increased complexity and variation of computer systems, the above mentioned number of operations is becoming obsolete. Algorithms are increasingly judged by other, system-related qualities, like: type of memory access (do they access memory in blocks or randomly), cache misses, etc. These criteria are beyond the scope of this course, but what you need to remember is that the textbook properties on an algorithm alone do not always predict its performance on the system you are going to run it. You may need to experiment with parameters and with algorithms to determine which algorithm is better suited for your data and system.

There are two regimes for each algorithm:

• the transitory, or approach regime, when $x^k$ is far away from $x^*$
• the asymptotic regime, near $x^*$ – most classic results are about this regime

3 Line minimization algorithms

3.1 Line minimization by the Golden Section Rule

(See also NR)

1. Bracket the minimum. Find an interval $[0, s]$ that contains the desired $\alpha$. This can be done by iteratively doubling $s$ until $f(x^k + sd^k) > f(x^k)$.

A more general definition of a bracketed minimum (not assuming that the function is decreasing at $x^k$ in the direction of $d^k$ is to find a triplet of points on the line $a = x^k$, $b$, $c$ with $b$ between $a$ and $c$ and $f(b) < f(a)$, $f(c)$. NR gives a method of finding such a triplet by starting with an initial value for $b$ and iteratively expanding the candidate triplet.
2. Find the minimum in \([0, s]\); this is the Golden Section method proper.

The golden number is the positive root of the equation

\[
\frac{R}{1} = \frac{1 - R}{R}
\]  

(5)

The reason is given by the figure below: we want to place point \(C\) in the segment \(AB\) so that

\[
\frac{AC}{AB} = \frac{BC}{AC}
\]  

(6)

**Golden Section Algorithm**

1. Start with a bracketing triplet \(x^0 < x^2 < x^1\) so that \(f(x^2) < f(x^1), f(x^0)\)
   and \(x^2 - x^0 = R(x^1 - x^0)\).
2. Choose \(x^3\) so as to divide the longest of \(x^0, x^2, x^1\) in two parts having ratio \(R\).
3. If \(f(x^3) < f(x^2)\)
   then \(x^1 \leftarrow x^2, x^2 \leftarrow x^3\) (this eliminates an interval representing a fraction \(1 - R\) of the initial interval)
   else \(x^0 \leftarrow x^3\) (this eliminates an interval representing the fraction \(R \times R = 1 - R\) of the initial interval)
4. Stop if \(x^1 - x^0 < tol\) a desired tolerance, else repeat the previous step.

Note that after each step, the interval containing the minimum shrinks by \(R = 0.681\ldots\). The value \(R\) is optimally chosen so that the reduction in step 3 of the algorithm is the same in either case. If the initial bracketing triplet is not in the golden ratio \(R\), it can be shown that the subsequent ratios will converge towards \(R\).
How small shall we make the tolerance tol be? Near the (unidimensional) minimum \( b \), 
\[
    f(x) \approx f(b) + \frac{(x-b)^2}{2} f''(b)
\]
with the second term being much smaller than the first. Hence, when this term becomes a factor of \( \epsilon \) smaller, it will be negligible when added to the first, and we may just as well stop the iteration. This happens when 
\[
    |x - b| < \sqrt{\epsilon} |b| \sqrt{\frac{2f(b)}{f''(b)b^2}}
\]  \( \tag{7} \)

The last \( \sqrt{\cdot} \) is often of order 1, therefore the above formula implies that the tolerance should be set to be of order \( \sqrt{\epsilon} |b| \) with \( \epsilon \) being the \( \epsilon \)-machine of the current implementation. For standard double precision numbers, \( \sqrt{\epsilon} \approx 3.10^{-8} \).

If the line minimization is performed without first bracketing the minimum, i.e. we look for the best \( \alpha^k \in [0, s] \) for some arbitrarily chosen \( s \), the method is called truncated minimization.

### 3.2 The Parabolic Interpolation method

If the function \( f \) is smooth then a method that assumes that will converge faster than the worst-case-safe Golden ratio rule. This is the parabolic interpolation method that assumes that the function is approximately a parabola in the interval considered.

**Parabolic Approximation Algorithm**

1. Start with a bracketing triplet \( x^0 < x^2 < x^1 \) so that \( f(x^2) < f(x^1), f(x^0) \).
2. Fit a parabola through the three points and let \( x^3, f(x^3) \) be its minimum. (We assume \( f(x^3) < f(x^1) \), otherwise this method is not useful.)
3. If \( x^3 \in (x^0, x^2) \) 
   - then \( x^2 \leftarrow x^3, x^1 \leftarrow x^2 \)
   - else \( x^0 \leftarrow x^2 \)
4. Stop if \( x^1 - x^0 \lesssim tol \) a desired tolerance, else repeat the previous step

In NR you can find the **Brent algorithm** which combines the golden section and the parabolic interpolation algorithms. It attempts to use parabolic
interpolation, but detects when this method fails to approach the minimum and switches to golden section.

3.3 The Armijo Rule

**Intuition** Assume that we found a bracketing interval \([0, s]\) for \(\alpha^k\). We start with \(\alpha^k = s\) and decrease it exponentially until we find that the function \(f\) has decreased “enough”. What is “enough”? In an infinitesimal interval near \(x^k\) along the direction of descent, the function will decrease linearly, hence

\[
f(x^k) - f(x^k + \alpha d^k) \approx \alpha(-\nabla f(x^k)^T d^k)
\]  

(8)

For a finite interval, we will ask for a decrease in \(f\) that is at least \(\sigma < 1\) smaller than the above. Note that for a sufficiently small \(\alpha\) such a decrease can always be attained.

**Armijo Line Search**

1. Start with \(\alpha^k = s\), \(\beta < 1\), \(\sigma < 1\)
2. If 
   \[
f(x^k) - f(x^k + \alpha d^k) > \sigma \alpha^k(-\nabla f(x^k)^T d^k)
   \]
   - then STOP
   - else \(\alpha^k \leftarrow \beta \alpha^k\) and repeat

In practice, \(\beta \approx 0.5\) and \(\sigma << 1\) e.g 0.1, 0.01 or even 0; \(s = 1\) if no bracketing is done.

4 Multidimensional minimization. The choice of direction

4.1 The steepest descent method

The steepest descent method follows the direction of the gradient. It can be shown [B] that gradient descent with line minimization has a linear rate of convergence. For other line search methods, including constant step size, the rate of convergence is no larger.
The convergence coefficient $\beta$ of equation (4) can get very close to 1 (very slow convergence) if the Hessian is ill conditioned. Let $M, m$ denote respectively the largest and the smallest eigenvalue of $\nabla^2 f(x^*)$. By continuity, we can assume that the Hessian around $x^*$ is approximately the same. If $M >> m$ then the function will have a “long, narrow valley” with an almost flat “bottom” around $x^*$, oriented along the smallest eigenvector. The gradient will be almost perpendicular to the valley, and the algorithm, even with the optimal line minimization, will advance very slowly. See also section 4.5 for a more precise evaluation of this effect.

Hence, all the following methods (except for stochastic gradient) can be seen as “applying some coordinate transformation” that will turn the elongated ellipses into circles, so that steepest descent in this new coordinate frame can move rapidly towards the optimum. Equivalently, having such a transformation (which is represented by the Hessian matrix), one can apply the “inverse transformation” to the descent direction, which is precisely what the Newton-Raphson method does.

### 4.2 The Newton-Raphson method

Assume that our function is quadratic, i.e

$$f(x) = \frac{1}{2} x^T A x + b^T x + c \quad \text{with} \quad A > 0.$$  \hfill (9)

Then,

$$\nabla f(x) = A x + b$$  \hfill (10)

$$\nabla^2 f(x) = A$$  \hfill (11)

and the minimum can be computed analytically as the solution of $A x + b = 0$, namely $x^* = -A^{-1} b$. Equivalently, for any $x$

$$x^* - x = -A^{-1} b - A^{-1} A x = -A^{-1} (Ax + b) = -\nabla^2 f(x)^{-1} \nabla f(x)$$  \hfill (12)

Hence, if $f$ is quadratic, from any point $x$ we can move in one step equal to $-\nabla^2 f(x)^{-1} \nabla f(x)$ to the minimum. Therefore, the Newton-Raphson method takes $D^k = \nabla^2 f(x)^{-1}$ as if the function was quadratic. Usually one also does a line search method, i.e $\alpha^k \neq 1$ in practice.
Newton-Raphson is practically and theoretically very fast once we are in the vicinity of the optimum (section 4.5 gives one result). However, its behavior far away from the optimum must be monitored carefully. Note for example that this is not a descent method, in the sense that it’s not guaranteed that $f(x^{k+1}) < f(x^k)$ unless some form of line minimization is used. Also, the method is attracted by local maxima just as much as by local minima, so attention must be paid any time the Hessian is not positive definite. See [B] for modern methods that deals with these problem (trust region method and variations of quasi-Newton methods).

Another drawback is the need to compute and store the Hessian ($\mathcal{O}(n^2)$ storage and $\mathcal{O}(n^{2-3})$ operations). Computation makes Newton-Raphson prohibitive in high dimensions.

A quick fix called **diagonal scaling**, where only the diagonal terms of the Hessian are computed.

$$D^k = \text{diag} \left\{ \frac{\partial^2 f}{\partial x_i^2} \right\}$$ (13)

This method is obviously linear in storage and number of operations but it tends to underestimate the ratio $M/m$. Diagonal scaling amounts to rescaling each variable separately, and it is effective in those cases when the variables have very different ranges because of “imbalanced” measurement units (e.g in one direction the unit is miles, in the other one it is millimeters).

It is useful to make also the general observation that the Newton method is “scale free”, i.e it is unaffected by linear coordinate changes.

**Computing the direction** Let $H = \nabla^2 f(x^k)$, $g = \nabla f(x^k)$. Computing the descent direction $d = -H^{-1}g$ by inverting the Hessian is neither the most efficient, nor the most exact method in terms of numerical error. Here are two alternatives:

1. Solve the system $Hd = -g$. The matrix $H$ being symmetric, $\succ 0$, it can be factored into a product $H = LL^T$, with $L$ lower triangular. This is called the **Cholesky factorization** and takes $\sim n^3 6$ operations. Solving an upper/lower triangular linear system is $\sim \frac{n^2}{2}$

2. Minimize the quadratic function $\frac{1}{2}d^T H d + g^T d$, whose gradient equals $Hd + g$. Of course, the minimization should be done by a method other than Newton-Raphson! A possibility is to use steepest descent, starting
from $d = -g$. Note that after every step, the current approximation $d$ of the solution is a descent direction. The number of flops per iteration for this method is $\sim \frac{n^2}{2}$.

4.3 The Conjugate gradient method

**Conjugate Gradient Algorithm**

$$x^{k+1} = x^k + \alpha^k d^k$$

where $\alpha^k$ is chosen by line minimization

1. $d^0 = -\nabla f(x^0)$
2. for $k = 1, \ldots, n-1$, $d^k = -\nabla f(x^k) + \beta^k d^{k-1}$ with

$$\beta^k = \frac{\nabla f(x^k)^T (\nabla f(x^k) - \nabla f(x^{k-1}))}{\nabla f(x^{k-1})^T \nabla f(x^{k-1})}$$  \hspace{1cm} (14)

3. restart from step 1 if $k = n$

**Intuition** If $H = \nabla^2 f(x^*) = I$ and $f$ quadratic, then the level curves of $f$ around $x^*$ would be circles. After line minimization along any direction of descent, the next direction of descent will be $\perp$ on the previous one, and in $n$ steps the min is attained.

Let

$$z = H^{1/2}(x - x^*)$$  \hspace{1cm} (15)

where the notation $A^{1/2}$ represents the **matrix square root** of the symmetric positive definite matrix $A$, i.e the matrix $B$ such that $B^T B = A$. $B$ is real whenever the eigenvalues of $A$ are non-negative; $B$ is not necessarily symmetric, and it is not unique as multiplying $B$ to the right with an orthogonal matrix produces another square root of $A$.

The usual Taylor approximation of $f$ around the minimumum gives

$$f \approx f(x^*) + (x - x^*)^T H(x - x^*) + \ldots$$  \hspace{1cm} (16)

$$= f(x^*) + z^T z + \ldots$$  \hspace{1cm} (17)

$$= f(x^*) + ||z||^2 + \ldots$$  \hspace{1cm} (18)

(19)
In the new variable $z$, the level curves are (approximately) circles. Let $\tilde{d}_{1:k}$ be orthogonal consecutive directions of descent in the $z$ space. We have then

$$z^k - z^{k-1} = \beta_k \tilde{d}_k$$  \hspace{1cm} (20)

$$H^{-1/2}(z^k - z^{k-1}) = \beta_k H^{-1/2} \tilde{d}_k$$  \hspace{1cm} (21)

$$x^k - x^{k-1} = \beta_k H^{-1/2} \tilde{d}_k$$ \hspace{1cm} (22)

The directions $d^k$ in the above satisfy $d^k H d^j = 0$ for all $k \neq j$ (this is the definition of conjugate directions). Equivalently, $H^{1/2} d^k \perp H^{1/2} d^j$, which means that these directions become orthogonal if we applied the coordinate transformation $H^{-1/2}$. But this transformation is exactly the transformation we need to turn the level curves of $f$ into circles. Hence, we can view the directions $d^k$ as the correct orthogonal direction of descent in a transformed coordinate system. Note that exact line minimization is essential for the conjugate gradient method. It is also essential to restart the method after $n$ steps. In practice one can also restart after less than $n$ steps, especially if loss of conjugacy is suspected (because of e.g inexact line minimization).

The smart thing about the method is that these directions are computed recursively, without explicitly estimating, storing or inverting the $H$ matrix, a big saving in high dimensions.

**Rates of convergence** Practical experience and exact results for quadratic functions suggest that the conjugate gradients method is superlinear. The superlinear convergence is preserved even if the method is restarted after less than $n$ steps. [B]

### 4.4 Quasi-Newton (variable metric) methods

**Idea** Let

$$p^k = x^{k+1} - x^k$$ \hspace{1cm} (23)

$$q^k = \nabla f(x^{k+1}) - \nabla f(x^k)$$ \hspace{1cm} (24)

Then

$$q^k \approx \nabla^2 f(x^{k+1}) p^k$$ \hspace{1cm} (25)
Therefore, given $n$ linearly independent pairs $(p^k, q^k)$ one can approximate the (inverse) Hessian by
\[ \nabla^2 f(x^n)^{-1} \approx [p^0 \, p^1 \, \ldots \, p^n][q^0 \, q^1 \, \ldots \, q^n]^{-1} \] (26)
Most quasi-Newton methods use this idea to recursively construct an approximation to $\nabla^2 f^{-1}$.

**Quasi-Newton Direction Update**

1. Start with any $D^0 \succ 0$
2. For $k = 1, 2, \ldots$ update
\[
D^{k+1} = D^k + \frac{p^k p^k T}{p^k T q^k} - \frac{D^k q^k q^k T D^k}{q^k T D^k q^k} + \xi^k \tau^k v^k v^k T \] (27)
\[
v^k = \frac{p^k}{p^k T q^k} - \frac{D^k q^k}{\tau^k} \] (28)
\[
\tau^k = q^k T D^k q^k \] (29)

The parameter $\xi^k \in [0, 1]$; for $\xi^k = 1$ one obtains the BFGS (Broyden-Fletcher-Goldfarb-Shanno) method, which is considered the best general purpose quasi-Newton method.

**Proposition 1** If $f$ is quadratic as in (9) and we run the quasi-Newton method with line minimization for $n$ steps, then
(1) the vectors $d^0, d^1, \ldots, d^{n-1}$ are $A$-conjugate
(2) $D^n = A^{-1}$

**Advantages and disadvantages** The main disadvantage of the quasi-Newton methods is their large computational complexity compared to the conjugate directions methods ($n^2$ versus $n$ and versus $n^3$ for Newton-Raphson). The advantage is that the method is robust to inexact line minimization, and that it does not need restarts. In addition, near the optimum, the directions quasi-Newton generates are conjugate and so it is equivalent to conjugate gradients and Newton-Raphson and thus converges very fast.
4.5 Convergence and rates for gradient based methods

We say that a direction (sequence) is \textit{gradient related}, if for any sequence \((x^k)_k\) such that \(x^k \rightarrow x\) with \(x\) non-stationary, the corresponding sequence of directions \((d^k)_k\) is bounded and satisfies

\[
\limsup_{k \rightarrow \infty} \nabla f(x^k)^T d^k < 0 \quad (30)
\]

This condition ensures that the search directions do not become, in the limit, orthogonal to the gradient, nor 0, unless we approach a stationary point.

**Proposition 2** A direction \(d^k = -D^k \nabla f(x^k)\) is gradient related if for every eigenvalue \(\lambda\) of \(D^k\) we have

\[
0 < m \leq \lambda \leq M
\]

for some constants \(m, M\).

The first question is: do the methods described here converge to stationary points (or minima) of \(f\)? We shall see that the answer is yes, given that the direction choice is gradient related (always so for steepest descent) modulo some stronger or weaker assumptions. The “safest” methods (assuming the least) require line minimization, while having a constant step size requires the strongest assumptions.

**Proposition 3 Convergence of line search** Let \(x^{k+1} = x^k + \alpha_k d^k\) be a sequence generated by a gradient method with \(\{d^k\}\) gradient related and \(\alpha_k\) generated by the line minimization rule, truncated line minimization or the Armijo rule. Then every limit point of \(\{x^k\}\) is a stationary point.

**Proposition 4 Convergence of constant step methods** Let \(x^{k+1} = x^k + \alpha^k d^k\) be a sequence generated by a gradient method with \(\{d^k\}\) gradient related. Assume that for some constant \(L > 0\) we have

\[
||\nabla f(x) - \nabla f(y)|| \leq L||x - y|| \quad \text{for all} \; x, y \in \mathbb{R}^n \quad (i.e. \; \nabla f \text{ is Lipschitz}) \quad (31)
\]

and that for all \(k\) we have \(d^k \neq 0\) and for some fixed \(\epsilon > 0\)

\[
\epsilon \leq \alpha^k \leq (2 - \epsilon)\bar{\alpha}^k \quad \text{where} \quad \bar{\alpha}^k = \frac{||\nabla f(x^k)^T d^k||}{L||d^k||^2} \quad (32)
\]

Then every limit point of \(\{x^k\}\) is a stationary point.
Proposition 5 Convergence for diminishing step size Let \( x^{k+1} = x^k + \alpha^k d^k \) be a sequence generated by a gradient method with \( \{d^k\} \) gradient related and \( \alpha^k \rightarrow 0, \sum_k \alpha^k = \infty \). Assume that for some constant \( L > 0 \) we have
\[
||\nabla f(x) - \nabla f(y)|| \leq L||x - y|| \quad \text{for all } x, y \in \mathbb{R}^n \quad (i.e. \nabla f \text{ is Lipschitz}) \quad (33)
\]
and that there exist \( c_1, c_2 > 0 \) so that for all \( k \) we have
\[
c_1||d^k||^2 \leq ||\nabla f(x^k)||^2 \leq c_2(-\nabla f(x^k))^T d^k \quad (34)
\]
Then either \( f(x^k) \rightarrow -\infty \) or \( f(x^k) \) converges to a finite value and \( \nabla f(x^k) \rightarrow 0 \). (Consequently, every limit point of \( \{x^k\} \) is a stationary point for \( f \)).

These results guarantee that the values \( x^k \) for the above methods converge to a unique stationary point under fairly mild conditions, given below.

Proposition 6 Capture theorem Let \( f \) be continuously differentiable, let \( \{x^k\} \) be a sequence generated by a gradient method so that all its limit points are stationary points of \( f \). Assume there exist \( s, c > 0 \) so that
\[
\alpha^k \leq s, \quad ||d^k|| \leq c||\nabla f(x^k)||, \quad f(x^{k+1}) \leq f(x^k) \quad (35)
\]
Then, for any isolated stationary point \( x^* \) which is a local minimum, there is an open set \( S \) so that, if \( x^k \in S \) for some \( k \), then \( x^k \in S \) for all \( k > k \) and \( x^k \rightarrow x^* \).

Now we shall see some results about convergence rates. Here, the conclusions are that steepest descent is linear (at best), while Newton-Raphson and the methods that attempt to approximate the Hessian are typically superlinear.

Proposition 7 (B) Assume that \( \alpha^k \) is chosen by the line minimization rule and that \( x^k \rightarrow x^* \) with \( \nabla f(x^*) = 0, \nabla^2 f(x^*) \succ 0 \). Let \( Q^k = \sqrt{D^k} \nabla^2 f(x^k)^{-1} \sqrt{D^k} \) and let \( m^k, M^k \) be the smallest, respectively largest eigenvalue of \( Q^k \) (assuming that \( D^k, \nabla^2 f(x^k) \succ 0 \)). Then,
\[
\alpha^k_{opt} = \frac{2}{M^k + m^k} \quad (36)
\]
and
\[
\limsup_{k \rightarrow \infty} \frac{f(x^{k+1}) - f(x^*)}{f(x^k) - f(x^*)} \leq \limsup_{k \rightarrow \infty} \left( \frac{M^k - m^k}{M^k + m^k} \right)^2 \quad (37)
\]
One can show that the above bound is tight for steepest descent.

**Proposition 8 (B)** The Newton-Raphson algorithm converges superlinearly Assume $x^k \to x^*$, $\nabla f(x^*) = 0$, $\nabla^2 f(x^*) \succ 0$ and

$$
\lim_{k \to \infty} \frac{||d^k + \nabla^2 f(x^k)^{-1}\nabla f(x^k)||}{||\nabla f(x^k)||} = 0
$$

(38)

and $\alpha^k$ is chosen by the Armijo rule, with $s = 1$, $\sigma < 1/2$. Then,

$$
\lim_{k \to \infty} \frac{||x^{k+1} - x^*||}{||x^k - x^*||} = 0
$$

(39)

5 Noisy gradient and no gradient methods

These are the “cheap and slow” methods which can however be useful too. One should not confuse “theoretically slow” with “slow in practice” and on some problems the latter is true of the simpler methods. On other occasions, these methods perform well because they make fewer assumptions about the smoothness of the surface $f(x)$.

5.1 Stochastic gradient methods

This class of methods typically includes steepest descent methods with diminishing step size. It is assumed that the gradient is computed with some error. A typical situation is fitting models to data, especially in on-line procedures. In that case, the error in the gradient is assumed to have 0 mean and finite variance.

The following example will illustrate this procedure. Let $D_N$ be an i.i.d sample of size $n$ from an unknown distribution. Denote by $f(\theta) = \ln P(D_N | \theta)$ the negative log-likelihood to be minimized. Because the sample is i.i.d., $f(\theta) = -\sum_{i=1}^{N} \ln p(x^i | \theta)$. Since $f$ is a sum, so will be the gradient:

$$
\nabla f(\theta) = -\sum_{i=1}^{N} \frac{\partial p(x^i | \theta)}{\partial \theta} \frac{p(x^i | \theta)}{p(x^i | \theta)}
$$

(40)
If \( N \) is large, something desirable from the statistical point of view, then the computation of the gradient is very costly (linear in \( N \)). A practical option is to take

\[
d^k = \sum_{i \in S^k} \frac{\partial p(x^i|\theta)}{\partial \theta} p(x^i|\theta)
\]

(41)

where \( S^k \subseteq D_N \) is a small subset of the data. Typical choices for \( S^k \) are:

- partition \( D_N \) into \( M \) subsets of size \( N/M \) and cycle (pick a random one) at each iteration
- fix the size \( N' << N \) of \( S^k \) and sample a random subset of size \( N' \)
- set \( N' = 1 \) and cycle through the data or pick a random observation at each iteration

Note that this method can naturally be adapted to on-line learning; i.e situations where data come one by one, and are not stored but used immediately to update the parameters, then discarded.

It has been proved under various technical conditions that stochastic gradient methods converge to the true value of the parameters if \( \sum k \alpha^k = \infty, \sum k (\alpha^k)^2 < \infty \) (the latter implies \( \alpha^k \to 0 \)) and the noise variance is bounded. Essentially, for convergence the \( \alpha^k \)'s should decrease like \( \frac{1}{k} \). Note however that typically in practice the decrease needs to be very slow, almost constant e.g \( \frac{1}{b+k/c} \) with \( b, c \) large numbers.

The convergence is slow (typically sublinear). The increase in the number of steps to convergence is (hopefully!) balanced by the speedup in gradient evaluation, which can be of orders of magnitude. In particular, the choice of \( N' \) is independent from \( N \) for very large \( N \). The computation increases like \( N' \), while the inverse variance of the gradient error decreases like \( \sqrt{N'} \) which indicates that a trade-off point for \( N' \) exists.

It is also not necessary to keep \( N' \) fixed throughout the algorithm. In the early stages of learning, when approaching the optimum at high speed is important, one can use a small \( N' \), while in later steps, i.e once the asymptotic regime is attained, one may find it advantageous to increase \( N' \) or even to use the exact gradient.

Stochastic gradient and analog techniques are widely used in machine learn-
ing: training of neural networks, reinforcement learning (the TD-\(\lambda\) and Q-learning procedures are stochastic gradient methods), speedup of boosting.

5.2 Numerical evaluation of the gradient

Sometimes, if the gradient is expensive to compute (and \(f(x)\) is presumably not), the gradient can be evaluated by finite differences. The forward difference is

\[
\frac{\partial f}{\partial x_i}(x^k) \approx \frac{f(x^k + hu_i) - f(x^k)}{h}
\]

and the symmetric difference is

\[
\frac{\partial f}{\partial x_i}(x^k) \approx \frac{f(x^k + hu_i) - f(x^k - hu_i)}{2h}
\]

The symmetric difference involves twice as much computations, but it is significantly more stable numerically (the rates of convergence are respectively \(O(h)\) for the forward method and \(O(h^2)\) for the symmetric method).

It is important to remember that \(h\) need not be extremely small (remember the \(\epsilon_{machine}\) limit!), but that it is good to choose the best \(h\) for each coordinate separately. This is because different coordinate axes may have different measurement units, and rates of variation of \(\nabla f\) and it may be impossible to find one \(h\) which is acceptable for all.

5.3 Coordinate descent

Here, the direction of descent \(d^k\) is always one of the coordinate axes \(u_i^k\). Hence \(x^{k+1} = x^k + \alpha_k u_i^k\). Note that line search is necessary, and that the minimum can be on either side of \(x^k\) so \(\alpha_k\) can take negative values.

Theoretical and empirical results suggest that coordinate descent has similar convergence properties as the steepest descent method (i.e linear in the best case).

While in a general case coordinate descent is suboptimal, there are several situations when it is worth considering
1. When line minimization can be done analytically. This can save one 
the often expensive gradient computation.
2. When the coordinate axes affect the function value approximately inde- 
pendently, or (in statistics) when the coordinate axes are uncorrelated.
Then minimizing along each axis separately is (nearly) optimal.
3. When there exists a natural grouping of the variables. Then one can op-
timize one group of variables while keeping the other constant. Again, 
we hope that the groups are “independent”, or that optimizing one 
group at a time can be done analytically, or it’s much easier than 
computing the gradient w.r.t all variables simultaneously. This idea is 
the basis of many alternate minimization methods, including the well 
known EM algorithm.

5.4 The simplex method

This is a method that does not evaluate derivatives. (Not to be confused 
with the Simplex algorithm of linear programming!) A set of \( n - 1 \) points is 
maintained, and iterative replacement of the worst of them aims to move the 
simplex so that it encloses the optimum, and to simultaneously shrink it so 
that the optimum is bracketed with small tolerance.

\[ \text{Simplex method} \]

\begin{itemize}
  \item \textbf{Initialize} \( x_1, \ldots, x_{n+1} \) points in general position (i.e that enclose a non-
  zero volume)
  \item Denote \( x_{\text{max}}, f_{\text{max}}, f_{\text{min}} \) respectively \( \arg\max_i f(x_i), \max_i f(x_i), \min_i f(x_i) \), 
    and \( \hat{x} \) the mean of \( \{x_{1:n+1}\} \setminus x_{\text{max}} \).
  \item \textbf{Reflection} Compute \( x_{\text{ref}} = \hat{x} + (\hat{x} - x_{\text{max}}) \) and \( f_{\text{ref}} = f(x_{\text{ref}}) \).
  \item If \( f_{\text{ref}} > f_{\text{max}} \), \( x_{\text{new}} = \arg\min\left(f((x_{\text{ref}} + \hat{x})/2), f((x_{\text{max}} + \hat{x})/2)\right) \) and go 
    to 5
  \item If \( f_{\text{min}} < f_{\text{ref}} \leq f_{\text{max}} \), \( x_{\text{new}} = x_{\text{ref}} \) and go to 5
  \item If \( f_{\text{min}} \geq f_{\text{ref}} \) try to expand
    \( x_{\text{exp}} = x_{\text{ref}} + (x_{\text{ref}} - \hat{x}), x_{\text{new}} = \arg\min(f_{\text{ref}}, f(x_{\text{exp}})) \).
  \item replace \( x_{\text{max}} \) with \( x_{\text{new}} \), then recompute \( f_{\text{min}}, x_{\text{max}}, f_{\text{max}} \)
\end{itemize}
The algorithm is not guaranteed to converge in all cases, but it can be shown that it works for convex $f$. The above is not the most efficient implementation, it’s the one that’s easier to read. Consult also NR for this algorithm.

6 Stopping criteria

**Proposition 9** If the smallest eigenvalue of $\nabla^2 f(x) \geq m > 0$ and $||\nabla f(x)|| \leq \epsilon$ for $x \in S$ a neighborhood of $x^*$, then

$$||x - x^*|| \leq \frac{\epsilon}{m} \quad f(x) - f(x^*) \leq \frac{\epsilon^2}{m}$$

(44)

So, knowing when to stop involves assumptions (or information) about (1) the gradient, (2) the Hessian (its eigenvalues), (3) the continuity of the Hessian. In practice, a good rule of thumb is

1. check that $\nabla^2 f \succ 0$ and, if feasible, that the third derivatives are not too high
2. if you care about the value $f(x^*)$, stop when the relative decrease $\left|\frac{1 - f(x^{k+1})}{f(x^*)}\right| < tol$ with $tol \geq \epsilon_{\text{machine}}$ or use the above rule if $m$ available
3. if you care about $x^*$ then either use the above rule (if some estimate of $m$ is available) or bound the relative change ($||x^{k+1} - x^{k}|| \leq tol$) or use the simple rule

$$||\nabla^2 f(x^{k})^{-1}\nabla f(x^{k})|| \leq tol$$

(45)

The above l.h.s represents the distance to $x^*$ if the function was quadratic. Alternatively, assuming that the direction of search incorporates a “good” variable scaling, (i.e NOT in steepest descent) one can stop when

$$||d^k|| \leq tol$$

(46)

If the Hessian is too expensive to compute, then you can save time by estimating it only every $L$ iterations, assuming that its values don’t change much at one iteration. (This holds for the Newton steps as well, not only for the stopping test).