8 Newton's method for minimization

Again, we want to solve

$$\text{(P)} \quad \begin{array}{l} \min f(x) \\ x \in \mathbb{R}^n. \end{array}$$

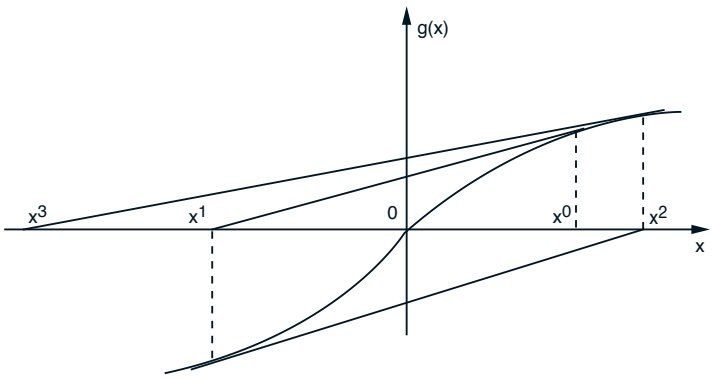
The Newton's method can also be interpreted in the framework of the general optimization algorithm, but it truly stems from the Newton's method for solving systems of nonlinear equations. Recall that if $g : \mathbb{R}^n \to \mathbb{R}^n$, to solve the system of equations

$$g(x) = 0,$$

one can apply an iterative method. Starting at a point x_0, approximate the function g by $g(x_0+d) \approx g(x_0) + \nabla g(x_0)^T d$, where $\nabla g(x_0)^T \in \mathbb{R}^{n\times n}$ is the Jacobian of g at x_0, and provided that $\nabla g(x_0)$ is non-singular, solve the system of linear equations

$$\nabla g(x_0)^T d = -g(x_0)$$

to obtain d. Set the next iterate $x_1 = x_0 + d$, and continue. This method is well-studied, and is well-known for its good performance when the starting point x_0 is chosen appropriately. However, for other choices of x_0 the algorithm may not converge, as demonstrated in the following well-known picture:



The Newton's method for minimization is precisely an application of this equation-solving method to the (system of) first-order optimality conditions $\nabla f(x) = 0$. As such, the algorithm does not distinguish between local minimizers, maximizers, or saddle points.

Here is another view of the motivation behind the Newton's method for optimization. At $x = \bar{x}$, $f(x)$ can be approximated by

$$f(x) \approx q(x) \stackrel{\triangle}{=} f(\bar{x}) + \nabla f(\bar{x})^T (x - \bar{x}) + \frac{1}{2}(x - \bar{x})^T H(\bar{x})(x - \bar{x}),$$

which is the quadratic Taylor expansion of $f(x)$ at $x = \bar{x}$. $q(x)$ is a quadratic function which, if it is convex, is minimized by solving $\nabla q(x) = 0$, i.e., $\nabla f(\bar{x}) + H(\bar{x})(x - \bar{x}) = 0$, which yields

$$x = \bar{x} - H(\bar{x})^{-1} \nabla f(\bar{x}).$$

The direction $-H(\bar{x})^{-1}\nabla f(\bar{x})$ is called the *Newton direction*, or the *Newton step*.

This leads to the following algorithm for solving (P):

Newton's Method:

Step 0 Given x_0, set $k \leftarrow 0$

Step 1 $d_k = -H(x_k)^{-1}\nabla f(x_k)$. If $d_k = 0$, then stop.

Step 2 Choose stepsize $\lambda_k = 1$.

Step 3 Set $x_{k+1} \leftarrow x_k + \lambda_k d_k,\ k \leftarrow k+1$. Go to **Step 1**.

Proposition 17 *If $H(x) \succ 0$, then $d = -H(x)^{-1}\nabla f(x)$ is a descent direction.*

Proof: It is sufficient to show that $\nabla f(x)^T d = -\nabla f(x)^T H(x)^{-1}\nabla f(x) < 0$. Since $H(x)$ is positive definite, if $v \neq 0$,

$$0 < (H(x)^{-1}v)^T H(x)(H(x)^{-1}v) = v^T H(x)^{-1} v,$$

completing the proof. ∎

Note that:

- Work per iteration: $O(n^3)$
- The iterates of Newton's method are, in general, equally attracted to local minima and local maxima. Indeed, the method is just trying to solve the system of equations $\nabla f(x) = 0$.
- There is no guarantee that $f(x_{k+1}) \leq f(x_k)$.
- Step 2 could be augmented by a linesearch of $f(x_k + \lambda d_k)$ over the value of λ; then previous consideration would not be an issue.
- The method assumes $H(x_k)$ is nonsingular at each iteration. Moreover, unless $H(x_k)$ is positive definite, d_k is not guaranteed to be a descent direction.
- What if $H(x_k)$ becomes increasingly singular (or not positive definite)? Use $H(x_k) + \epsilon I$.
- In general, points generated by the Newton's method as it is described above, may not converge. For example, $H(x_k)^{-1}$ may not exist. Even if $H(x)$ is always non-singular, the method may not converge, unless started "close enough" to the right point.

Example 1: Let $f(x) = 7x - \ln(x)$. Then $\nabla f(x) = f'(x) = 7 - \frac{1}{x}$ and $H(x) = f''(x) = \frac{1}{x_2}$. It is not hard to check that $x^\star = \frac{1}{7} = 0.142857143$ is the unique global minimizer. The Newton direction at x is

$$d = -H(x)^{-1}\nabla f(x) = -\frac{f'(x)}{f''(x)} = -x_2\left(7 - \frac{1}{x}\right) = x - 7x_2,$$

and is defined so long as $x > 0$. So, Newton's method will generate the sequence of iterates $\{x_k\}$ with $x_{k+1} = x_k + (x_k - 7(x_k)^2) = 2x_k - 7(x_k)^2$. Below are some examples of the sequences generated

by this method for different starting points:

k	x_k	x_k	x_k
0	1	0.1	0.01
1	-5	0.13	0.0193
2		0.1417	0.03599257
3		0.14284777	0.062916884
4		0.142857142	0.098124028
5		0.142857143	0.128849782
6			0.1414837
7			0.142843938
8			0.142857142
9			0.142857143
10			0.142857143

(note that the iterate in the first column is not in the domain of the objective function, so the algorithm has to terminate...).

Example 2: $f(x) = -\ln(1 - x_1 - x_2) - \ln x_1 - \ln x_2$.

$$\nabla f(x) = \begin{bmatrix} \frac{1}{1-x_1-x_2} - \frac{1}{x_1} \\ \frac{1}{1-x_1-x_2} - \frac{1}{x_2} \end{bmatrix},$$

$$H(x) = \begin{bmatrix} \left(\frac{1}{1-x_1-x_2}\right)^2 + \left(\frac{1}{x_1}\right)^2 & \left(\frac{1}{1-x_1-x_2}\right)^2 \\ \left(\frac{1}{1-x_1-x_2}\right)^2 & \left(\frac{1}{1-x_1-x_2}\right)^2 + \left(\frac{1}{x_2}\right)^2 \end{bmatrix}.$$

$x^\star = \left(\frac{1}{3}, \frac{1}{3}\right)$, $f(x^\star) = 3.295836866$.

k	$(x_k)_1$	$(x_k)_2$	$\|x_k - \bar{x}\|$
0	0.85	0.05	0.58925565098879
1	0.717006802721088	0.0965986394557823	0.450831061926011
2	0.512975199133209	0.176479706723556	0.238483249157462
3	0.352478577567272	0.273248784105084	0.0630610294297446
4	0.338449016006352	0.32623807005996	0.00874716926379655
5	0.333337722134802	0.333259330511655	$7.41328482837195e^{-5}$
6	0.333333343617612	0.33333332724128	$1.19532211855443e^{-8}$
7	0.333333333333333	0.333333333333333	$1.57009245868378e^{-16}$

Termination criteria Since Newton's method is working with the Hessian as well as the gradient, it would be natural to augment the termination criterion we used in the Steepest Descent algorithm with the requirement that $H(x_k)$ is positive semi-definite, or, taking into account the potential for the computational errors, that $H(x_k) + \epsilon I$ is positive semi-definite for some $\epsilon > 0$ (this parameter may be different than the one used in the condition on the gradient).

8.1 Convergence analysis of Newton's method

8.1.1 Rate of convergence

Suppose we have a *converging* sequence $\lim_{k\to\infty} s_k = \bar{s}$, and we would like to characterize the speed, or rate, at which the iterates s_k approach the limit $\bar{s}$.

A converging sequence of numbers $\{s_k\}$ exhibits *linear* convergence if for some $0 \leq C < 1$,

$$\limsup_{k\to\infty} \frac{|s_{k+1} - \bar{s}|}{|s_k - \bar{s}|} = C.$$

"$\limsup_{k\to\infty}$" denotes the largest of the limit points of a sequence (possibly infinite). C in the above expression is referred to as the *rate constant*; if $C = 0$, the sequence exhibits *superlinear* convergence.

A sequence of numbers $\{s_k\}$ exhibits *quadratic* convergence if it converges to some limit $\bar{s}$ and

$$\limsup_{k\to\infty} \frac{|s_{k+1} - \bar{s}|}{|s_k - \bar{s}|^2} = \delta < \infty.$$

Examples:

Linear convergence $s_k = \left(\frac{1}{10}\right)^k$: 0.1, 0.01, 0.001, etc. $\bar{s} = 0$.

$$\frac{|s_{k+1} - \bar{s}|}{|s_k - \bar{s}|} = 0.1.$$

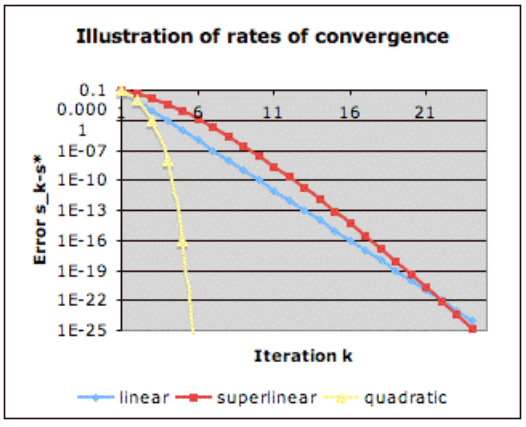
Superlinear convergence $s_k = 0.1 \cdot \frac{1}{k!}$: $\frac{1}{10}$, $\frac{1}{20}$, $\frac{1}{60}$, $\frac{1}{240}$, $\frac{1}{1250}$, etc. $\bar{s} = 0$.

$$\frac{|s_{k+1} - \bar{s}|}{|s_k - \bar{s}|} = \frac{k!}{(k+1)!} = \frac{1}{k+1} \to 0 \text{ as } k \to \infty.$$

Quadratic convergence $s_k = \left(\frac{1}{10}\right)^{(2^{k-1})}$: 0.1, 0.01, 0.0001, 0.00000001, etc. $\bar{s} = 0$.

$$\frac{|s_{k+1} - \bar{s}|}{|s_k - \bar{s}|^2} = \frac{(10^{2^{k-1}})^2}{10^{2^k}} = 1.$$

This illustration compares the rates of convergence of the above sequences:



Since an algorithm for nonlinear optimization problems, in its abstract form, generates an *infinite* sequence of points $\{x_k\}$ converging to a solution $\bar{x}$ only in the limit, it makes sense to discuss the rate of convergence of the sequence $\|e^k\| = \|x_k - \bar{x}\|$, or $E^k = |f(x_k) - f(\bar{x})|$, which both have limit 0. For example, in the previous section we've shown that, on a convex quadratic function, the steepest descent algorithm exhibits linear convergence, with rate bounded by the condition number of the Hessian. For non-quadratic functions, the steepest descent algorithm behaves similarly in the limit, i.e., once the iterates reach a small neighborhood of the limit point.

8.1.2 Rate of convergence of the pure Newton's method

We have seen from our examples that, even for convex functions, the Newton's method in its pure form (i.e., with stepsize of 1 at every iteration) does not guarantee descent at each iteration, and may produce a diverging sequence of iterates. Moreover, each iteration of the Newton's method is much more computationally intensive then that of the steepest descent. However, under certain conditions, the method exhibits *quadratic* rate of convergence, making it the "ideal" method for solving convex optimization problems. Recall that a method exhibits quadratic convergence when $\|e_k\| = \|x_k - \bar{x}\| \to 0$ and

$$\lim_{k\to\infty} \frac{\|e^{k+1}\|}{\|e^k\|^2} = C.$$

Roughly speaking, if the iterates converge quadratically, the accuracy (i.e., the number of correct digits) of the solution doubles in a fixed number of iterations.

There are many ways to state and prove results regarding the convergence on the Newton's method. We provide one that provides a particular insight into the circumstances under which pure Newton's method demonstrates quadratic convergence (compare the theorem below to BSS 8.6.5).

Let $\|v\|$ denote the usual Euclidian norm of a vector, namely $\|v\| := \sqrt{v^T v}$. Recall that the operator norm of a matrix M is defined as follows:

$$\|M\| := \max_x \{\|Mx\| : \|x\| = 1\}.$$

As a consequence of this definition, for any x, $\|Mx\| \le \|M\| \cdot \|x\|$.

Theorem 18 (Quadratic convergence) *Suppose $f(x)$ is twice continuously differentiable and $x^\star$ is a point for which $\nabla f(x^\star) = 0$. Suppose $H(x)$ satisfies the following conditions:*

- *there exists a scalar $h > 0$ for which $\|[H(x^\star)]^{-1}\| \le \frac{1}{h}$*
- *there exists scalars $\beta > 0$ and $L > 0$ for which $\|H(x) - H(y)\| \le L\|x - y\|$ for all x and y satisfying $\|x - x^\star\| \le \beta$ and $\|y - x^\star\| \le \beta$.*

Let x satisfy $\|x - x^\star\| \le \delta\gamma$, where $0 < \delta < 1$ and $\gamma := \min\left\{\beta, \frac{2h}{3L}\right\}$, and let $x_N := x - H(x)^{-1}\nabla f(x)$. Then:

(i) $\|x_N - x^\star\| \le \|x - x^\star\|^2 \left(\frac{L}{2(h - L\|x - x^\star\|)}\right)$

(ii) $\|x_N - x^\star\| < \delta\|x - x^\star\|$, *and hence the iterates converge to $x^\star$*

(iii) $\|x_N - x^\star\| \le \|x - x^\star\|^2 \left(\frac{3L}{2h}\right)$.

The proof relies on the following two "elementary" facts.

Proposition 19 *Suppose that M is a symmetric matrix. Then the following are equivalent:*

1. *$h > 0$ satisfies $\|M^{-1}\| \le \frac{1}{h}$*
2. *$h > 0$ satisfies $\|Mv\| \ge h \cdot \|v\|$ for any vector v*

Proof: Left as an exercise. ∎

Proposition 20 *Suppose that $f(x)$ is twice differentiable. Then*

$$\nabla f(z) - \nabla f(x) = \int_0^1 [H(x + t(z - x))]\,(z - x)dt\ .$$

Proof: Let $\phi(t) := \nabla f(x + t(z - x))$. Then $\phi(0) = \nabla f(x)$ and $\phi(1) = \nabla f(z)$, and $\phi^{'}(t) = [H(x + t(z - x))]\,(z - x)$. From the fundamental theorem of calculus, we have:

$$\begin{aligned}\nabla f(z) - \nabla f(x) &= \phi(1) - \phi(0) \\ &= \int_0^1 \phi^{'}(t)dt \\ &= \int_0^1 [H(x + t(z - x))]\,(z - x)dt \ . \blacksquare\end{aligned}$$

Proof of Theorem 18

We have:

$$\begin{aligned}x_N - x^\star &= x - H(x)^{-1}\nabla f(x) - x^\star \\ &= x - x^\star + H(x)^{-1}\left(\nabla f(x^\star) - \nabla f(x)\right) \\ &= x - x^\star + H(x)^{-1}\int_0^1 [H(x + t(x^\star - x))]\,(x^\star - x)dt \text{ (from Proposition 20)} \\ &= H(x)^{-1}\int_0^1 [H(x + t(x^\star - x)) - H(x)]\,(x^\star - x)dt.\end{aligned}$$

Therefore

$$\begin{aligned}\|x_N - x^\star\| &\leq \|H(x)^{-1}\|\int_0^1 \|\,[H(x + t(x^\star - x)) - H(x)]\,\| \cdot \|(x^\star - x)\|dt \\ &\leq \|x^\star - x\| \cdot \|H(x)^{-1}\|\int_0^1 L \cdot t \cdot \|(x^\star - x)\|dt \\ &= \|x^\star - x\|^2\|H(x)^{-1}\|L\int_0^1 tdt \\ &= \frac{\|x^\star - x\|^2\|H(x)^{-1}\|L}{2}.\end{aligned}$$

We now bound $\|H(x)^{-1}\|$. Let v be any vector. Then

$$\begin{aligned}\|H(x)v\| &= \|H(x^\star)v + (H(x) - H(x^\star))v\| \\ &\geq \|H(x^\star)v\| - \|(H(x) - H(x^\star))v\| \\ &\geq h \cdot \|v\| - \|H(x) - H(x^\star)\|\|v\| \text{ (from Proposition 19)} \\ &\geq h \cdot \|v\| - L\|x^\star - x\| \cdot \|v\| \\ &= (h - L\|x^\star - x\|) \cdot \|v\| \ .\end{aligned}$$

Invoking Proposition 19 again, we see that this implies that

$$\|H(x)^{-1}\| \leq \frac{1}{h - L\|x^\star - x\|} \ .$$

Combining this with the above yields

$$\|x_N - x^\star\| \leq \|x^\star - x\|^2 \cdot \frac{L}{2\,(h - L\|x^\star - x\|)} \ ,$$

which is *(i)* of the theorem. Because $L\|x^\star - x\| \leq \delta \cdot \frac{2h}{3} < \frac{2h}{3}$ we have:

$$\|x_N - x^\star\| \leq \|x^\star - x\| \cdot \frac{L\|x^\star - x\|}{2\left(h - L\|x^\star - x\|\right)} \leq \frac{\delta \cdot \frac{2h}{3}}{2\left(h - \frac{2h}{3}\right)}\|x^\star - x\| = \delta\|x^\star - x\| \ ,$$

which establishes *(ii)* of the theorem. Finally, we have

$$\|x_N - x^\star\| \leq \|x^\star - x\|^2 \frac{L}{2\left(h - L\|x^\star - x\|\right)} \leq \|x^\star - x\|^2 \cdot \frac{L}{2\left(h - \frac{2h}{3}\right)} = \frac{3L}{2h}\|x^\star - x\|^2 \ ,$$

which establishes *(iii)* of the theorem. ∎

Notice that the results regarding the convergence and rate of convergence in the above theorem are *local*, i.e., they apply only if the algorithm is initialized at certain starting points (the ones "sufficiently close" to the desired limit). In practice, it is not known how to pick such starting points, or to check if the proposed starting point is adequate. (With the very important exception of self-concordant functions.)

8.2 Further discussion and modifications of the Newton's method

8.2.1 Global convergence for strongly convex functions with a two-phase Newton's method

We have noted that, to ensure descent at each iteration, the Newton's method can be augmented by a line search. This idea can be formalized, and the efficiency of the resulting algorithm can be analyzed (see, for example, "Convex Optimization" by Stephen Boyd and Lieven Vandenberghe, available at `http://www.stanford.edu/~boyd/cvxbook.html` for a fairly simple presentation of the analysis).

Suppose that $f(x)$ is *strongly convex* on it domain, i.e., assume there exists $\mu > 0$ such that $H(x) - \mu I$ is positive semidefinite for all x (I is the identity matrix), and that the Hessian is Lipschitz continuous everywhere on the domain of f. Suppose we apply the Newton's method with the stepsize at each iteration determined by the backtracking procedure of section 5.2.2. That is, at each iteration of the algorithm we first attempt to take a full Newton step, but reduce the stepsize if the decrease in the function value is not sufficient. Then there exist positive numbers η and γ such that

- if $\|\nabla f(x_k)\| \geq \eta$, then $f(x_{k+1}) - f(x_k) \leq -\gamma$, and
- if $\|\nabla f(x_k)\| < \eta$, then stepsize $\lambda_k = 1$ will be selected, and the next iterate will satisfy $\|\nabla f(x_{k+1})\| < \eta$, and so will all the further iterates. Moreover, quadratic convergence will be observed in this phase.

As hinted above, the algorithm will proceed in two phases: while the iterates are far from the minimizer, a "dampening" of the Newton step will be required, but there will be a guaranteed decrease in the objective function values. This phase (referred to as "dampened Newton phase") cannot take more than $\frac{f(x_0) - f(x^\star)}{\gamma}$ iterations. Once the norm of the gradient becomes sufficiently small, no dampening of the Newton step will required in the rest of the algorithm, and quadratic convergence will be observed, thus making it the "quadratically convergence phase."

Note that it is not necessary to know the values of η and γ to apply this version of the algorithm! The two-phase Newton's method is globally convergent; however, to ensure global convergence, the function being minimized needs to posses particularly nice *global* properties.

8.2.2 Other modifications of the Newton's method

We have seen that if Newton's method is initialized sufficiently close to the point $\bar{x}$ such that $\nabla f(\bar{x}) = 0$ and $H(\bar{x})$ is positive definite (i.e., $\bar{x}$ is a local minimizer), then it will converge quadratically, using stepsizes of $\lambda = 1$. There are three issues in the above statement that we should be concerned with:

- What if $H(\bar{x})$ is singular, or nearly-singular?
- How do we know if we are "close enough," and what to do if we are not?
- Can we modify Newton's method to guarantee global convergence?

In the previous subsection we "assumed away" the first issue, and, under an additional assumption, showed how to address the other two. What if the function f is not strongly convex, and $H(x)$ may approach singularity?

There are two popular approaches (which are actually closely related) to address these issues. The first approach ensures that the method always uses a descent direction. For example, instead of the direction $-H(x_k)^{-1}\nabla f(x_k)$, use the direction $-(H(x_k)+\epsilon_k I)^{-1}\nabla f(x_k)$, where $\epsilon_k \geq 0$ is chosen so that the smallest eigenvalue of $H(x_k) + \epsilon_k I$ is bounded below by a fixed number $\delta > 0$. It is important to choose the value of δ appropriately — if it is chosen to be too small, the matrix employed in computing the direction can become ill-conditioned if $H(\bar{x})$ is nearly singular; if it is chosen to be too large, the direction becomes nearly that of the steepest descent algorithm, and hence only linear convergence can be guaranteed. Hence, the value of ϵ_k is often chosen dynamically.

The second approach is the so-called *trust region* method. Note that the main idea behind the Newton's method is to represent the function $f(x)$ by its quadratic approximation $q_k(x) = f(x_k) + \nabla f(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T H(x_k)(x - x_k)$ around the current iterate, and then minimize that approximation. While locally the approximation is a good one, this may no longer be the case when a large step is taken. The trust region methods hence find the next iterate by solving the following *constrained* optimization problem:

$$\min q_k(x) \text{ s.t. } \|x - x_k\| \leq \Delta_k$$

(as it turns out, this problem is not much harder to solve than the unconstrained minimization of $q_k(s)$).

The value of Δ_k is set to represent the size of the region in which we can "trust" $q_k(x)$ to provide a good approximation of $f(x)$. Smaller values of Δ_k ensure that we are working with an accurate representation of $f(x)$, but result in conservative steps. Larger values of Δ_k allow for larger steps, but may lead to inaccurate estimation of the objective function. To account for this, the value if Δ_k is updated dynamically throughout the algorithm, namely, it is increased if it is observed that $q_k(x)$ provided an exceptionally good approximation of $f(x)$ at the previous iteration, and decreased is the approximation was exceptionally bad.