Chapter 9

Unconstrained minimization

9.1 Unconstrained minimization problems

In this chapter we discuss methods for solving the unconstrained optimization problem

$$\text{minimize} \quad f(x) \tag{9.1}$$

where $f : \mathbf{R}^n \to \mathbf{R}$ is convex and twice continuously differentiable (which implies that $\mathbf{dom}\, f$ is open). We will assume that the problem is solvable, *i.e.*, there exists an optimal point $x^\star$. (More precisely, the assumptions later in the chapter will imply that $x^\star$ exists and is unique.) We denote the optimal value, $\inf_x f(x) = f(x^\star)$, as $p^\star$.

Since f is differentiable and convex, a necessary and sufficient condition for a point $x^\star$ to be optimal is

$$\nabla f(x^\star) = 0 \tag{9.2}$$

(see §4.2.3). Thus, solving the unconstrained minimization problem (9.1) is the same as finding a solution of (9.2), which is a set of n equations in the n variables $x_1, \ldots, x_n$. In a few special cases, we can find a solution to the problem (9.1) by analytically solving the optimality equation (9.2), but usually the problem must be solved by an iterative algorithm. By this we mean an algorithm that computes a sequence of points $x^{(0)},\ x^{(1)}, \ldots \in \mathbf{dom}\, f$ with $f(x^{(k)}) \to p^\star$ as $k \to \infty$. Such a sequence of points is called a *minimizing sequence* for the problem (9.1). The algorithm is terminated when $f(x^{(k)}) - p^\star \le \epsilon$, where $\epsilon > 0$ is some specified tolerance.

Initial point and sublevel set

The methods described in this chapter require a suitable starting point $x^{(0)}$. The starting point must lie in $\mathbf{dom}\, f$, and in addition the sublevel set

$$S = \{x \in \mathbf{dom}\, f \mid f(x) \le f(x^{(0)})\} \tag{9.3}$$

must be closed. This condition is satisfied for all $x^{(0)} \in \mathbf{dom}\, f$ if the function f is *closed*, *i.e.*, all its sublevel sets are closed (see §A.3.3). Continuous functions with

$\mathbf{dom}\, f = \mathbf{R}^n$ are closed, so if $\mathbf{dom}\, f = \mathbf{R}^n$, the initial sublevel set condition is satisfied by any $x^{(0)}$. Another important class of closed functions are continuous functions with open domains, for which $f(x)$ tends to infinity as x approaches $\mathbf{bd}\,\mathbf{dom}\, f$.

9.1.1 Examples

Quadratic minimization and least-squares

The general convex quadratic minimization problem has the form

$$\text{minimize} \quad (1/2)x^T Px + q^T x + r, \tag{9.4}$$

where $P \in \mathbf{S}^n_+$, $q \in \mathbf{R}^n$, and $r \in \mathbf{R}$. This problem can be solved via the optimality conditions, $Px^\star + q = 0$, which is a set of linear equations. When $P \succ 0$, there is a unique solution, $x^\star = -P^{-1}q$. In the more general case when P is not positive definite, any solution of $Px^\star = -q$ is optimal for (9.4); if $Px^\star = -q$ does not have a solution, then the problem (9.4) is unbounded below (see exercise 9.1). Our ability to analytically solve the quadratic minimization problem (9.4) is the basis for Newton's method, a powerful method for unconstrained minimization described in §9.5.

One special case of the quadratic minimization problem that arises very frequently is the least-squares problem

$$\text{minimize} \quad \|Ax - b\|_2^2 = x^T(A^T A)x - 2(A^T b)^T x + b^T b.$$

The optimality conditions

$$A^T A x^\star = A^T b$$

are called the *normal equations* of the least-squares problem.

Unconstrained geometric programming

As a second example, we consider an unconstrained geometric program in convex form,

$$\text{minimize} \quad f(x) = \log\left(\textstyle\sum_{i=1}^m \exp(a_i^T x + b_i)\right).$$

The optimality condition is

$$\nabla f(x^\star) = \frac{1}{\sum_{j=1}^m \exp(a_j^T x^\star + b_j)} \sum_{i=1}^m \exp(a_i^T x^\star + b_i) a_i = 0,$$

which in general has no analytical solution, so here we must resort to an iterative algorithm. For this problem, $\mathbf{dom}\, f = \mathbf{R}^n$, so any point can be chosen as the initial point $x^{(0)}$.

Analytic center of linear inequalities

We consider the optimization problem

$$\text{minimize} \quad f(x) = -\textstyle\sum_{i=1}^m \log(b_i - a_i^T x), \tag{9.5}$$

where the domain of f is the open set

$$\mathbf{dom}\, f = \{x \mid a_i^T x < b_i,\ i = 1, \ldots, m\}.$$

The objective function f in this problem is called the *logarithmic barrier* for the inequalities $a_i^T x \leq b_i$. The solution of (9.5), if it exists, is called the *analytic center* of the inequalities. The initial point $x^{(0)}$ must satisfy the strict inequalities $a_i^T x^{(0)} < b_i$, $i = 1, \ldots, m$. Since f is closed, the sublevel set S for any such point is closed.

Analytic center of a linear matrix inequality

A closely related problem is

$$\text{minimize} \quad f(x) = \log\det F(x)^{-1} \tag{9.6}$$

where $F : \mathbf{R}^n \to \mathbf{S}^p$ is affine, *i.e.*,

$$F(x) = F_0 + x_1 F_1 + \cdots + x_n F_n,$$

with $F_i \in \mathbf{S}^p$. Here the domain of f is

$$\mathbf{dom}\, f = \{x \mid F(x) \succ 0\}.$$

The objective function f is called the *logarithmic barrier* for the linear matrix inequality $F(x) \succeq 0$, and the solution (if it exists) is called the analytic center of the linear matrix inequality. The initial point $x^{(0)}$ must satisfy the strict linear matrix inequality $F(x^{(0)}) \succ 0$. As in the previous example, the sublevel set of any such point will be closed, since f is closed.

9.1.2 Strong convexity and implications

In much of this chapter (with the exception of §9.6) we assume that the objective function is *strongly convex* on S, which means that there exists an $m > 0$ such that

$$\nabla^2 f(x) \succeq mI \tag{9.7}$$

for all $x \in S$. Strong convexity has several interesting consequences. For $x, y \in S$ we have

$$f(y) = f(x) + \nabla f(x)^T (y - x) + \frac{1}{2}(y - x)^T \nabla^2 f(z)(y - x)$$

for some z on the line segment $[x, y]$. By the strong convexity assumption (9.7), the last term on the righthand side is at least $(m/2)\|y - x\|_2^2$, so we have the inequality

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{m}{2}\|y - x\|_2^2 \tag{9.8}$$

for all x and y in S. When $m = 0$, we recover the basic inequality characterizing convexity; for $m > 0$ we obtain a better lower bound on $f(y)$ than follows from convexity alone.

We will first show that the inequality (9.8) can be used to bound $f(x) - p^\star$, which is the suboptimality of the point x, in terms of $\|\nabla f(x)\|_2$. The righthand side of (9.8) is a convex quadratic function of y (for fixed x). Setting the gradient with respect to y equal to zero, we find that $\tilde{y} = x - (1/m)\nabla f(x)$ minimizes the righthand side. Therefore we have

$$\begin{array}{rcl} f(y) & \geq & f(x) + \nabla f(x)^T(y-x) + \dfrac{m}{2}\|y-x\|_2^2 \\ & \geq & f(x) + \nabla f(x)^T(\tilde{y}-x) + \dfrac{m}{2}\|\tilde{y}-x\|_2^2 \\ & = & f(x) - \dfrac{1}{2m}\|\nabla f(x)\|_2^2. \end{array}$$

Since this holds for any $y \in S$, we have

$$p^\star \geq f(x) - \frac{1}{2m}\|\nabla f(x)\|_2^2. \tag{9.9}$$

This inequality shows that if the gradient is small at a point, then the point is nearly optimal. The inequality (9.9) can also be interpreted as a condition for *suboptimality* which generalizes the optimality condition (9.2):

$$\|\nabla f(x)\|_2 \leq (2m\epsilon)^{1/2} \Longrightarrow f(x) - p^\star \leq \epsilon. \tag{9.10}$$

We can also derive a bound on $\|x - x^\star\|_2$, the distance between x and any optimal point $x^\star$, in terms of $\|\nabla f(x)\|_2$:

$$\|x - x^\star\|_2 \leq \frac{2}{m}\|\nabla f(x)\|_2. \tag{9.11}$$

To see this, we apply (9.8) with $y = x^\star$ to obtain

$$\begin{array}{rcl} p^\star = f(x^\star) & \geq & f(x) + \nabla f(x)^T(x^\star - x) + \dfrac{m}{2}\|x^\star - x\|_2^2 \\ & \geq & f(x) - \|\nabla f(x)\|_2\|x^\star - x\|_2 + \dfrac{m}{2}\|x^\star - x\|_2^2, \end{array}$$

where we use the Cauchy-Schwarz inequality in the second inequality. Since $p^\star \leq f(x)$, we must have

$$-\|\nabla f(x)\|_2 \, \|x^\star - x\|_2 + \frac{m}{2}\|x^\star - x\|_2^2 \leq 0,$$

from which (9.11) follows. One consequence of (9.11) is that the optimal point $x^\star$ is unique.

Upper bound on $\nabla^2 f(x)$

The inequality (9.8) implies that the sublevel sets contained in S are bounded, so in particular, S is bounded. Therefore the maximum eigenvalue of $\nabla^2 f(x)$, which is a continuous function of x on S, is bounded above on S, *i.e.*, there exists a constant M such that

$$\nabla^2 f(x) \preceq MI \tag{9.12}$$

for all $x \in S$. This upper bound on the Hessian implies for any $x,\ y \in S$,

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{M}{2}\|y-x\|_2^2, \tag{9.13}$$

which is analogous to (9.8). Minimizing each side over y yields

$$p^\star \leq f(x) - \frac{1}{2M}\|\nabla f(x)\|_2^2, \tag{9.14}$$

the counterpart of (9.9).

Condition number of sublevel sets

From the strong convexity inequality (9.7) and the inequality (9.12), we have

$$mI \preceq \nabla^2 f(x) \preceq MI \tag{9.15}$$

for all $x \in S$. The ratio $\kappa = M/m$ is thus an upper bound on the condition number of the matrix $\nabla^2 f(x)$, *i.e.*, the ratio of its largest eigenvalue to its smallest eigenvalue. We can also give a geometric interpretation of (9.15) in terms of the sublevel sets of f.

We define the *width* of a convex set $C \subseteq \mathbf{R}^n$, in the direction q, where $\|q\|_2 = 1$, as

$$W(C,q) = \sup_{z\in C} q^T z - \inf_{z \in C} q^T z.$$

The *minimum width* and *maximum width* of C are given by

$$W_{\min} = \inf_{\|q\|_2=1} W(C,q), \qquad W_{\max} = \sup_{\|q\|_2=1} W(C,q).$$

The *condition number* of the convex set C is defined as

$$\mathbf{cond}(C) = \frac{W_{\max}^2}{W_{\min}^2},$$

i.e., the square of the ratio of its maximum width to its minimum width. The condition number of C gives a measure of its *anisotropy* or *eccentricity*. If the condition number of a set C is small (say, near one) it means that the set has approximately the same width in all directions, *i.e.*, it is nearly spherical. If the condition number is large, it means that the set is far wider in some directions than in others.

Example 9.1 *Condition number of an ellipsoid.* Let $\mathcal{E}$ be the ellipsoid

$$\mathcal{E} = \{x \mid (x-x_0)^T A^{-1}(x-x_0) \leq 1\},$$

where $A \in \mathbf{S}^n_{++}$. The width of $\mathcal{E}$ in the direction q is

$$\begin{aligned} \sup_{z\in\mathcal{E}} q^T z - \inf_{z\in\mathcal{E}} q^T z &= (\|A^{1/2}q\|_2 + q^T x_0) - (-\|A^{1/2}q\|_2 + q^T x_0) \\ &= 2\|A^{1/2}q\|_2. \end{aligned}$$

It follows that its minimum and maximum width are

$$W_{\min} = 2\lambda_{\min}(A)^{1/2}, \qquad W_{\max} = 2\lambda_{\max}(A)^{1/2},$$

and its condition number is

$$\mathbf{cond}(\mathcal{E}) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)} = \kappa(A),$$

where $\kappa(A)$ denotes the condition number of the matrix A, *i.e.*, the ratio of its maximum singular value to its minimum singular value. Thus the condition number of the ellipsoid $\mathcal{E}$ is the same as the condition number of the matrix A that defines it.

Now suppose f satisfies $mI \preceq \nabla^2 f(x) \preceq MI$ for all $x \in S$. We will derive a bound on the condition number of the α-sublevel $C_\alpha = \{x \mid f(x) \leq \alpha\}$, where $p^\star < \alpha \leq f(x^{(0)})$. Applying (9.13) and (9.8) with $x = x^\star$, we have

$$p^\star + (M/2)\|y - x^\star\|_2^2 \geq f(y) \geq p^\star + (m/2)\|y - x^\star\|_2^2.$$

This implies that $B_{\text{inner}} \subseteq C_\alpha \subseteq B_{\text{outer}}$ where

$$\begin{array}{rcl} B_{\text{inner}} & = & \{y \mid \|y - x^\star\|_2 \leq (2(\alpha - p^\star)/M)^{1/2}\}, \\ B_{\text{outer}} & = & \{y \mid \|y - x^\star\|_2 \leq (2(\alpha - p^\star)/m)^{1/2}\}. \end{array}$$

In other words, the α-sublevel set contains B_{inner}, and is contained in B_{outer}, which are balls with radii

$$(2(\alpha - p^\star)/M)^{1/2}, \qquad (2(\alpha - p^\star)/m)^{1/2},$$

respectively. The ratio of the radii squared gives an upper bound on the condition number of C_α:

$$\mathbf{cond}(C_\alpha) \leq \frac{M}{m}.$$

We can also give a geometric interpretation of the condition number $\kappa(\nabla^2 f(x^\star))$ of the Hessian at the optimum. From the Taylor series expansion of f around $x^\star$,

$$f(y) \approx p^\star + \frac{1}{2}(y - x^\star)^T \nabla^2 f(x^\star)(y - x^\star),$$

we see that, for α close to $p^\star$,

$$C_\alpha \approx \{y \mid (y - x^\star)^T \nabla^2 f(x^\star)(y - x^\star) \leq 2(\alpha - p^\star)\},$$

i.e., the sublevel set is well approximated by an ellipsoid with center $x^\star$. Therefore

$$\lim_{\alpha \to p^\star} \mathbf{cond}(C_\alpha) = \kappa(\nabla^2 f(x^\star)).$$

We will see that the condition number of the sublevel sets of f (which is bounded by M/m) has a strong effect on the efficiency of some common methods for unconstrained minimization.

The strong convexity constants

It must be kept in mind that the constants m and M are known only in rare cases, so the inequality (9.10) cannot be used as a practical stopping criterion. It can be considered a *conceptual* stopping criterion; it shows that if the gradient of f at x is small enough, then the difference between $f(x)$ and $p^\star$ is small. If we terminate an algorithm when $\|\nabla f(x^{(k)})\|_2 \leq \eta$, where η is chosen small enough to be (very likely) smaller than $(m\epsilon)^{1/2}$, then we have $f(x^{(k)}) - p^\star \leq \epsilon$ (very likely).

In the following sections we give convergence proofs for algorithms, which include bounds on the number of iterations required before $f(x^{(k)}) - p^\star \leq \epsilon$, where ϵ is some positive tolerance. Many of these bounds involve the (usually unknown) constants m and M, so the same comments apply. These results are at least conceptually useful; they establish that the algorithm converges, even if the bound on the number of iterations required to reach a given accuracy depends on constants that are unknown.

We will encounter one important exception to this situation. In §9.6 we will study a special class of convex functions, called *self-concordant*, for which we can provide a complete convergence analysis (for Newton's method) that does not depend on any unknown constants.

9.2 Descent methods

The algorithms described in this chapter produce a minimizing sequence $x^{(k)}$, $k = 1, \ldots,$ where

$$x^{(k+1)} = x^{(k)} + t^{(k)} \Delta x^{(k)}$$

and $t^{(k)} > 0$ (except when $x^{(k)}$ is optimal). Here the concatenated symbols Δ and x that form Δx are to be read as a single entity, a vector in $\mathbf{R}^n$ called the *step* or *search direction* (even though it need not have unit norm), and $k = 0, 1, \ldots$ denotes the iteration number. The scalar $t^{(k)} \geq 0$ is called the *step size* or *step length* at iteration k (even though it is not equal to $\|x^{(k+1)} - x^{(k)}\|$ unless $\|\Delta x^{(k)}\| = 1$). The terms 'search step' and 'scale factor' are more accurate, but 'search direction' and 'step length' are the ones widely used. When we focus on one iteration of an algorithm, we sometimes drop the superscripts and use the lighter notation $x^+ = x + t\Delta x$, or $x := x + t\Delta x$, in place of $x^{(k+1)} = x^{(k)} + t^{(k)}\Delta x^{(k)}$.

All the methods we study are *descent methods*, which means that

$$f(x^{(k+1)}) < f(x^{(k)}),$$

except when $x^{(k)}$ is optimal. This implies that for all k we have $x^{(k)} \in S$, the initial sublevel set, and in particular we have $x^{(k)} \in \mathbf{dom}\, f$. From convexity we know that $\nabla f(x^{(k)})^T (y - x^{(k)}) \geq 0$ implies $f(y) \geq f(x^{(k)})$, so the search direction in a descent method must satisfy

$$\nabla f(x^{(k)})^T \Delta x^{(k)} < 0,$$

i.e., it must make an acute angle with the negative gradient. We call such a direction a *descent direction* (for f, at $x^{(k)}$).

The outline of a general descent method is as follows. It alternates between two steps: determining a descent direction Δx, and the selection of a step size t.

Algorithm 9.1 *General descent method.*

```
given a starting point $x \in \mathbf{dom}\, f$.
repeat
    1. Determine a descent direction $\Delta x$.
    2. Line search. Choose a step size $t > 0$.
    3. Update. $x := x + t\Delta x$.
until stopping criterion is satisfied.
```

The second step is called the *line search* since selection of the step size t determines where along the line $\{x + t\Delta x \mid t \in \mathbf{R}_+\}$ the next iterate will be. (A more accurate term might be *ray search*.)

A practical descent method has the same general structure, but might be organized differently. For example, the stopping criterion is often checked while, or immediately after, the descent direction Δx is computed. The stopping criterion is often of the form $\|\nabla f(x)\|_2 \leq \eta$, where η is small and positive, as suggested by the suboptimality condition (9.9).

Exact line search

One line search method sometimes used in practice is *exact line search*, in which t is chosen to minimize f along the ray $\{x + t\Delta x \mid t \geq 0\}$:

$$t = \mathrm{argmin}_{s \geq 0}\, f(x + s\Delta x). \tag{9.16}$$

An exact line search is used when the cost of the minimization problem with one variable, required in (9.16), is low compared to the cost of computing the search direction itself. In some special cases the minimizer along the ray can be found analytically, and in others it can be computed efficiently. (This is discussed in §9.7.1.)

Backtracking line search

Most line searches used in practice are *inexact*: the step length is chosen to approximately minimize f along the ray $\{x + t\Delta x \mid t \geq 0\}$, or even to just reduce f 'enough'. Many inexact line search methods have been proposed. One inexact line search method that is very simple and quite effective is called *backtracking* line search. It depends on two constants α, β with $0 < \alpha < 0.5$, $0 < \beta < 1$.

Algorithm 9.2 *Backtracking line search.*

```
given a descent direction $\Delta x$ for $f$ at $x \in \mathbf{dom}\, f$, $\alpha \in (0, 0.5)$, $\beta \in (0, 1)$.
$t := 1$.
while $f(x + t\Delta x) > f(x) + \alpha t \nabla f(x)^T \Delta x$,    $t := \beta t$.
```

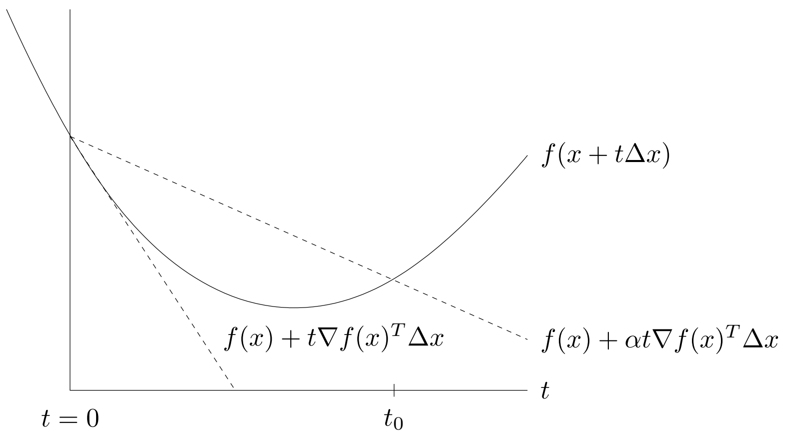
$f(x + t\Delta x)$
$f(x) + t\nabla f(x)^T\Delta x$
$f(x) + \alpha t\nabla f(x)^T\Delta x$
t
$t = 0$
t_0

Figure 9.1 *Backtracking line search.* The curve shows f, restricted to the line over which we search. The lower dashed line shows the linear extrapolation of f, and the upper dashed line has a slope a factor of α smaller. The backtracking condition is that f lies below the upper dashed line, *i.e.*, $0 \leq t \leq t_0$.

The line search is called backtracking because it starts with unit step size and then reduces it by the factor β until the stopping condition $f(x + t\Delta x) \leq f(x) + \alpha t\nabla f(x)^T\Delta x$ holds. Since Δx is a descent direction, we have $\nabla f(x)^T\Delta x < 0$, so for small enough t we have

$$f(x + t\Delta x) \approx f(x) + t\nabla f(x)^T\Delta x < f(x) + \alpha t\nabla f(x)^T\Delta x,$$

which shows that the backtracking line search eventually terminates. The constant α can be interpreted as the fraction of the decrease in f predicted by linear extrapolation that we will accept. (The reason for requiring α to be smaller than 0.5 will become clear later.)

The backtracking condition is illustrated in figure 9.1. This figure suggests, and it can be shown, that the backtracking exit inequality $f(x + t\Delta x) \leq f(x) + \alpha t\nabla f(x)^T\Delta x$ holds for $t \geq 0$ in an interval $(0, t_0]$. It follows that the backtracking line search stops with a step length t that satisfies

$$t = 1, \qquad \text{or} \qquad t \in (\beta t_0, t_0].$$

The first case occurs when the step length $t = 1$ satisfies the backtracking condition, *i.e.*, $1 \leq t_0$. In particular, we can say that the step length obtained by backtracking line search satisfies

$$t \geq \min\{1, \beta t_0\}.$$

When $\mathbf{dom}\, f$ is not all of $\mathbf{R}^n$, the condition $f(x+t\Delta x) \leq f(x)+\alpha t\nabla f(x)^T\Delta x$ in the backtracking line search must be interpreted carefully. By our convention that f is infinite outside its domain, the inequality implies that $x + t\Delta x \in \mathbf{dom}\, f$. In a practical implementation, we first multiply t by β until $x + t\Delta x \in \mathbf{dom}\, f$;

then we start to check whether the inequality $f(x + t\Delta x) \leq f(x) + \alpha t \nabla f(x)^T \Delta x$ holds.

The parameter α is typically chosen between 0.01 and 0.3, meaning that we accept a decrease in f between 1% and 30% of the prediction based on the linear extrapolation. The parameter β is often chosen to be between 0.1 (which corresponds to a very crude search) and 0.8 (which corresponds to a less crude search).

9.3 Gradient descent method

A natural choice for the search direction is the negative gradient $\Delta x = -\nabla f(x)$. The resulting algorithm is called the *gradient algorithm* or *gradient descent method*.

Algorithm 9.3 *Gradient descent method.*

```
given a starting point $x \in \mathbf{dom}\, f$.
repeat
    1. $\Delta x := -\nabla f(x)$.
    2. Line search. Choose step size $t$ via exact or backtracking line search.
    3. Update. $x := x + t\Delta x$.
until stopping criterion is satisfied.
```

The stopping criterion is usually of the form $\|\nabla f(x)\|_2 \leq \eta$, where η is small and positive. In most implementations, this condition is checked after step 1, rather than after the update.

9.3.1 Convergence analysis

In this section we present a simple convergence analysis for the gradient method, using the lighter notation $x^+ = x + t\Delta x$ for $x^{(k+1)} = x^{(k)} + t^{(k)}\Delta x^{(k)}$, where $\Delta x = -\nabla f(x)$. We assume f is strongly convex on S, so there are positive constants m and M such that $mI \preceq \nabla^2 f(x) \preceq MI$ for all $x \in S$. Define the function $\tilde{f} : \mathbf{R} \to \mathbf{R}$ by $\tilde{f}(t) = f(x - t\nabla f(x))$, *i.e.*, f as a function of the step length t in the negative gradient direction. In the following discussion we will only consider t for which $x - t\nabla f(x) \in S$. From the inequality (9.13), with $y = x - t\nabla f(x)$, we obtain a quadratic upper bound on $\tilde{f}$:

$$\tilde{f}(t) \leq f(x) - t\|\nabla f(x)\|_2^2 + \frac{Mt^2}{2}\|\nabla f(x)\|_2^2. \tag{9.17}$$

Analysis for exact line search

We now assume that an exact line search is used, and minimize over t both sides of the inequality (9.17). On the lefthand side we get $\tilde{f}(t_{\text{exact}})$, where t_{exact} is the step length that minimizes $\tilde{f}$. The righthand side is a simple quadratic, which is minimized by $t = 1/M$, and has minimum value $f(x) - (1/(2M))\|\nabla f(x)\|_2^2$. Therefore we have

$$f(x^+) = \tilde{f}(t_{\text{exact}}) \leq f(x) - \frac{1}{2M}\|\nabla(f(x))\|_2^2.$$

Subtracting $p^\star$ from both sides, we get

$$f(x^+) - p^\star \leq f(x) - p^\star - \frac{1}{2M}\|\nabla f(x)\|_2^2.$$

We combine this with $\|\nabla f(x)\|_2^2 \geq 2m(f(x) - p^\star)$ (which follows from (9.9)) to conclude

$$f(x^+) - p^\star \leq (1 - m/M)(f(x) - p^\star).$$

Applying this inequality recursively, we find that

$$f(x^{(k)}) - p^\star \leq c^k(f(x^{(0)}) - p^\star) \tag{9.18}$$

where $c = 1 - m/M < 1$, which shows that $f(x^{(k)})$ converges to $p^\star$ as $k \to \infty$. In particular, we must have $f(x^{(k)}) - p^\star \leq \epsilon$ after at most

$$\frac{\log((f(x^{(0)}) - p^\star)/\epsilon)}{\log(1/c)} \tag{9.19}$$

iterations of the gradient method with exact line search.

This bound on the number of iterations required, even though crude, can give some insight into the gradient method. The numerator,

$$\log((f(x^{(0)}) - p^\star)/\epsilon)$$

can be interpreted as the log of the ratio of the initial suboptimality (*i.e.*, gap between $f(x^{(0)})$ and $p^\star$), to the final suboptimality (*i.e.*, less than ϵ). This term suggests that the number of iterations depends on how good the initial point is, and what the final required accuracy is.

The denominator appearing in the bound (9.19), $\log(1/c)$, is a function of M/m, which we have seen is a bound on the condition number of $\nabla^2 f(x)$ over S, or the condition number of the sublevel sets $\{z \mid f(z) \leq \alpha\}$. For large condition number bound M/m, we have

$$\log(1/c) = -\log(1 - m/M) \approx m/M,$$

so our bound on the number of iterations required increases approximately linearly with increasing M/m.

We will see that the gradient method does in fact require a large number of iterations when the Hessian of f, near $x^\star$, has a large condition number. Conversely, when the sublevel sets of f are relatively isotropic, so that the condition number bound M/m can be chosen to be relatively small, the bound (9.18) shows that convergence is rapid, since c is small, or at least not too close to one.

The bound (9.18) shows that the error $f(x^{(k)}) - p^\star$ converges to zero at least as fast as a geometric series. In the context of iterative numerical methods, this is called *linear convergence*, since the error lies below a line on a log-linear plot of error versus iteration number.

Analysis for backtracking line search

Now we consider the case where a backtracking line search is used in the gradient descent method. We will show that the backtracking exit condition,

$$\tilde{f}(t) \le f(x) - \alpha t \|\nabla f(x)\|_2^2,$$

is satisfied whenever $0 \le t \le 1/M$. First note that

$$0 \le t \le 1/M \implies -t + \frac{Mt^2}{2} \le -t/2$$

(which follows from convexity of $-t+Mt^2/2$). Using this result and the bound (9.17), we have, for $0 \le t \le 1/M$,

$$\begin{array}{rcl} \tilde{f}(t) & \le & f(x) - t\|\nabla f(x)\|_2^2 + \dfrac{Mt^2}{2}\|\nabla(f(x))\|_2^2 \\ & \le & f(x) - (t/2)\|\nabla f(x)\|_2^2 \\ & \le & f(x) - \alpha t\|\nabla f(x)\|_2^2, \end{array}$$

since $\alpha < 1/2$. Therefore the backtracking line search terminates either with $t = 1$ or with a value $t \ge \beta/M$. This provides a lower bound on the decrease in the objective function. In the first case we have

$$f(x^+) \le f(x) - \alpha\|\nabla f(x)\|_2^2,$$

and in the second case we have

$$f(x^+) \le f(x) - (\beta\alpha/M)\|\nabla f(x)\|_2^2.$$

Putting these together, we always have

$$f(x^+) \le f(x) - \min\{\alpha, \beta\alpha/M\}\|\nabla f(x)\|_2^2.$$

Now we can proceed exactly as in the case of exact line search. We subtract $p^\star$ from both sides to get

$$f(x^+) - p^\star \le f(x) - p^\star - \min\{\alpha, \beta\alpha/M\}\|\nabla f(x)\|_2^2,$$

and combine this with $\|\nabla f(x)\|_2^2 \ge 2m(f(x) - p^\star)$ to obtain

$$f(x^+) - p^\star \le (1 - \min\{2m\alpha, 2\beta\alpha m/M\})(f(x) - p^\star).$$

From this we conclude

$$f(x^{(k)}) - p^\star \le c^k(f(x^{(0)}) - p^\star)$$

where

$$c = 1 - \min\{2m\alpha, 2\beta\alpha m/M\} < 1.$$

In particular, $f(x^{(k)})$ converges to $p^\star$ at least as fast as a geometric series with an exponent that depends (at least in part) on the condition number bound M/m. In the terminology of iterative methods, the convergence is at least linear.

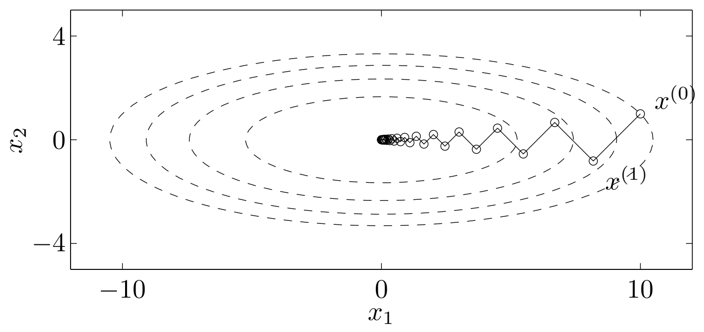

Figure 9.2 Some contour lines of the function $f(x) = (1/2)(x_1^2 + 10x_2^2)$. The condition number of the sublevel sets, which are ellipsoids, is exactly 10. The figure shows the iterates of the gradient method with exact line search, started at $x^{(0)} = (10, 1)$.

9.3.2 Examples

A quadratic problem in $\mathbf{R}^2$

Our first example is very simple. We consider the quadratic objective function on $\mathbf{R}^2$

$$f(x) = \frac{1}{2}(x_1^2 + \gamma x_2^2),$$

where $\gamma > 0$. Clearly, the optimal point is $x^\star = 0$, and the optimal value is 0. The Hessian of f is constant, and has eigenvalues 1 and γ, so the condition numbers of the sublevel sets of f are all exactly

$$\frac{\max\{1, \gamma\}}{\min\{1, \gamma\}} = \max\{\gamma, 1/\gamma\}.$$

The tightest choices for the strong convexity constants m and M are

$$m = \min\{1, \gamma\}, \qquad M = \max\{1, \gamma\}.$$

We apply the gradient descent method with exact line search, starting at the point $x^{(0)} = (\gamma, 1)$. In this case we can derive the following closed-form expressions for the iterates $x^{(k)}$ and their function values (exercise 9.6):

$$x_1^{(k)} = \gamma \left(\frac{\gamma - 1}{\gamma + 1}\right)^k, \qquad x_2^{(k)} = \left(-\frac{\gamma - 1}{\gamma + 1}\right)^k,$$

and

$$f(x^{(k)}) = \frac{\gamma(\gamma + 1)}{2}\left(\frac{\gamma - 1}{\gamma + 1}\right)^{2k} = \left(\frac{\gamma - 1}{\gamma + 1}\right)^{2k} f(x^{(0)}).$$

This is illustrated in figure 9.2, for $\gamma = 10$.

For this simple example, convergence is exactly linear, *i.e.*, the error is exactly a geometric series, reduced by the factor $|(\gamma - 1)/(\gamma + 1)|^2$ at each iteration. For

$\gamma = 1$, the exact solution is found in one iteration; for γ not far from one (say, between 1/3 and 3) convergence is rapid. The convergence is very slow for $\gamma \gg 1$ or $\gamma \ll 1$.

We can compare the convergence with the bound derived above in §9.3.1. Using the least conservative values $m = \min\{1, \gamma\}$ and $M = \max\{1, \gamma\}$, the bound (9.18) guarantees that the error in each iteration is reduced at least by the factor $c = (1 - m/M)$. We have seen that the error is in fact reduced exactly by the factor

$$\left(\frac{1 - m/M}{1 + m/M}\right)^2$$

in each iteration. For small m/M, which corresponds to large condition number, the upper bound (9.19) implies that the number of iterations required to obtain a given level of accuracy grows at most like M/m. For this example, the exact number of iterations required grows approximately like $(M/m)/4$, *i.e.*, one quarter of the value of the bound. This shows that for this simple example, the bound on the number of iterations derived in our simple analysis is only about a factor of four conservative (using the least conservative values for m and M). In particular, the convergence rate (as well as its upper bound) is very dependent on the condition number of the sublevel sets.

A nonquadratic problem in $\mathbf{R}^2$

We now consider a nonquadratic example in $\mathbf{R}^2$, with

$$f(x_1, x_2) = e^{x_1 + 3x_2 - 0.1} + e^{x_1 - 3x_2 - 0.1} + e^{-x_1 - 0.1}. \tag{9.20}$$

We apply the gradient method with a backtracking line search, with $\alpha = 0.1$, $\beta = 0.7$. Figure 9.3 shows some level curves of f, and the iterates $x^{(k)}$ generated by the gradient method (shown as small circles). The lines connecting successive iterates show the scaled steps,

$$x^{(k+1)} - x^{(k)} = -t^{(k)} \nabla f(x^{(k)}).$$

Figure 9.4 shows the error $f(x^{(k)}) - p^\star$ versus iteration k. The plot reveals that the error converges to zero approximately as a geometric series, *i.e.*, the convergence is approximately linear. In this example, the error is reduced from about 10 to about 10^{-7} in 20 iterations, so the error is reduced by a factor of approximately $10^{-8/20} \approx 0.4$ each iteration. This reasonably rapid convergence is predicted by our convergence analysis, since the sublevel sets of f are not too badly conditioned, which in turn means that M/m can be chosen as not too large.

To compare backtracking line search with an exact line search, we use the gradient method with an exact line search, on the same problem, and with the same starting point. The results are given in figures 9.5 and 9.4. Here too the convergence is approximately linear, about twice as fast as the gradient method with backtracking line search. With exact line search, the error is reduced by about 10^{-11} in 15 iterations, *i.e.*, a reduction by a factor of about $10^{-11/15} \approx 0.2$ per iteration.

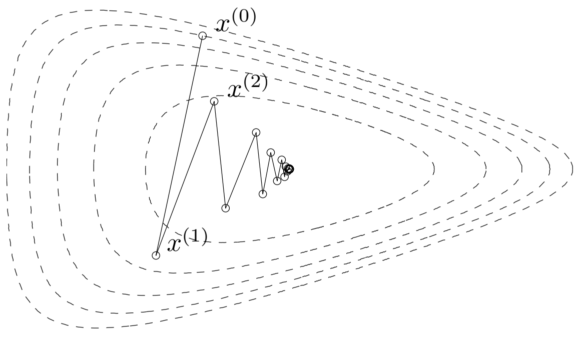

Figure 9.3 Iterates of the gradient method with backtracking line search, for the problem in $\mathbf{R}^2$ with objective f given in (9.20). The dashed curves are level curves of f, and the small circles are the iterates of the gradient method. The solid lines, which connect successive iterates, show the scaled steps $t^{(k)}\Delta x^{(k)}$.

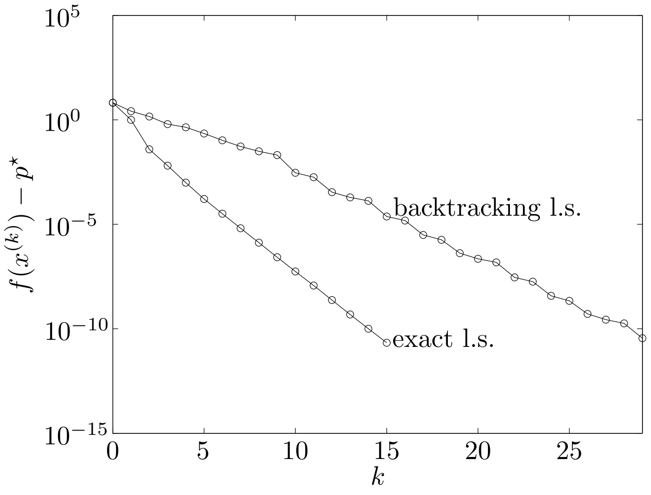

Figure 9.4 Error $f(x^{(k)}) - p^\star$ versus iteration k of the gradient method with backtracking and exact line search, for the problem in $\mathbf{R}^2$ with objective f given in (9.20). The plot shows nearly linear convergence, with the error reduced approximately by the factor 0.4 in each iteration of the gradient method with backtracking line search, and by the factor 0.2 in each iteration of the gradient method with exact line search.

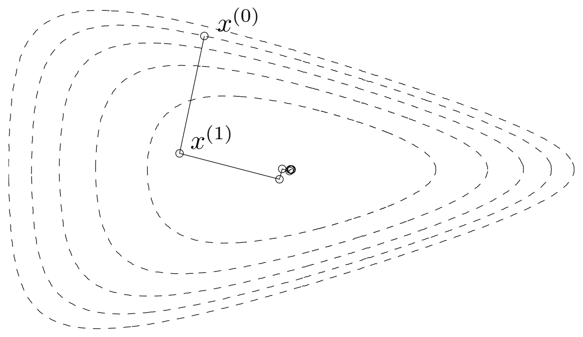

Figure 9.5 Iterates of the gradient method with exact line search for the problem in $\mathbf{R}^2$ with objective f given in (9.20).

A problem in $\mathbf{R}^{100}$

We next consider a larger example, of the form

$$f(x) = c^T x - \sum_{i=1}^{m} \log(b_i - a_i^T x), \tag{9.21}$$

with $m = 500$ terms and $n = 100$ variables.

The progress of the gradient method with backtracking line search, with parameters $\alpha = 0.1$, $\beta = 0.5$, is shown in figure 9.6. In this example we see an initial approximately linear and fairly rapid convergence for about 20 iterations, followed by a slower linear convergence. Overall, the error is reduced by a factor of around 10^6 in around 175 iterations, which gives an average error reduction by a factor of around $10^{-6/175} \approx 0.92$ per iteration. The initial convergence rate, for the first 20 iterations, is around a factor of 0.8 per iteration; the slower final convergence rate, after the first 20 iterations, is around a factor of 0.94 per iteration.

Figure 9.6 shows the convergence of the gradient method with exact line search. The convergence is again approximately linear, with an overall error reduction by approximately a factor $10^{-6/140} \approx 0.91$ per iteration. This is only a bit faster than the gradient method with backtracking line search.

Finally, we examine the influence of the backtracking line search parameters α and β on the convergence rate, by determining the number of iterations required to obtain $f(x^{(k)}) - p^\star \leq 10^{-5}$. In the first experiment, we fix $\beta = 0.5$, and vary α from 0.05 to 0.5. The number of iterations required varies from about 80, for larger values of α, in the range 0.2–0.5, to about 170 for smaller values of α. This, and other experiments, suggest that the gradient method works better with fairly large α, in the range 0.2–0.5.

Similarly, we can study the effect of the choice of β by fixing $\alpha = 0.1$ and varying β from 0.05 to 0.95. Again the variation in the total number of iterations is not large, ranging from around 80 (when $\beta \approx 0.5$) to around 200 (for β small, or near 1). This experiment, and others, suggest that $\beta \approx 0.5$ is a good choice.

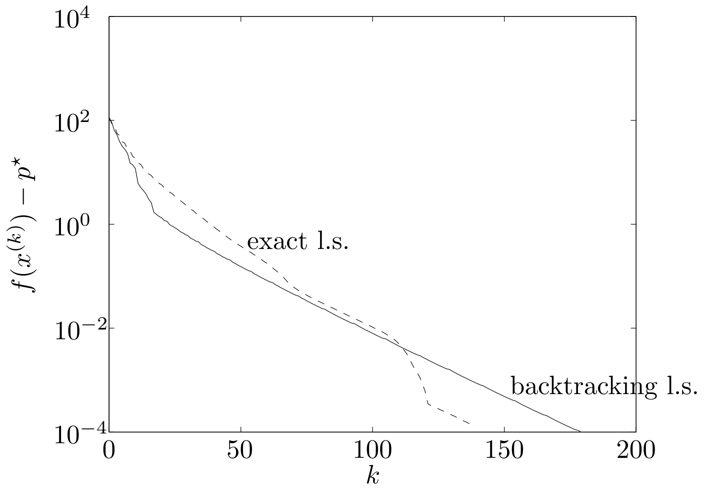

Figure 9.6 Error $f(x^{(k)}) - p^\star$ versus iteration k for the gradient method with backtracking and exact line search, for a problem in $\mathbf{R}^{100}$.

These experiments suggest that the effect of the backtracking parameters on the convergence is not large, no more than a factor of two or so.

Gradient method and condition number

Our last experiment will illustrate the importance of the condition number of $\nabla^2 f(x)$ (or the sublevel sets) on the rate of convergence of the gradient method. We start with the function given by (9.21), but replace the variable x by $x = T\bar{x}$, where

$$T = \mathbf{diag}((1, \gamma^{1/n}, \gamma^{2/n}, \ldots, \gamma^{(n-1)/n})),$$

i.e., we minimize

$$\bar{f}(\bar{x}) = c^T T\bar{x} - \sum_{i=1}^{m} \log(b_i - a_i^T T\bar{x}). \tag{9.22}$$

This gives us a family of optimization problems, indexed by γ, which affects the problem condition number.

Figure 9.7 shows the number of iterations required to achieve $\bar{f}(\bar{x}^{(k)}) - \bar{p}^\star < 10^{-5}$ as a function of γ, using a backtracking line search with $\alpha = 0.3$ and $\beta = 0.7$. This plot shows that for diagonal scaling as small as $10:1$ (*i.e.*, $\gamma = 10$), the number of iterations grows to more than a thousand; for a diagonal scaling of 20 or more, the gradient method slows to essentially useless.

The condition number of the Hessian $\nabla^2 \bar{f}(\bar{x}^\star)$ at the optimum is shown in figure 9.8. For large and small γ, the condition number increases roughly as $\max\{\gamma^2, 1/\gamma^2\}$, in a very similar way as the number of iterations depends on γ. This shows again that the relation between conditioning and convergence speed is a real phenomenon, and not just an artifact of our analysis.

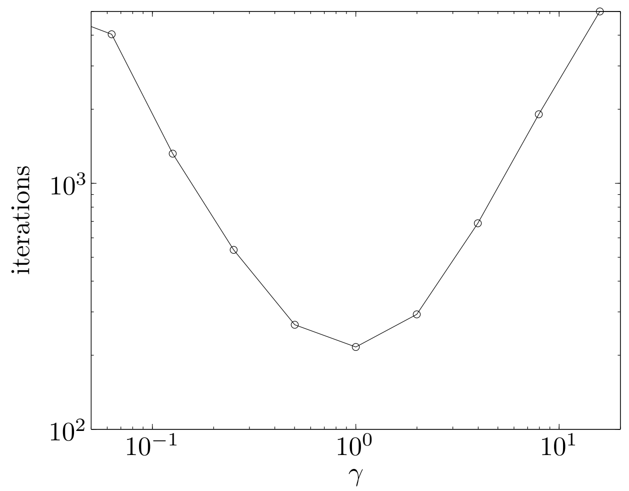

Figure 9.7 Number of iterations of the gradient method applied to problem (9.22). The vertical axis shows the number of iterations required to obtain $\bar{f}(\bar{x}^{(k)}) - \bar{p}^\star < 10^{-5}$. The horizontal axis shows γ, which is a parameter that controls the amount of diagonal scaling. We use a backtracking line search with $\alpha = 0.3$, $\beta = 0.7$.

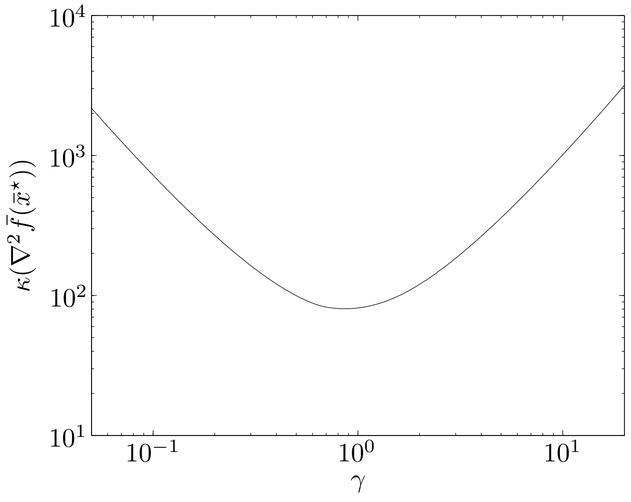

Figure 9.8 Condition number of the Hessian of the function at its minimum, as a function of γ. By comparing this plot with the one in figure 9.7, we see that the condition number has a very strong influence on convergence rate.

Conclusions

From the numerical examples shown, and others, we can make the conclusions summarized below.

- The gradient method often exhibits approximately linear convergence, *i.e.*, the error $f(x^{(k)}) - p^\star$ converges to zero approximately as a geometric series.
- The choice of backtracking parameters α, β has a noticeable but not dramatic effect on the convergence. An exact line search sometimes improves the convergence of the gradient method, but the effect is not large (and probably not worth the trouble of implementing the exact line search).
- The convergence rate depends greatly on the condition number of the Hessian, or the sublevel sets. Convergence can be very slow, even for problems that are moderately well conditioned (say, with condition number in the 100s). When the condition number is larger (say, 1000 or more) the gradient method is so slow that it is useless in practice.

The main advantage of the gradient method is its simplicity. Its main disadvantage is that its convergence rate depends so critically on the condition number of the Hessian or sublevel sets.

9.4 Steepest descent method

The first-order Taylor approximation of $f(x+v)$ around x is

$$f(x+v) \approx \widehat{f}(x+v) = f(x) + \nabla f(x)^T v.$$

The second term on the righthand side, $\nabla f(x)^T v$, is the *directional derivative* of f at x in the direction v. It gives the approximate change in f for a small step v. The step v is a descent direction if the directional derivative is negative.

We now address the question of how to choose v to make the directional derivative as negative as possible. Since the directional derivative $\nabla f(x)^T v$ is linear in v, it can be made as negative as we like by taking v large (provided v is a descent direction, *i.e.*, $\nabla f(x)^T v < 0$). To make the question sensible we have to limit the size of v, or normalize by the length of v.

Let $\|\cdot\|$ be any norm on $\mathbf{R}^n$. We define a *normalized steepest descent direction* (with respect to the norm $\|\cdot\|$) as

$$\Delta x_{\mathrm{nsd}} = \mathrm{argmin}\{\nabla f(x)^T v \mid \|v\| = 1\}. \qquad (9.23)$$

(We say 'a' steepest descent direction because there can be multiple minimizers.) A normalized steepest descent direction Δx_{nsd} is a step of unit norm that gives the largest decrease in the linear approximation of f.

A normalized steepest descent direction can be interpreted geometrically as follows. We can just as well define Δx_{nsd} as

$$\Delta x_{\mathrm{nsd}} = \mathrm{argmin}\{\nabla f(x)^T v \mid \|v\| \le 1\},$$

i.e., as the direction in the unit ball of $\|\cdot\|$ that extends farthest in the direction $-\nabla f(x)$.

It is also convenient to consider a steepest descent step $\Delta x_{\rm sd}$ that is *unnormalized*, by scaling the normalized steepest descent direction in a particular way:

$$\Delta x_{\rm sd} = \|\nabla f(x)\|_* \Delta x_{\rm nsd}, \tag{9.24}$$

where $\|\cdot\|_*$ denotes the dual norm. Note that for the steepest descent step, we have

$$\nabla f(x)^T \Delta x_{\rm sd} = \|\nabla f(x)\|_* \nabla f(x)^T \Delta x_{\rm nsd} = -\|\nabla f(x)\|_*^2$$

(see exercise 9.7).

The *steepest descent method* uses the steepest descent direction as search direction.

Algorithm 9.4 *Steepest descent method.*

```
given a starting point $x \in \mathbf{dom}\, f$.
repeat
    1. Compute steepest descent direction $\Delta x_{\rm sd}$.
    2. Line search. Choose $t$ via backtracking or exact line search.
    3. Update. $x := x + t\Delta x_{\rm sd}$.
until stopping criterion is satisfied.
```

When exact line search is used, scale factors in the descent direction have no effect, so the normalized or unnormalized direction can be used.

9.4.1 Steepest descent for Euclidean and quadratic norms

Steepest descent for Euclidean norm

If we take the norm $\|\cdot\|$ to be the Euclidean norm we find that the steepest descent direction is simply the negative gradient, *i.e.*, $\Delta x_{\rm sd} = -\nabla f(x)$. The steepest descent method for the Euclidean norm coincides with the gradient descent method.

Steepest descent for quadratic norm

We consider the quadratic norm

$$\|z\|_P = (z^T P z)^{1/2} = \|P^{1/2} z\|_2,$$

where $P \in \mathbf{S}^n_{++}$. The normalized steepest descent direction is given by

$$\Delta x_{\rm nsd} = -\left(\nabla f(x)^T P^{-1} \nabla f(x)\right)^{-1/2} P^{-1} \nabla f(x).$$

The dual norm is given by $\|z\|_* = \|P^{-1/2} z\|_2$, so the steepest descent step with respect to $\|\cdot\|_P$ is given by

$$\Delta x_{\rm sd} = -P^{-1} \nabla f(x). \tag{9.25}$$

The normalized steepest descent direction for a quadratic norm is illustrated in figure 9.9.

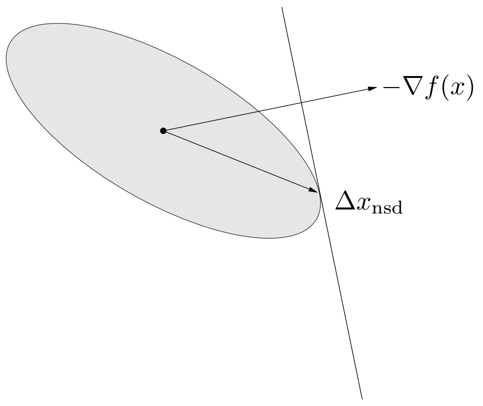

Figure 9.9 Normalized steepest descent direction for a quadratic norm. The ellipsoid shown is the unit ball of the norm, translated to the point x. The normalized steepest descent direction $\Delta x_{\rm nsd}$ at x extends as far as possible in the direction $-\nabla f(x)$ while staying in the ellipsoid. The gradient and normalized steepest descent directions are shown.

Interpretation via change of coordinates

We can give an interesting alternative interpretation of the steepest descent direction $\Delta x_{\rm sd}$ as the gradient search direction after a change of coordinates is applied to the problem. Define $\bar{u} = P^{1/2}u$, so we have $\|u\|_P = \|\bar{u}\|_2$. Using this change of coordinates, we can solve the original problem of minimizing f by solving the equivalent problem of minimizing the function $\bar{f} : \mathbf{R}^n \to \mathbf{R}$, given by

$$\bar{f}(\bar{u}) = f(P^{-1/2}\bar{u}) = f(u).$$

If we apply the gradient method to $\bar{f}$, the search direction at a point $\bar{x}$ (which corresponds to the point $x = P^{-1/2}\bar{x}$ for the original problem) is

$$\Delta\bar{x} = -\nabla\bar{f}(\bar{x}) = -P^{-1/2}\nabla f(P^{-1/2}\bar{x}) = -P^{-1/2}\nabla f(x).$$

This gradient search direction corresponds to the direction

$$\Delta x = P^{-1/2}\left(-P^{-1/2}\nabla f(x)\right) = -P^{-1}\nabla f(x)$$

for the original variable x. In other words, the steepest descent method in the quadratic norm $\|\cdot\|_P$ can be thought of as the gradient method applied to the problem after the change of coordinates $\bar{x} = P^{1/2}x$.

9.4.2 Steepest descent for ℓ_1-norm

As another example, we consider the steepest descent method for the ℓ_1-norm. A normalized steepest descent direction,

$$\Delta x_{\rm nsd} = \operatorname{argmin}\{\nabla f(x)^T v \mid \|v\|_1 \leq 1\},$$

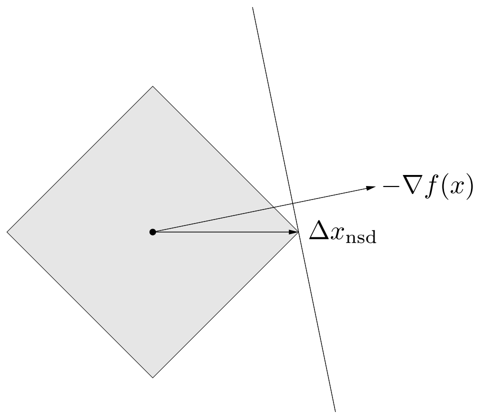

Figure 9.10 Normalized steepest descent direction for the ℓ_1-norm. The diamond is the unit ball of the ℓ_1-norm, translated to the point x. The normalized steepest descent direction can always be chosen in the direction of a standard basis vector; in this example we have $\Delta x_{\mathrm{nsd}} = e_1$.

is easily characterized. Let i be any index for which $\|\nabla f(x)\|_\infty = |(\nabla f(x))_i|$. Then a normalized steepest descent direction Δx_{nsd} for the ℓ_1-norm is given by

$$\Delta x_{\mathrm{nsd}} = -\mathrm{sign}\left(\frac{\partial f(x)}{\partial x_i}\right) e_i,$$

where e_i is the ith standard basis vector. An unnormalized steepest descent step is then

$$\Delta x_{\mathrm{sd}} = \Delta x_{\mathrm{nsd}} \|\nabla f(x)\|_\infty = -\frac{\partial f(x)}{\partial x_i} e_i.$$

Thus, the normalized steepest descent step in ℓ_1-norm can always be chosen to be a standard basis vector (or a negative standard basis vector). It is the coordinate axis direction along which the approximate decrease in f is greatest. This is illustrated in figure 9.10.

The steepest descent algorithm in the ℓ_1-norm has a very natural interpretation: At each iteration we select a component of $\nabla f(x)$ with maximum absolute value, and then decrease or increase the corresponding component of x, according to the sign of $(\nabla f(x))_i$. The algorithm is sometimes called a *coordinate-descent* algorithm, since only one component of the variable x is updated at each iteration. This can greatly simplify, or even trivialize, the line search.

Example 9.2 *Frobenius norm scaling.* In §4.5.4 we encountered the unconstrained geometric program

$$\text{minimize} \quad \textstyle\sum_{i,j=1}^n M_{ij}^2 d_i^2 / d_j^2,$$

where $M \in \mathbf{R}^{n \times n}$ is given, and the variable is $d \in \mathbf{R}^n$. Using the change of variables $x_i = 2 \log d_i$ we can express this geometric program in convex form as

$$\text{minimize} \quad f(x) = \log\left(\textstyle\sum_{i,j=1}^n M_{ij}^2 e^{x_i - x_j}\right).$$

It is easy to minimize f one component at a time. Keeping all components except the kth fixed, we can write $f(x) = \log(\alpha_k + \beta_k e^{-x_k} + \gamma_k e^{x_k})$, where

$$\alpha_k = M_{kk}^2 + \sum_{i,j \neq k} M_{ij}^2 e^{x_i - x_j}, \qquad \beta_k = \sum_{i \neq k} M_{ik}^2 e^{x_i}, \qquad \gamma_k = \sum_{j \neq k} M_{kj}^2 e^{-x_j}.$$

The minimum of $f(x)$, as a function of x_k, is obtained for $x_k = \log(\beta_k/\gamma_k)/2$. So for this problem an exact line search can be carried out using a simple analytical formula.

The ℓ_1-steepest descent algorithm with exact line search consists of repeating the following steps.

1. Compute the gradient
$$(\nabla f(x))_i = \frac{-\beta_i e^{-x_i} + \gamma_i e^{x_i}}{\alpha_i + \beta_i e^{-x_i} + \gamma_i e^{x_i}}, \quad i = 1, \ldots, n.$$
2. Select a largest (in absolute value) component of $\nabla f(x)$: $|\nabla f(x)|_k = \|\nabla f(x)\|_\infty$.
3. Minimize f over the scalar variable x_k, by setting $x_k = \log(\beta_k/\gamma_k)/2$.

9.4.3 Convergence analysis

In this section we extend the convergence analysis for the gradient method with backtracking line search to the steepest descent method for an arbitrary norm. We will use the fact that any norm can be bounded in terms of the Euclidean norm, so there exists constants $\gamma,\ \tilde{\gamma} \in (0,1]$ such that

$$\|x\| \geq \gamma \|x\|_2, \qquad \|x\|_* \geq \tilde{\gamma} \|x\|_2$$

(see §A.1.4).

Again we assume f is strongly convex on the initial sublevel set S. The upper bound $\nabla^2 f(x) \preceq MI$ implies an upper bound on the function $f(x + t\Delta x_{\mathrm{sd}})$ as a function of t:

$$\begin{aligned} f(x + t\Delta x_{\mathrm{sd}}) &\leq f(x) + t\nabla f(x)^T \Delta x_{\mathrm{sd}} + \frac{M\|\Delta x_{\mathrm{sd}}\|_2^2}{2} t^2 \\ &\leq f(x) + t\nabla f(x)^T \Delta x_{\mathrm{sd}} + \frac{M\|\Delta x_{\mathrm{sd}}\|^2}{2\gamma^2} t^2 \\ &= f(x) - t\|\nabla f(x)\|_*^2 + \frac{M}{2\gamma^2} t^2 \|\nabla f(x)\|_*^2. \end{aligned} \tag{9.26}$$

The step size $\hat{t} = \gamma^2/M$ (which minimizes the quadratic upper bound (9.26)) satisfies the exit condition for the backtracking line search:

$$f(x + \hat{t}\Delta x_{\mathrm{sd}}) \leq f(x) - \frac{\gamma^2}{2M}\|\nabla f(x)\|_*^2 \leq f(x) + \frac{\alpha\gamma^2}{M}\nabla f(x)^T \Delta x_{\mathrm{sd}} \tag{9.27}$$

since $\alpha < 1/2$ and $\nabla f(x)^T \Delta x_{\mathrm{sd}} = -\|\nabla f(x)\|_*^2$. The line search therefore returns a step size $t \geq \min\{1, \beta\gamma^2/M\}$, and we have

$$\begin{array}{rcl} f(x^+) = f(x + t\Delta x_{\mathrm{sd}}) & \leq & f(x) - \alpha \min\{1, \beta\gamma^2/M\} \|\nabla f(x)\|_*^2 \\ & \leq & f(x) - \alpha\tilde{\gamma}^2 \min\{1, \beta\gamma^2/M\} \|\nabla f(x)\|_2^2. \end{array}$$

Subtracting $p^\star$ from both sides and using (9.9), we obtain

$$f(x^+) - p^\star \leq c(f(x) - p^\star),$$

where

$$c = 1 - 2m\alpha\tilde{\gamma}^2 \min\{1, \beta\gamma^2/M\} < 1.$$

Therefore we have

$$f(x^{(k)}) - p^\star \leq c^k (f(x^{(0)}) - p^\star),$$

i.e., linear convergence exactly as in the gradient method.

9.4.4 Discussion and examples

Choice of norm for steepest descent

The choice of norm used to define the steepest descent direction can have a dramatic effect on the convergence rate. For simplicity, we consider the case of steepest descent with quadratic P-norm. In §9.4.1, we showed that the steepest descent method with quadratic P-norm is the same as the gradient method applied to the problem after the change of coordinates $\bar{x} = P^{1/2}x$. We know that the gradient method works well when the condition numbers of the sublevel sets (or the Hessian near the optimal point) are moderate, and works poorly when the condition numbers are large. It follows that when the sublevel sets, after the change of coordinates $\bar{x} = P^{1/2}x$, are moderately conditioned, the steepest descent method will work well.

This observation provides a prescription for choosing P: It should be chosen so that the sublevel sets of f, transformed by $P^{-1/2}$, are well conditioned. For example if an approximation $\hat{H}$ of the Hessian at the optimal point $H(x^\star)$ were known, a very good choice of P would be $P = \hat{H}$, since the Hessian of $\tilde{f}$ at the optimum is then

$$\hat{H}^{-1/2} \nabla^2 f(x^\star) \hat{H}^{-1/2} \approx I,$$

and so is likely to have a low condition number.

This same idea can be described without a change of coordinates. Saying that a sublevel set has low condition number after the change of coordinates $\bar{x} = P^{1/2}x$ is the same as saying that the ellipsoid

$$\mathcal{E} = \{x \mid x^T P x \leq 1\}$$

approximates the shape of the sublevel set. (In other words, it gives a good approximation after appropriate scaling and translation.)

This dependence of the convergence rate on the choice of P can be viewed from two sides. The optimist's viewpoint is that for any problem, there is always a

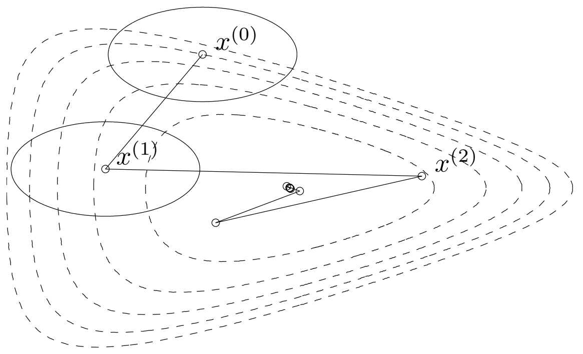

Figure 9.11 Steepest descent method with a quadratic norm $\|\cdot\|_{P_1}$. The ellipses are the boundaries of the norm balls $\{x \mid \|x - x^{(k)}\|_{P_1} \le 1\}$ at $x^{(0)}$ and $x^{(1)}$.

choice of P for which the steepest descent method works very well. The challenge, of course, is to find such a P. The pessimist's viewpoint is that for any problem, there are a huge number of choices of P for which steepest descent works very poorly. In summary, we can say that the steepest descent method works well in cases where we can identify a matrix P for which the transformed problem has moderate condition number.

Examples

In this section we illustrate some of these ideas using the nonquadratic problem in $\mathbf{R}^2$ with objective function (9.20). We apply the steepest descent method to the problem, using the two quadratic norms defined by

$$P_1 = \begin{bmatrix} 2 & 0 \\ 0 & 8 \end{bmatrix}, \qquad P_2 = \begin{bmatrix} 8 & 0 \\ 0 & 2 \end{bmatrix}.$$

In both cases we use a backtracking line search with $\alpha = 0.1$ and $\beta = 0.7$.

Figures 9.11 and 9.12 show the iterates for steepest descent with norm $\|\cdot\|_{P_1}$ and norm $\|\cdot\|_{P_2}$. Figure 9.13 shows the error versus iteration number for both norms. Figure 9.13 shows that the choice of norm strongly influences the convergence. With the norm $\|\cdot\|_{P_1}$, convergence is a bit more rapid than the gradient method, whereas with the norm $\|\cdot\|_{P_2}$, convergence is far slower.

This can be explained by examining the problems after the changes of coordinates $\bar{x} = P_1^{1/2}x$ and $\bar{x} = P_2^{1/2}x$, respectively. Figures 9.14 and 9.15 show the problems in the transformed coordinates. The change of variables associated with P_1 yields sublevel sets with modest condition number, so convergence is fast. The change of variables associated with P_2 yields sublevel sets that are more poorly conditioned, which explains the slower convergence.

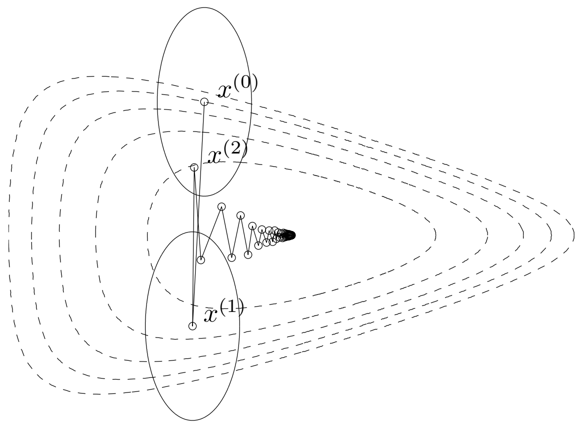

Figure 9.12 Steepest descent method, with quadratic norm $\|\cdot\|_{P_2}$.

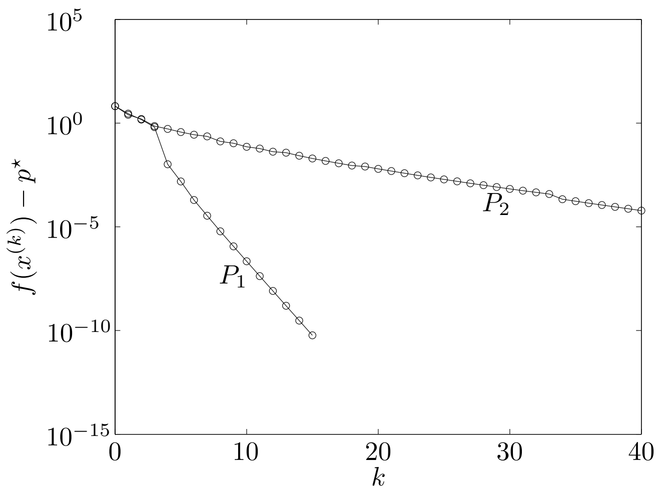

Figure 9.13 Error $f(x^{(k)}) - p^\star$ versus iteration k, for the steepest descent method with the quadratic norm $\|\cdot\|_{P_1}$ and the quadratic norm $\|\cdot\|_{P_2}$. Convergence is rapid for the norm $\|\cdot\|_{P_1}$ and very slow for $\|\cdot\|_{P_2}$.

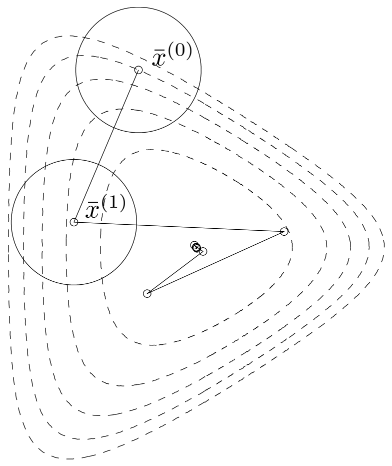

Figure 9.14 The iterates of steepest descent with norm $\|\cdot\|_{P_1}$, after the change of coordinates. This change of coordinates reduces the condition number of the sublevel sets, and so speeds up convergence.

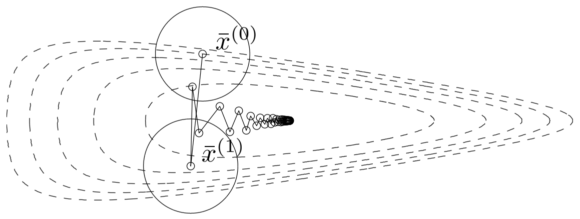

Figure 9.15 The iterates of steepest descent with norm $\|\cdot\|_{P_2}$, after the change of coordinates. This change of coordinates increases the condition number of the sublevel sets, and so slows down convergence.

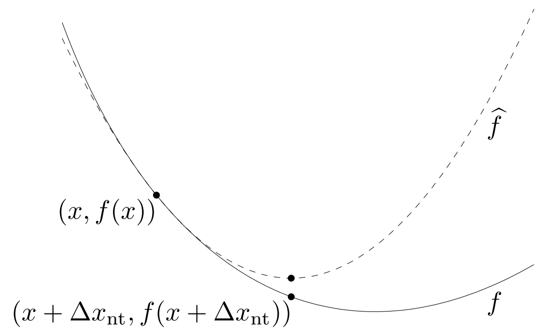

Figure 9.16 The function f (shown solid) and its second-order approximation $\widehat{f}$ at x (dashed). The Newton step $\Delta x_{\rm nt}$ is what must be added to x to give the minimizer of $\widehat{f}$.

9.5 Newton's method

9.5.1 The Newton step

For $x \in \mathbf{dom}\, f$, the vector

$$\Delta x_{\rm nt} = -\nabla^2 f(x)^{-1} \nabla f(x)$$

is called the *Newton step* (for f, at x). Positive definiteness of $\nabla^2 f(x)$ implies that

$$\nabla f(x)^T \Delta x_{\rm nt} = -\nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x) < 0$$

unless $\nabla f(x) = 0$, so the Newton step is a descent direction (unless x is optimal). The Newton step can be interpreted and motivated in several ways.

Minimizer of second-order approximation

The second-order Taylor approximation (or model) $\widehat{f}$ of f at x is

$$\widehat{f}(x + v) = f(x) + \nabla f(x)^T v + \frac{1}{2} v^T \nabla^2 f(x) v, \qquad (9.28)$$

which is a convex quadratic function of v, and is minimized when $v = \Delta x_{\rm nt}$. Thus, the Newton step $\Delta x_{\rm nt}$ is what should be added to the point x to minimize the second-order approximation of f at x. This is illustrated in figure 9.16.

This interpretation gives us some insight into the Newton step. If the function f is quadratic, then $x + \Delta x_{\rm nt}$ is the exact minimizer of f. If the function f is nearly quadratic, intuition suggests that $x + \Delta x_{\rm nt}$ should be a very good estimate of the minimizer of f, *i.e.*, $x^\star$. Since f is twice differentiable, the quadratic model of f will be very accurate when x is near $x^\star$. It follows that when x is near $x^\star$, the point $x + \Delta x_{\rm nt}$ should be a very good estimate of $x^\star$. We will see that this intuition is correct.

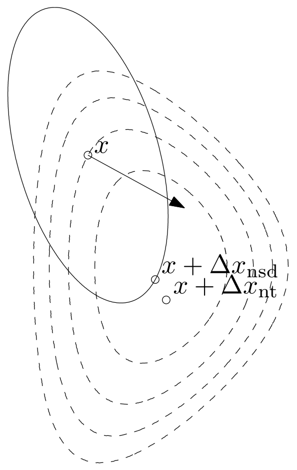

Figure 9.17 The dashed lines are level curves of a convex function. The ellipsoid shown (with solid line) is $\{x + v \mid v^T \nabla^2 f(x) v \leq 1\}$. The arrow shows $-\nabla f(x)$, the gradient descent direction. The Newton step $\Delta x_{\rm nt}$ is the steepest descent direction in the norm $\|\cdot\|_{\nabla^2 f(x)}$. The figure also shows $\Delta x_{\rm nsd}$, the normalized steepest descent direction for the same norm.

Steepest descent direction in Hessian norm

The Newton step is also the steepest descent direction at x, for the quadratic norm defined by the Hessian $\nabla^2 f(x)$, *i.e.*,

$$\|u\|_{\nabla^2 f(x)} = (u^T \nabla^2 f(x) u)^{1/2}.$$

This gives another insight into why the Newton step should be a good search direction, and a very good search direction when x is near $x^\star$.

Recall from our discussion above that steepest descent, with quadratic norm $\|\cdot\|_P$, converges very rapidly when the Hessian, after the associated change of coordinates, has small condition number. In particular, near $x^\star$, a very good choice is $P = \nabla^2 f(x^\star)$. When x is near $x^\star$, we have $\nabla^2 f(x) \approx \nabla^2 f(x^\star)$, which explains why the Newton step is a very good choice of search direction. This is illustrated in figure 9.17.

Solution of linearized optimality condition

If we linearize the optimality condition $\nabla f(x^\star) = 0$ near x we obtain

$$\nabla f(x + v) \approx \nabla f(x) + \nabla^2 f(x) v = 0,$$

which is a linear equation in v, with solution $v = \Delta x_{\rm nt}$. So the Newton step $\Delta x_{\rm nt}$ is what must be added to x so that the linearized optimality condition holds. Again, this suggests that when x is near $x^\star$ (so the optimality conditions almost hold), the update $x + \Delta x_{\rm nt}$ should be a very good approximation of $x^\star$.

When $n = 1$, *i.e.*, $f : \mathbf{R} \to \mathbf{R}$, this interpretation is particularly simple. The solution $x^\star$ of the minimization problem is characterized by $f'(x^\star) = 0$, *i.e.*, it is

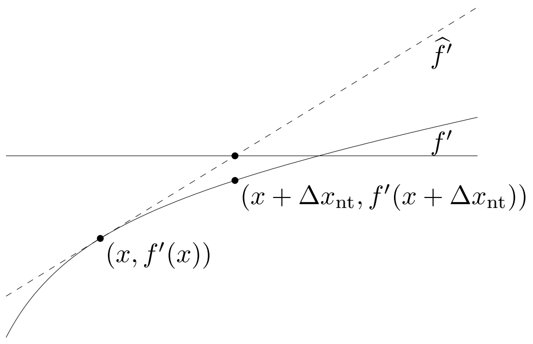

Figure 9.18 The solid curve is the derivative f' of the function f shown in figure 9.16. $\widehat{f'}$ is the linear approximation of f' at x. The Newton step $\Delta x_{\rm nt}$ is the difference between the root of $\widehat{f'}$ and the point x.

the zero-crossing of the derivative f', which is monotonically increasing since f is convex. Given our current approximation x of the solution, we form a first-order Taylor approximation of f' at x. The zero-crossing of this affine approximation is then $x + \Delta x_{\rm nt}$. This interpretation is illustrated in figure 9.18.

Affine invariance of the Newton step

An important feature of the Newton step is that it is independent of linear (or affine) changes of coordinates. Suppose $T \in \mathbf{R}^{n\times n}$ is nonsingular, and define $\bar{f}(y) = f(Ty)$. Then we have

$$\nabla \bar{f}(y) = T^T \nabla f(x), \qquad \nabla^2 \bar{f}(y) = T^T \nabla^2 f(x) T,$$

where $x = Ty$. The Newton step for $\bar{f}$ at y is therefore

$$\begin{array}{rcl} \Delta y_{\rm nt} & = & -\left(T^T \nabla^2 f(x) T\right)^{-1} \left(T^T \nabla f(x)\right) \\ & = & -T^{-1} \nabla^2 f(x)^{-1} \nabla f(x) \\ & = & T^{-1} \Delta x_{\rm nt}, \end{array}$$

where $\Delta x_{\rm nt}$ is the Newton step for f at x. Hence the Newton steps of f and $\bar{f}$ are related by the same linear transformation, and

$$x + \Delta x_{\rm nt} = T(y + \Delta y_{\rm nt}).$$

The Newton decrement

The quantity

$$\lambda(x) = \left(\nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x)\right)^{1/2}$$

is called the *Newton decrement* at x. We will see that the Newton decrement plays an important role in the analysis of Newton's method, and is also useful as a stopping criterion. We can relate the Newton decrement to the quantity $f(x) - \inf_y \widehat{f}(y)$, where $\widehat{f}$ is the second-order approximation of f at x:

$$f(x) - \inf_y \widehat{f}(y) = f(x) - \widehat{f}(x + \Delta x_{\rm nt}) = \frac{1}{2}\lambda(x)^2.$$

Thus, $\lambda^2/2$ is an estimate of $f(x) - p^\star$, based on the quadratic approximation of f at x.

We can also express the Newton decrement as

$$\lambda(x) = \left(\Delta x_{\rm nt}^T \nabla^2 f(x) \Delta x_{\rm nt}\right)^{1/2}. \tag{9.29}$$

This shows that λ is the norm of the Newton step, in the quadratic norm defined by the Hessian, *i.e.*, the norm

$$\|u\|_{\nabla^2 f(x)} = \left(u^T \nabla^2 f(x) u\right)^{1/2}.$$

The Newton decrement comes up in backtracking line search as well, since we have

$$\nabla f(x)^T \Delta x_{\rm nt} = -\lambda(x)^2. \tag{9.30}$$

This is the constant used in a backtracking line search, and can be interpreted as the directional derivative of f at x in the direction of the Newton step:

$$-\lambda(x)^2 = \nabla f(x)^T \Delta x_{\rm nt} = \left.\frac{d}{dt} f(x + \Delta x_{\rm nt} t)\right|_{t=0}.$$

Finally, we note that the Newton decrement is, like the Newton step, affine invariant. In other words, the Newton decrement of $\bar{f}(y) = f(Ty)$ at y, where T is nonsingular, is the same as the Newton decrement of f at $x = Ty$.

9.5.2 Newton's method

Newton's method, as outlined below, is sometimes called the *damped* Newton method or *guarded* Newton method, to distinguish it from the *pure* Newton method, which uses a fixed step size $t = 1$.

Algorithm 9.5 *Newton's method.*

```
given a starting point $x \in \mathbf{dom}\, f$, tolerance $\epsilon > 0$.
repeat
    1. Compute the Newton step and decrement.
        $\Delta x_{\rm nt} := -\nabla^2 f(x)^{-1} \nabla f(x)$;    $\lambda^2 := \nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x)$.
    2. Stopping criterion. quit if $\lambda^2/2 \le \epsilon$.
    3. Line search. Choose step size $t$ by backtracking line search.
    4. Update. $x := x + t\Delta x_{\rm nt}$.
```

This is essentially the general descent method described in §9.2, using the Newton step as search direction. The only difference (which is very minor) is that the stopping criterion is checked after computing the search direction, rather than after the update.

9.5.3 Convergence analysis

We assume, as before, that f is twice continuously differentiable, and strongly convex with constant m, *i.e.*, $\nabla^2 f(x) \succeq mI$ for $x \in S$. We have seen that this also implies that there exists an $M > 0$ such that $\nabla^2 f(x) \preceq MI$ for all $x \in S$.

In addition, we assume that the Hessian of f is Lipschitz continuous on S with constant L, *i.e.*,

$$\|\nabla^2 f(x) - \nabla^2 f(y)\|_2 \le L\|x-y\|_2 \tag{9.31}$$

for all x, $y \in S$. The coefficient L, which can be interpreted as a bound on the third derivative of f, can be taken to be zero for a quadratic function. More generally L measures how well f can be approximated by a quadratic model, so we can expect the Lipschitz constant L to play a critical role in the performance of Newton's method. Intuition suggests that Newton's method will work very well for a function whose quadratic model varies slowly (*i.e.*, has small L).

Idea and outline of convergence proof

We first give the idea and outline of the convergence proof, and the main conclusion, and then the details of the proof. We will show there are numbers η and γ with $0 < \eta \le m^2/L$ and $\gamma > 0$ such that the following hold.

- If $\|\nabla f(x^{(k)})\|_2 \ge \eta$, then

$$f(x^{(k+1)}) - f(x^{(k)}) \le -\gamma. \tag{9.32}$$

- If $\|\nabla f(x^{(k)})\|_2 < \eta$, then the backtracking line search selects $t^{(k)} = 1$ and

$$\frac{L}{2m^2}\|\nabla f(x^{(k+1)})\|_2 \le \left(\frac{L}{2m^2}\|\nabla f(x^{(k)})\|_2\right)^2. \tag{9.33}$$

Let us analyze the implications of the second condition. Suppose that it is satisfied for iteration k, *i.e.*, $\|\nabla f(x^{(k)})\|_2 < \eta$. Since $\eta \le m^2/L$, we have $\|\nabla f(x^{(k+1)})\|_2 < \eta$, *i.e.*, the second condition is also satisfied at iteration $k+1$. Continuing recursively, we conclude that once the second condition holds, it will hold for all future iterates, *i.e.*, for all $l \ge k$, we have $\|\nabla f(x^{(l)})\|_2 < \eta$. Therefore for all $l \ge k$, the algorithm takes a full Newton step $t = 1$, and

$$\frac{L}{2m^2}\|\nabla f(x^{(l+1)})\|_2 \le \left(\frac{L}{2m^2}\|\nabla f(x^{(l)})\|_2\right)^2. \tag{9.34}$$

Applying this inequality recursively, we find that for $l \ge k$,

$$\frac{L}{2m^2}\|\nabla f(x^{(l)})\|_2 \le \left(\frac{L}{2m^2}\|\nabla f(x^{(k)})\|_2\right)^{2^{l-k}} \le \left(\frac{1}{2}\right)^{2^{l-k}},$$

and hence

$$f(x^{(l)}) - p^\star \le \frac{1}{2m}\|\nabla f(x^{(l)})\|_2^2 \le \frac{2m^3}{L^2}\left(\frac{1}{2}\right)^{2^{l-k+1}}. \tag{9.35}$$

This last inequality shows that convergence is extremely rapid once the second condition is satisfied. This phenomenon is called *quadratic convergence*. Roughly speaking, the inequality (9.35) means that, after a sufficiently large number of iterations, the number of correct digits doubles at each iteration.

The iterations in Newton's method naturally fall into two stages. The second stage, which occurs once the condition $\|\nabla f(x)\|_2 \leq \eta$ holds, is called the *quadratically convergent stage*. We refer to the first stage as the *damped Newton phase*, because the algorithm can choose a step size $t < 1$. The quadratically convergent stage is also called the *pure Newton* phase, since in these iterations a step size $t = 1$ is always chosen.

Now we can estimate the total complexity. First we derive an upper bound on the number of iterations in the damped Newton phase. Since f decreases by at least γ at each iteration, the number of damped Newton steps cannot exceed

$$\frac{f(x^{(0)}) - p^\star}{\gamma},$$

since if it did, f would be less than $p^\star$, which is impossible.

We can bound the number of iterations in the quadratically convergent phase using the inequality (9.35). It implies that we must have $f(x) - p^\star \leq \epsilon$ after no more than

$$\log_2 \log_2(\epsilon_0/\epsilon)$$

iterations in the quadratically convergent phase, where $\epsilon_0 = 2m^3/L^2$.

Overall, then, the number of iterations until $f(x) - p^\star \leq \epsilon$ is bounded above by

$$\frac{f(x^{(0)}) - p^\star}{\gamma} + \log_2 \log_2(\epsilon_0/\epsilon). \tag{9.36}$$

The term $\log_2 \log_2(\epsilon_0/\epsilon)$, which bounds the number of iterations in the quadratically convergent phase, grows *extremely slowly* with required accuracy ϵ, and can be considered a constant for practical purposes, say five or six. (Six iterations of the quadratically convergent stage gives an accuracy of about $\epsilon \approx 5 \cdot 10^{-20}\epsilon_0$.)

Not quite accurately, then, we can say that the number of Newton iterations required to minimize f is bounded above by

$$\frac{f(x^{(0)}) - p^\star}{\gamma} + 6. \tag{9.37}$$

A more precise statement is that (9.37) is a bound on the number of iterations to compute an extremely good approximation of the solution.

Damped Newton phase

We now establish the inequality (9.32). Assume $\|\nabla f(x)\|_2 \geq \eta$. We first derive a lower bound on the step size selected by the line search. Strong convexity implies that $\nabla^2 f(x) \preceq MI$ on S, and therefore

$$\begin{array}{rcl} f(x + t\Delta x_{\mathrm{nt}}) & \leq & f(x) + t\nabla f(x)^T \Delta x_{\mathrm{nt}} + \dfrac{M\|\Delta x_{\mathrm{nt}}\|_2^2}{2} t^2 \\ & \leq & f(x) - t\lambda(x)^2 + \dfrac{M}{2m} t^2 \lambda(x)^2, \end{array}$$

where we use (9.30) and

$$\lambda(x)^2 = \Delta x_{\rm nt}^T \nabla^2 f(x) \Delta x_{\rm nt} \geq m\|\Delta x_{\rm nt}\|_2^2.$$

The step size $\hat{t} = m/M$ satisfies the exit condition of the line search, since

$$f(x + \hat{t}\Delta x_{\rm nt}) \leq f(x) - \frac{m}{2M}\lambda(x)^2 \leq f(x) - \alpha\hat{t}\lambda(x)^2.$$

Therefore the line search returns a step size $t \geq \beta m/M$, resulting in a decrease of the objective function

$$\begin{array}{rcl} f(x^+) - f(x) & \leq & -\alpha t \lambda(x)^2 \\ & \leq & -\alpha\beta \dfrac{m}{M}\lambda(x)^2 \\ & \leq & -\alpha\beta \dfrac{m}{M^2}\|\nabla f(x)\|_2^2 \\ & \leq & -\alpha\beta\eta^2 \dfrac{m}{M^2}, \end{array}$$

where we use

$$\lambda(x)^2 = \nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x) \geq (1/M)\|\nabla f(x)\|_2^2.$$

Therefore, (9.32) is satisfied with

$$\gamma = \alpha\beta\eta^2 \frac{m}{M^2}. \tag{9.38}$$

Quadratically convergent phase

We now establish the inequality (9.33). Assume $\|\nabla f(x)\|_2 < \eta$. We first show that the backtracking line search selects unit steps, provided

$$\eta \leq 3(1 - 2\alpha)\frac{m^2}{L}.$$

By the Lipschitz condition (9.31), we have, for $t \geq 0$,

$$\|\nabla^2 f(x + t\Delta x_{\rm nt}) - \nabla^2 f(x)\|_2 \leq tL\|\Delta x_{\rm nt}\|_2,$$

and therefore

$$\left|\Delta x_{\rm nt}^T \left(\nabla^2 f(x + t\Delta x_{\rm nt}) - \nabla^2 f(x)\right) \Delta x_{\rm nt}\right| \leq tL\|\Delta x_{\rm nt}\|_2^3.$$

With $\tilde{f}(t) = f(x + t\Delta x_{\rm nt})$, we have $\tilde{f}''(t) = \Delta x_{\rm nt}^T \nabla^2 f(x + t\Delta x_{\rm nt})\Delta x_{\rm nt}$, so the inequality above is

$$|\tilde{f}''(t) - \tilde{f}''(0)| \leq tL\|\Delta x_{\rm nt}\|_2^3.$$

We will use this inequality to determine an upper bound on $\tilde{f}(t)$. We start with

$$\tilde{f}''(t) \leq \tilde{f}''(0) + tL\|\Delta x_{\rm nt}\|_2^3 \leq \lambda(x)^2 + t\frac{L}{m^{3/2}}\lambda(x)^3,$$

where we use $\tilde{f}''(0) = \lambda(x)^2$ and $\lambda(x)^2 \geq m\|\Delta x_{\rm nt}\|_2^2$. We integrate the inequality to get

$$\begin{array}{rcl} \tilde{f}'(t) & \leq & \tilde{f}'(0) + t\lambda(x)^2 + t^2 \dfrac{L}{2m^{3/2}}\lambda(x)^3 \\ & = & -\lambda(x)^2 + t\lambda(x)^2 + t^2 \dfrac{L}{2m^{3/2}}\lambda(x)^3, \end{array}$$

using $\tilde{f}'(0) = -\lambda(x)^2$. We integrate once more to get

$$\tilde{f}(t) \leq \tilde{f}(0) - t\lambda(x)^2 + t^2\frac{1}{2}\lambda(x)^2 + t^3\frac{L}{6m^{3/2}}\lambda(x)^3.$$

Finally, we take $t = 1$ to obtain

$$f(x + \Delta x_{\rm nt}) \leq f(x) - \frac{1}{2}\lambda(x)^2 + \frac{L}{6m^{3/2}}\lambda(x)^3. \qquad (9.39)$$

Now suppose $\|\nabla f(x)\|_2 \leq \eta \leq 3(1-2\alpha)m^2/L$. By strong convexity, we have

$$\lambda(x) \leq 3(1-2\alpha)m^{3/2}/L,$$

and by (9.39) we have

$$\begin{array}{rcl} f(x + \Delta x_{\rm nt}) & \leq & f(x) - \lambda(x)^2\left(\dfrac{1}{2} - \dfrac{L\lambda(x)}{6m^{3/2}}\right) \\ & \leq & f(x) - \alpha\lambda(x)^2 \\ & = & f(x) + \alpha\nabla f(x)^T\Delta x_{\rm nt}, \end{array}$$

which shows that the unit step $t = 1$ is accepted by the backtracking line search.

Let us now examine the rate of convergence. Applying the Lipschitz condition, we have

$$\begin{array}{rcl} \|\nabla f(x^+)\|_2 & = & \|\nabla f(x+\Delta x_{\rm nt}) - \nabla f(x) - \nabla^2 f(x)\Delta x_{\rm nt}\|_2 \\ & = & \left\|\displaystyle\int_0^1 \left(\nabla^2 f(x + t\Delta x_{\rm nt}) - \nabla^2 f(x)\right)\Delta x_{\rm nt}\; dt\right\|_2 \\ & \leq & \dfrac{L}{2}\|\Delta x_{\rm nt}\|_2^2 \\ & = & \dfrac{L}{2}\|\nabla^2 f(x)^{-1}\nabla f(x)\|_2^2 \\ & \leq & \dfrac{L}{2m^2}\|\nabla f(x)\|_2^2, \end{array}$$

i.e., the inequality (9.33).

In conclusion, the algorithm selects unit steps and satisfies the condition (9.33) if $\|\nabla f(x^{(k)})\|_2 < \eta$, where

$$\eta = \min\{1, 3(1-2\alpha)\}\frac{m^2}{L}.$$

Substituting this bound and (9.38) into (9.37), we find that the number of iterations is bounded above by

$$6 + \frac{M^2L^2/m^5}{\alpha\beta\min\{1, 9(1-2\alpha)^2\}}(f(x^{(0)}) - p^\star). \qquad (9.40)$$

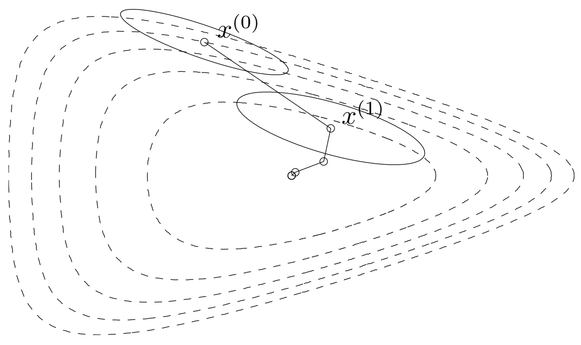

Figure 9.19 Newton's method for the problem in $\mathbf{R}^2$, with objective f given in (9.20), and backtracking line search parameters $\alpha = 0.1$, $\beta = 0.7$. Also shown are the ellipsoids $\{x \mid \|x-x^{(k)}\|_{\nabla^2 f(x^{(k)})} \leq 1\}$ at the first two iterates.

9.5.4 Examples

Example in $\mathbf{R}^2$

We first apply Newton's method with backtracking line search on the test function (9.20), with line search parameters $\alpha = 0.1$, $\beta = 0.7$. Figure 9.19 shows the Newton iterates, and also the ellipsoids

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

for the first two iterates $k = 0,\ 1$. The method works well because these ellipsoids give good approximations of the shape of the sublevel sets.

Figure 9.20 shows the error versus iteration number for the same example. This plot shows that convergence to a very high accuracy is achieved in only five iterations. Quadratic convergence is clearly apparent: The last step reduces the error from about 10^{-5} to 10^{-10}.

Example in $\mathbf{R}^{100}$

Figure 9.21 shows the convergence of Newton's method with backtracking and exact line search for a problem in $\mathbf{R}^{100}$. The objective function has the form (9.21), with the same problem data and the same starting point as was used in figure 9.6. The plot for the backtracking line search shows that a very high accuracy is attained in eight iterations. Like the example in $\mathbf{R}^2$, quadratic convergence is clearly evident after about the third iteration. The number of iterations in Newton's method with exact line search is only one smaller than with a backtracking line search. This is also typical. An exact line search usually gives a very small improvement in convergence of Newton's method. Figure 9.22 shows the step sizes for this example. After two damped steps, the steps taken by the backtracking line search are all full, *i.e.*, $t = 1$.

Experiments with the values of the backtracking parameters α and β reveal that they have little effect on the performance of Newton's method, for this example

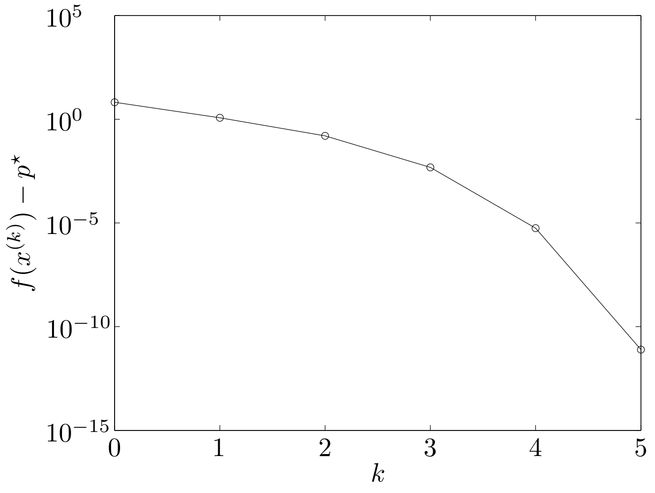

Figure 9.20 Error versus iteration k of Newton's method for the problem in $\mathbf{R}^2$. Convergence to a very high accuracy is achieved in five iterations.

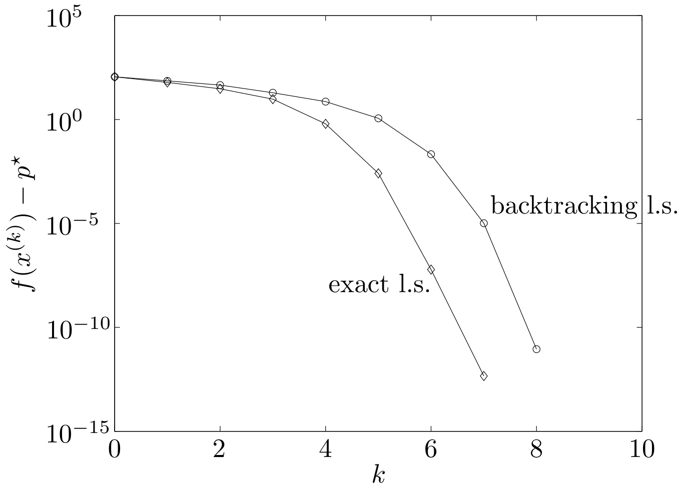

Figure 9.21 Error versus iteration for Newton's method for the problem in $\mathbf{R}^{100}$. The backtracking line search parameters are $\alpha = 0.01$, $\beta = 0.5$. Here too convergence is extremely rapid: a very high accuracy is attained in only seven or eight iterations. The convergence of Newton's method with exact line search is only one iteration faster than with backtracking line search.

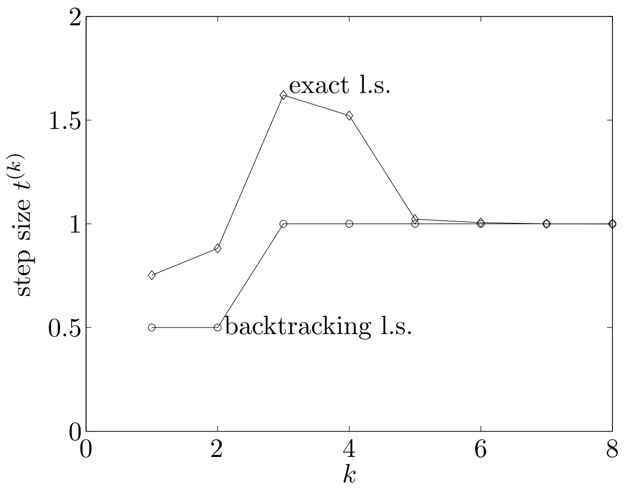

Figure 9.22 The step size t versus iteration for Newton's method with backtracking and exact line search, applied to the problem in $\mathbf{R}^{100}$. The backtracking line search takes one backtracking step in the first two iterations. After the first two iterations it always selects $t = 1$.

(and others). With α fixed at 0.01, and values of β varying between 0.2 and 1, the number of iterations required varies between 8 and 12. With β fixed at 0.5, the number of iterations is 8, for all values of α between 0.005 and 0.5. For these reasons, most practical implementations use a backtracking line search with a small value of α, such as 0.01, and a larger value of β, such as 0.5.

Example in $\mathbf{R}^{10000}$

In this last example we consider a larger problem, of the form

$$\text{minimize} \;\; -\sum_{i=1}^{n} \log(1 - x_i^2) - \sum_{i=1}^{m} \log(b_i - a_i^T x)$$

with $m = 100000$ and $n = 10000$. The problem data a_i are randomly generated sparse vectors. Figure 9.23 shows the convergence of Newton's method with backtracking line search, with parameters $\alpha = 0.01$, $\beta = 0.5$. The performance is very similar to the previous convergence plots. A linearly convergent initial phase of about 13 iterations is followed by a quadratically convergent phase, that achieves a very high accuracy in 4 or 5 more iterations.

Affine invariance of Newton's method

A very important feature of Newton's method is that it is independent of linear (or affine) changes of coordinates. Let $x^{(k)}$ be the kth iterate of Newton's method, applied to $f : \mathbf{R}^n \to \mathbf{R}$. Suppose $T \in \mathbf{R}^{n \times n}$ is nonsingular, and define $\bar{f}(y) = f(Ty)$. If we use Newton's method (with the same backtracking parameters) to

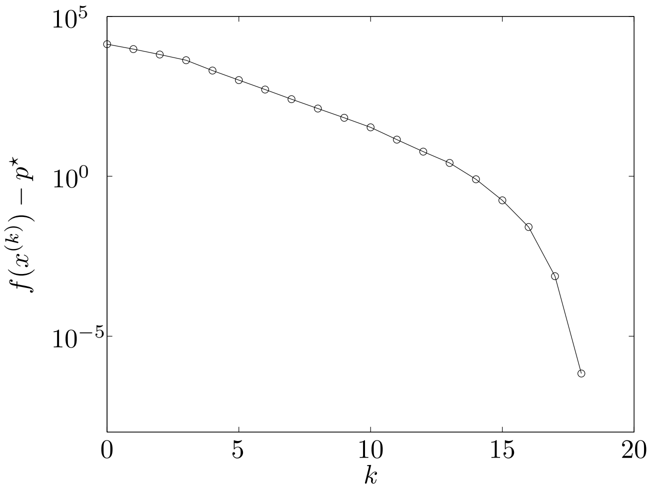

Figure 9.23 Error versus iteration of Newton's method, for a problem in $\mathbf{R}^{10000}$. A backtracking line search with parameters $\alpha = 0.01$, $\beta = 0.5$ is used. Even for this large scale problem, Newton's method requires only 18 iterations to achieve very high accuracy.

minimize $\bar{f}$, starting from $y^{(0)} = T^{-1}x^{(0)}$, then we have

$$Ty^{(k)} = x^{(k)}$$

for all k. In other words, Newton's method is the same: The iterates are related by the same change of coordinates. Even the stopping criterion is the same, since the Newton decrement for $\bar{f}$ at $y^{(k)}$ is the same as the Newton decrement for f at $x^{(k)}$. This is in stark contrast to the gradient (or steepest descent) method, which is strongly affected by changes of coordinates.

As an example, consider the family of problems given in (9.22), indexed by the parameter γ, which affects the condition number of the sublevel sets. We observed (in figures 9.7 and 9.8) that the gradient method slows to useless for values of γ smaller than 0.05 or larger than 20. In contrast, Newton's method (with $\alpha = 0.01$, $\beta = 0.5$) solves this problem (in fact, to a far higher accuracy) in nine iterations, for all values of γ between 10^{-10} and 10^{10}.

In a real implementation, with finite precision arithmetic, Newton's method is not exactly independent of affine changes of coordinates, or the condition number of the sublevel sets. But we can say that condition numbers ranging up to very large values such as 10^{10} do not adversely affect a real implementation of Newton's method. For the gradient method, a far smaller range of condition numbers can be tolerated. While choice of coordinates (or condition number of sublevel sets) is a first-order issue for gradient and steepest descent methods, it is a second-order issue for Newton's method; its only effect is in the numerical linear algebra required to compute the Newton step.

Summary

Newton's method has several very strong advantages over gradient and steepest descent methods:

- Convergence of Newton's method is rapid in general, and quadratic near $x^\star$. Once the quadratic convergence phase is reached, at most six or so iterations are required to produce a solution of very high accuracy.

- Newton's method is affine invariant. It is insensitive to the choice of coordinates, or the condition number of the sublevel sets of the objective.

- Newton's method scales well with problem size. Its performance on problems in $\mathbf{R}^{10000}$ is similar to its performance on problems in $\mathbf{R}^{10}$, with only a modest increase in the number of steps required.

- The good performance of Newton's method is not dependent on the choice of algorithm parameters. In contrast, the choice of norm for steepest descent plays a critical role in its performance.

The main disadvantage of Newton's method is the cost of forming and storing the Hessian, and the cost of computing the Newton step, which requires solving a set of linear equations. We will see in §9.7 that in many cases it is possible to exploit problem structure to substantially reduce the cost of computing the Newton step.

Another alternative is provided by a family of algorithms for unconstrained optimization called *quasi-Newton methods*. These methods require less computational effort to form the search direction, but they share some of the strong advantages of Newton methods, such as rapid convergence near $x^\star$. Since quasi-Newton methods are described in many books, and tangential to our main theme, we will not consider them in this book.

9.6 Self-concordance

There are two major shortcomings of the classical convergence analysis of Newton's method given in §9.5.3. The first is a practical one: The resulting complexity estimates involve the three constants m, M, and L, which are almost never known in practice. As a result, the bound (9.40) on the number of Newton steps required is almost never known specifically, since it depends on three constants that are, in general, not known. Of course the convergence analysis and complexity estimate are still conceptually useful.

The second shortcoming is that while Newton's method is affinely invariant, the classical analysis of Newton's method is very much dependent on the coordinate system used. If we change coordinates the constants m, M, and L all change. If for no reason other than aesthetic, we should seek an analysis of Newton's method that is, like the method itself, independent of affine changes of coordinates. In other words, we seek an alternative to the assumptions

$$mI \preceq \nabla^2 f(x) \preceq MI, \qquad \|\nabla^2 f(x) - \nabla^2 f(y)\|_2 \le L\|x-y\|_2,$$

that is independent of affine changes of coordinates, and also allows us to analyze Newton's method.

A simple and elegant assumption that achieves this goal was discovered by Nesterov and Nemirovski, who gave the name *self-concordance* to their condition. Self-concordant functions are important for several reasons.

- They include many of the logarithmic barrier functions that play an important role in interior-point methods for solving convex optimization problems.
- The analysis of Newton's method for self-concordant functions does not depend on any unknown constants.
- Self-concordance is an affine-invariant property, *i.e.*, if we apply a linear transformation of variables to a self-concordant function, we obtain a self-concordant function. Therefore the complexity estimate that we obtain for Newton's method applied to a self-concordant function is independent of affine changes of coordinates.

9.6.1 Definition and examples

Self-concordant functions on $\mathbf{R}$

We start by considering functions on $\mathbf{R}$. A convex function $f : \mathbf{R} \to \mathbf{R}$ is *self-concordant* if

$$|f'''(x)| \le 2f''(x)^{3/2} \tag{9.41}$$

for all $x \in \mathbf{dom}\, f$. Since linear and (convex) quadratic functions have zero third derivative, they are evidently self-concordant. Some more interesting examples are given below.

Example 9.3 *Logarithm and entropy.*

- *Negative logarithm.* The function $f(x) = -\log x$ is self-concordant. Using $f''(x) = 1/x^2$, $f'''(x) = -2/x^3$, we find that

$$\frac{|f'''(x)|}{2f''(x)^{3/2}} = \frac{2/x^3}{2(1/x^2)^{3/2}} = 1,$$

so the defining inequality (9.41) holds with equality.

- *Negative entropy plus negative logarithm.* The function $f(x) = x\log x - \log x$ is self-concordant. To verify this, we use

$$f''(x) = \frac{x+1}{x^2}, \qquad f'''(x) = -\frac{x+2}{x^3}$$

to obtain

$$\frac{|f'''(x)|}{2f''(x)^{3/2}} = \frac{x+2}{2(x+1)^{3/2}}.$$

The function on the righthand side is maximized on $\mathbf{R}_+$ by $x = 0$, where its value is 1.

The negative entropy function by itself is *not* self-concordant; see exercise 11.13.

We should make two important remarks about the self-concordance definition (9.41). The first concerns the mysterious constant 2 that appears in the definition. In fact, this constant is chosen for convenience, in order to simplify the formulas later on; any other positive constant could be used instead. Suppose, for example, that the convex function $f : \mathbf{R} \to \mathbf{R}$ satisfies

$$|f'''(x)| \leq k f''(x)^{3/2} \tag{9.42}$$

where k is some positive constant. Then the function $\tilde{f}(x) = (k^2/4)f(x)$ satisfies

$$\begin{array}{rcl} |\tilde{f}'''(x)| & = & (k^2/4)|f'''(x)| \\ & \leq & (k^3/4)f''(x)^{3/2} \\ & = & (k^3/4)\left((4/k^2)\tilde{f}''(x)\right)^{3/2} \\ & = & 2\tilde{f}''(x)^{3/2} \end{array}$$

and therefore is self-concordant. This shows that a function that satisfies (9.42) for some positive k can be scaled to satisfy the standard self-concordance inequality (9.41). So what is important is that the third derivative of the function is bounded by some multiple of the 3/2-power of its second derivative. By appropriately scaling the function, we can change the multiple to the constant 2.

The second comment is a simple calculation that shows why self-concordance is so important: it is affine invariant. Suppose we define the function $\tilde{f}$ by $\tilde{f}(y) = f(ay + b)$, where $a \neq 0$. Then $\tilde{f}$ is self-concordant if and only if f is. To see this, we substitute

$$\tilde{f}''(y) = a^2 f''(x), \qquad \tilde{f}'''(y) = a^3 f'''(x),$$

where $x = ay + b$, into the self-concordance inequality for $\tilde{f}$, *i.e.*, $|\tilde{f}'''(y)| \leq 2\tilde{f}''(y)^{3/2}$, to obtain

$$|a^3 f'''(x)| \leq 2(a^2 f''(x))^{3/2},$$

which (after dividing by a^3) is the self-concordance inequality for f. Roughly speaking, the self-concordance condition (9.41) is a way to limit the third derivative of a function, in a way that is independent of affine coordinate changes.

Self-concordant functions on $\mathbf{R}^n$

We now consider functions on $\mathbf{R}^n$ with $n > 1$. We say a function $f : \mathbf{R}^n \to \mathbf{R}$ is self-concordant if it is self-concordant along every line in its domain, *i.e.*, if the function $\tilde{f}(t) = f(x + tv)$ is a self-concordant function of t for all $x \in \mathbf{dom}\, f$ and for all v.

9.6.2 Self-concordant calculus

Scaling and sum

Self-concordance is preserved by scaling by a factor exceeding one: If f is self-concordant and $a \geq 1$, then af is self-concordant. Self-concordance is also preserved by addition: If f_1, f_2 are self-concordant, then $f_1 + f_2$ is self-concordant. To show this, it is sufficient to consider functions $f_1, \ f_2 : \mathbf{R} \to \mathbf{R}$. We have

$$\begin{array}{rcl} |f_1'''(x) + f_2'''(x)| & \leq & |f_1'''(x)| + |f_2'''(x)| \\ & \leq & 2(f_1''(x)^{3/2} + f_2''(x)^{3/2}) \\ & \leq & 2(f_1''(x) + f_2''(x))^{3/2}. \end{array}$$

In the last step we use the inequality

$$(u^{3/2} + v^{3/2})^{2/3} \leq u + v,$$

which holds for $u, \ v \geq 0$.

Composition with affine function

If $f : \mathbf{R}^n \to \mathbf{R}$ is self-concordant, and $A \in \mathbf{R}^{n \times m}$, $b \in \mathbf{R}^n$, then $f(Ax + b)$ is self-concordant.

Example 9.4 *Log barrier for linear inequalities.* The function

$$f(x) = -\sum_{i=1}^{m} \log(b_i - a_i^T x),$$

with $\mathbf{dom}\, f = \{x \mid a_i^T x < b_i, \ i = 1, \ldots, m\}$, is self-concordant. Each term $-\log(b_i - a_i^T x)$ is the composition of $-\log y$ with the affine transformation $y = b_i - a_i^T x$, and hence self-concordant. Therefore the sum is also self-concordant.

Example 9.5 *Log-determinant.* The function $f(X) = -\log \det X$ is self-concordant on $\mathbf{dom}\, f = \mathbf{S}^n_{++}$. To show this, we consider the function $\tilde{f}(t) = f(X + tV)$, where $X \succ 0$ and $V \in \mathbf{S}^n$. It can be expressed as

$$\begin{array}{rcl} \tilde{f}(t) & = & -\log \det(X^{1/2}(I + tX^{-1/2}VX^{-1/2})X^{1/2}) \\ & = & -\log \det X - \log \det(I + tX^{-1/2}VX^{-1/2}) \\ & = & -\log \det X - \displaystyle\sum_{i=1}^{n} \log(1 + t\lambda_i) \end{array}$$

where λ_i are the eigenvalues of $X^{-1/2}VX^{-1/2}$. Each term $-\log(1 + t\lambda_i)$ is a self-concordant function of t, so the sum, $\tilde{f}$, is self-concordant. It follows that f is self-concordant.

Example 9.6 *Log of concave quadratic.* The function

$$f(x) = -\log(x^T Px + q^T x + r),$$

where $P \in -\mathbf{S}_+^n$, is self-concordant on

$$\mathbf{dom}\, f = \{x \mid x^T P x + q^T x + r > 0\}.$$

To show this, it suffices to consider the case $n = 1$ (since by restricting f to a line, the general case reduces to the $n = 1$ case). We can then express f as

$$f(x) = -\log(px^2 + qx + r) = -\log\left(-p(x-a)(b-x)\right)$$

where $\mathbf{dom}\, f = (a, b)$ (*i.e.*, a and b are the roots of px^2+qx+r). Using this expression we have

$$f(x) = -\log(-p) - \log(x-a) - \log(b-x),$$

which establishes self-concordance.

Composition with logarithm

Let $g : \mathbf{R} \to \mathbf{R}$ be a convex function with $\mathbf{dom}\, g = \mathbf{R}_{++}$, and

$$|g'''(x)| \leq 3\frac{g''(x)}{x} \tag{9.43}$$

for all x. Then

$$f(x) = -\log(-g(x)) - \log x$$

is self-concordant on $\{x \mid x > 0,\ g(x) < 0\}$. (For a proof, see exercise 9.14.)

The condition (9.43) is homogeneous and preserved under addition. It is satisfied by all (convex) quadratic functions, *i.e.*, functions of the form $ax^2 + bx + c$, where $a \geq 0$. Therefore if (9.43) holds for a function g, then it holds for the function $g(x) + ax^2 + bx + c$, where $a \geq 0$.

Example 9.7 The following functions g satisfy the condition (9.43).

- $g(x) = -x^p$ for $0 < p \leq 1$.
- $g(x) = -\log x$.
- $g(x) = x \log x$.
- $g(x) = x^p$ for $-1 \leq p \leq 0$.
- $g(x) = (ax+b)^2/x$.

It follows that in each case, the function $f(x) = -\log(-g(x)) - \log x$ is self-concordant. More generally, the function $f(x) = -\log(-g(x) - ax^2 - bx - c) - \log x$ is self-concordant on its domain,

$$\{x \mid x > 0,\ g(x) + ax^2 + bx + c < 0\},$$

provided $a \geq 0$.

Example 9.8 The composition with logarithm rule allows us to show self-concordance of the following functions.

- $f(x, y) = -\log(y^2 - x^T x)$ on $\{(x, y) \mid \|x\|_2 < y\}$.
- $f(x, y) = -2\log y - \log(y^{2/p} - x^2)$, with $p \geq 1$, on $\{(x, y) \in \mathbf{R}^2 \mid |x|^p < y\}$.
- $f(x, y) = -\log y - \log(\log y - x)$ on $\{(x, y) \mid e^x < y\}$.

We leave the details as an exercise (exercise 9.15).

9.6.3 Properties of self-concordant functions

In §9.1.2 we used strong convexity to derive bounds on the suboptimality of a point x in terms of the norm of the gradient at x. For strictly convex self-concordant functions, we can obtain similar bounds in terms of the Newton decrement

$$\lambda(x) = \left(\nabla f(x)^T \nabla^2 f(x)^{-1} \nabla f(x)\right)^{1/2}.$$

(It can be shown that the Hessian of a strictly convex self-concordant function is positive definite everywhere; see exercise 9.17.) Unlike the bounds based on the norm of the gradient, the bounds based on the Newton decrement are not affected by an affine change of coordinates.

For future reference we note that the Newton decrement can also be expressed as

$$\lambda(x) = \sup_{v \neq 0} \frac{-v^T \nabla f(x)}{(v^T \nabla^2 f(x) v)^{1/2}}$$

(see exercise 9.9). In other words, we have

$$\frac{-v^T \nabla f(x)}{(v^T \nabla^2 f(x) v)^{1/2}} \leq \lambda(x) \qquad (9.44)$$

for any nonzero v, with equality for $v = \Delta x_{\mathrm{nt}}$.

Upper and lower bounds on second derivatives

Suppose $f : \mathbf{R} \to \mathbf{R}$ is a strictly convex self-concordant function. We can write the self-concordance inequality (9.41) as

$$\left| \frac{d}{dt} \left(f''(t)^{-1/2} \right) \right| \leq 1 \qquad (9.45)$$

for all $t \in \mathbf{dom}\, f$ (see exercise 9.16). Assuming $t \geq 0$ and the interval between 0 and t is in $\mathbf{dom}\, f$, we can integrate (9.45) between 0 and t to obtain

$$-t \leq \int_0^t \frac{d}{d\tau} \left(f''(\tau)^{-1/2} \right) d\tau \leq t,$$

i.e., $-t \leq f''(t)^{-1/2} - f''(0)^{-1/2} \leq t$. From this we obtain lower and upper bounds on $f''(t)$:

$$\frac{f''(0)}{\left(1 + t f''(0)^{1/2}\right)^2} \leq f''(t) \leq \frac{f''(0)}{\left(1 - t f''(0)^{1/2}\right)^2}. \qquad (9.46)$$

The lower bound is valid for all nonnegative $t \in \mathbf{dom}\, f$; the upper bound is valid if $t \in \mathbf{dom}\, f$ and $0 \leq t < f''(0)^{-1/2}$.

Bound on suboptimality

Let $f : \mathbf{R}^n \to \mathbf{R}$ be a strictly convex self-concordant function, and let v be a descent direction (*i.e.*, any direction satisfying $v^T \nabla f(x) < 0$, not necessarily the

Newton direction). Define $\tilde{f} : \mathbf{R} \to \mathbf{R}$ as $\tilde{f}(t) = f(x + tv)$. By definition, the function $\tilde{f}$ is self-concordant.

Integrating the lower bound in (9.46) yields a lower bound on $\tilde{f}'(t)$:

$$\tilde{f}'(t) \geq \tilde{f}'(0) + \tilde{f}''(0)^{1/2} - \frac{\tilde{f}''(0)^{1/2}}{1 + t\tilde{f}''(0)^{1/2}}. \tag{9.47}$$

Integrating again yields a lower bound on $\tilde{f}(t)$:

$$\tilde{f}(t) \geq \tilde{f}(0) + t\tilde{f}'(0) + t\tilde{f}''(0)^{1/2} - \log(1 + t\tilde{f}''(0)^{1/2}). \tag{9.48}$$

The righthand side reaches its minimum at

$$\bar{t} = \frac{-\tilde{f}'(0)}{\tilde{f}''(0) + \tilde{f}''(0)^{1/2}\tilde{f}'(0)},$$

and evaluating at $\bar{t}$ provides a lower bound on $\tilde{f}$:

$$\begin{array}{rcl} \inf_{t \geq 0} \tilde{f}(t) & \geq & \tilde{f}(0) + \bar{t}\tilde{f}'(0) + \bar{t}\tilde{f}''(0)^{1/2} - \log(1 + \bar{t}\tilde{f}''(0)^{1/2}) \\ & = & \tilde{f}(0) - \tilde{f}'(0)\tilde{f}''(0)^{-1/2} + \log(1 + \tilde{f}'(0)\tilde{f}''(0)^{-1/2}). \end{array}$$

The inequality (9.44) can be expressed as

$$\lambda(x) \geq -\tilde{f}'(0)\tilde{f}''(0)^{-1/2}$$

(with equality when $v = \Delta x_{\rm nt}$), since we have

$$\tilde{f}'(0) = v^T \nabla f(x), \qquad \tilde{f}''(0) = v^T \nabla^2 f(x) v.$$

Now using the fact that $u + \log(1 - u)$ is a monotonically decreasing function of u, and the inequality above, we get

$$\inf_{t \geq 0} \tilde{f}(t) \geq \tilde{f}(0) + \lambda(x) + \log(1 - \lambda(x)).$$

This inequality holds for any descent direction v. Therefore

$$p^\star \geq f(x) + \lambda(x) + \log(1 - \lambda(x)) \tag{9.49}$$

provided $\lambda(x) < 1$. The function $-(\lambda + \log(1 - \lambda))$ is plotted in figure 9.24. It satisfies

$$-(\lambda + \log(1 - \lambda)) \approx \lambda^2/2,$$

for small λ, and the bound

$$-(\lambda + \log(1 - \lambda)) \leq \lambda^2$$

for $\lambda \leq 0.68$. Thus, we have the bound on suboptimality

$$p^\star \geq f(x) - \lambda(x)^2, \tag{9.50}$$

valid for $\lambda(x) \leq 0.68$.

Recall that $\lambda(x)^2/2$ is the estimate of $f(x) - p^\star$, based on the quadratic model at x; the inequality (9.50) shows that for self-concordant functions, doubling this estimate gives us a provable bound. In particular, it shows that for self-concordant functions, we can use the stopping criterion

$$\lambda(x)^2 \leq \epsilon,$$

(where $\epsilon < 0.68^2$), and guarantee that on exit $f(x) - p^\star \leq \epsilon$.

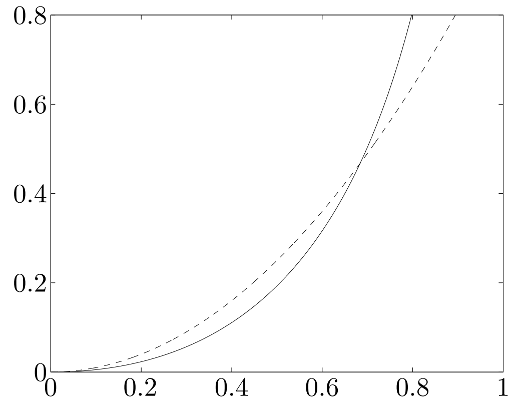

Figure 9.24 The solid line is the function $-(\lambda + \log(1-\lambda))$, which for small λ is approximately $\lambda^2/2$. The dashed line shows λ^2, which is an upper bound in the interval $0 \le \lambda \le 0.68$.

9.6.4 Analysis of Newton's method for self-concordant functions

We now analyze Newton's method with backtracking line search, when applied to a strictly convex self-concordant function f. As before, we assume that a starting point $x^{(0)}$ is known, and that the sublevel set $S = \{x \mid f(x) \le f(x^{(0)})\}$ is closed. We also assume that f is bounded below. (This implies that f has a minimizer $x^\star$; see exercise 9.19.)

The analysis is very similar to the classical analysis given in §9.5.2, except that we use self-concordance as the basic assumption instead of strong convexity and the Lipschitz condition on the Hessian, and the Newton decrement will play the role of the norm of the gradient. We will show that there are numbers η and $\gamma > 0$, with $0 < \eta \le 1/4$, that depend only on the line search parameters α and β, such that the following hold:

- If $\lambda(x^{(k)}) > \eta$, then
$$f(x^{(k+1)}) - f(x^{(k)}) \le -\gamma. \tag{9.51}$$
- If $\lambda(x^{(k)}) \le \eta$, then the backtracking line search selects $t = 1$ and
$$2\lambda(x^{(k+1)}) \le \left(2\lambda(x^{(k)})\right)^2. \tag{9.52}$$

These are the analogs of (9.32) and (9.33). As in §9.5.3, the second condition can be applied recursively, so we can conclude that for all $l \ge k$, we have $\lambda(x^{(l)}) \le \eta$, and
$$2\lambda(x^{(l)}) \le \left(2\lambda(x^{(k)})\right)^{2^{l-k}} \le (2\eta)^{2^{l-k}} \le \left(\frac{1}{2}\right)^{2^{l-k}}.$$
As a consequence, for all $l \ge k$,
$$f(x^{(l)}) - p^\star \le \lambda(x^{(l)})^2 \le \frac{1}{4}\left(\frac{1}{2}\right)^{2^{l-k+1}} \le \left(\frac{1}{2}\right)^{2^{l-k+1}},$$

and hence $f(x^{(l)}) - p^\star \leq \epsilon$ if $l - k \geq \log_2 \log_2(1/\epsilon)$.

The first inequality implies that the damped phase cannot require more than $(f(x^{(0)}) - p^\star)/\gamma$ steps. Thus the total number of iterations required to obtain an accuracy $f(x) - p^\star \leq \epsilon$, starting at a point $x^{(0)}$, is bounded by

$$\frac{f(x^{(0)}) - p^\star}{\gamma} + \log_2 \log_2(1/\epsilon). \tag{9.53}$$

This is the analog of the bound (9.36) in the classical analysis of Newton's method.

Damped Newton phase

Let $\tilde{f}(t) = f(x + t\Delta x_{\rm nt})$, so we have

$$\tilde{f}'(0) = -\lambda(x)^2, \qquad \tilde{f}''(0) = \lambda(x)^2.$$

If we integrate the upper bound in (9.46) twice, we obtain an upper bound for $\tilde{f}(t)$:

$$\begin{aligned} \tilde{f}(t) &\leq \tilde{f}(0) + t\tilde{f}'(0) - t\tilde{f}''(0)^{1/2} - \log\left(1 - t\tilde{f}''(0)^{1/2}\right) \\ &= \tilde{f}(0) - t\lambda(x)^2 - t\lambda(x) - \log(1 - t\lambda(x)), \end{aligned} \tag{9.54}$$

valid for $0 \leq t < 1/\lambda(x)$.

We can use this bound to show the backtracking line search always results in a step size $t \geq \beta/(1+\lambda(x))$. To prove this we note that the point $\hat{t} = 1/(1+\lambda(x))$ satisfies the exit condition of the line search:

$$\begin{aligned} \tilde{f}(\hat{t}) &\leq \tilde{f}(0) - \hat{t}\lambda(x)^2 - \hat{t}\lambda(x) - \log(1 - \hat{t}\lambda(x)) \\ &= \tilde{f}(0) - \lambda(x) + \log(1 + \lambda(x)) \\ &\leq \tilde{f}(0) - \alpha\frac{\lambda(x)^2}{1+\lambda(x)} \\ &= \tilde{f}(0) - \alpha\lambda(x)^2\hat{t}. \end{aligned}$$

The second inequality follows from the fact that

$$-x + \log(1+x) + \frac{x^2}{2(1+x)} \leq 0$$

for $x \geq 0$. Since $t \geq \beta/(1+\lambda(x))$, we have

$$\tilde{f}(t) - \tilde{f}(0) \leq -\alpha\beta\frac{\lambda(x)^2}{1+\lambda(x)},$$

so (9.51) holds with

$$\gamma = \alpha\beta\frac{\eta^2}{1+\eta}.$$

Quadratically convergent phase

We will show that we can take

$$\eta = (1-2\alpha)/4,$$

(which satisfies $0 < \eta < 1/4$, since $0 < \alpha < 1/2$), *i.e.*, if $\lambda(x^{(k)}) \leq (1-2\alpha)/4$, then the backtracking line search accepts the unit step and (9.52) holds.

We first note that the upper bound (9.54) implies that a unit step $t = 1$ yields a point in $\mathbf{dom}\, f$ if $\lambda(x) < 1$. Moreover, if $\lambda(x) \leq (1-2\alpha)/2$, we have, using (9.54),

$$\begin{array}{rcl} \tilde{f}(1) & \leq & \tilde{f}(0) - \lambda(x)^2 - \lambda(x) - \log(1-\lambda(x)) \\ & \leq & \tilde{f}(0) - \dfrac{1}{2}\lambda(x)^2 + \lambda(x)^3 \\ & \leq & \tilde{f}(0) - \alpha\lambda(x)^2, \end{array}$$

so the unit step satisfies the condition of sufficient decrease. (The second line follows from the fact that $-x - \log(1-x) \leq \frac{1}{2}x^2 + x^3$ for $0 \leq x \leq 0.81$.)

The inequality (9.52) follows from the following fact, proved in exercise 9.18. If $\lambda(x) < 1$, and $x^+ = x - \nabla^2 f(x)^{-1} \nabla f(x)$, then

$$\lambda(x^+) \leq \frac{\lambda(x)^2}{(1-\lambda(x))^2}. \tag{9.55}$$

In particular, if $\lambda(x) \leq 1/4$,

$$\lambda(x^+) \leq 2\lambda(x)^2,$$

which proves that (9.52) holds when $\lambda(x^{(k)}) \leq \eta$.

The final complexity bound

Putting it all together, the bound (9.53) on the number of Newton iterations becomes

$$\frac{f(x^{(0)}) - p^\star}{\gamma} + \log_2 \log_2(1/\epsilon) = \frac{20-8\alpha}{\alpha\beta(1-2\alpha)^2}(f(x^{(0)}) - p^\star) + \log_2\log_2(1/\epsilon). \tag{9.56}$$

This expression depends only on the line search parameters α and β, and the final accuracy ϵ. Moreover the term involving ϵ can be safely replaced by the constant six, so the bound really depends only on α and β. For typical values of α and β, the constant that scales $f(x^{(0)}) - p^\star$ is on the order of several hundred. For example, with $\alpha = 0.1$, $\beta = 0.8$, the scaling factor is 375. With tolerance $\epsilon = 10^{-10}$, we obtain the bound

$$375(f(x^{(0)}) - p^\star) + 6. \tag{9.57}$$

We will see that this bound is fairly conservative, but does capture what appears to be the general form of the worst-case number of Newton steps required. A more refined analysis, such as the one originally given by Nesterov and Nemirovski, gives a similar bound, with a substantially smaller constant scaling $f(x^{(0)}) - p^\star$.

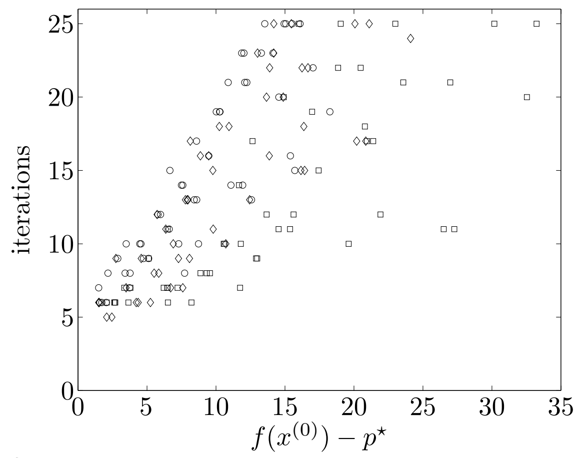

Figure 9.25 Number of Newton iterations required to minimize self-concordant functions versus $f(x^{(0)}) - p^\star$. The function f has the form $f(x) = -\sum_{i=1}^m \log(b_i - a_i^T x)$, where the problem data a_i and b are randomly generated. The circles show problems with $m = 100$, $n = 50$; the squares show problems with $m = 1000$, $n = 500$; and the diamonds show problems with $m = 1000$, $n = 50$. Fifty instances of each are shown.

9.6.5 Discussion and numerical examples

A family of self-concordant functions

It is interesting to compare the upper bound (9.57) with the actual number of iterations required to minimize a self-concordant function. We consider a family of problems of the form

$$f(x) = -\sum_{i=1}^m \log(b_i - a_i^T x).$$

The problem data a_i and b were generated as follows. For each problem instance, the coefficients of a_i were generated from independent normal distributions with mean zero and unit variance, and the coefficients b were generated from a uniform distribution on $[0, 1]$. Problem instances which were unbounded below were discarded. For each problem we first compute $x^\star$. We then generate a starting point by choosing a random direction v, and taking $x^{(0)} = x^\star + sv$, where s is chosen so that $f(x^{(0)}) - p^\star$ has a prescribed value between 0 and 35. (We should point out that starting points with values $f(x^{(0)}) - p^\star = 10$ or higher are actually very close to the boundary of the polyhedron.) We then minimize the function using Newton's method with a backtracking line search with parameters $\alpha = 0.1$, $\beta = 0.8$, and tolerance $\epsilon = 10^{-10}$.

Figure 9.25 shows the number of Newton iterations required versus $f(x^{(0)}) - p^\star$ for 150 problem instances. The circles show 50 problems with $m = 100$, $n = 50$; the squares show 50 problems with $m = 1000$, $n = 500$; and the diamonds show 50 problems with $m = 1000$, $n = 50$.

For the values of the backtracking parameters used, the complexity bound found above is

$$375(f(x^{(0)}) - p^\star) + 6, \qquad (9.58)$$

clearly a much larger value than the number of iterations required (for these 150 instances). The plot suggests that there is a valid bound of the same form, but with a much smaller constant (say, around 1.5) scaling $f(x^{(0)}) - p^\star$. Indeed, the expression

$$f(x^{(0)}) - p^\star + 6$$

is not a bad gross predictor of the number of Newton steps required, although it is clearly not the only factor. First, there are plenty of problems instances where the number of Newton steps is somewhat smaller, which correspond, we can guess, to 'lucky' starting points. Note also that for the larger problems, with 500 variables (represented by the squares), there seem to be even more cases where the number of Newton steps is unusually small.

We should mention here that the problem family we study is not just self-concordant, but in fact minimally self-concordant, by which we mean that αf is not self-concordant for $\alpha < 1$. Hence, the bound (9.58) cannot be improved by simply scaling f. (The function $f(x) = -20 \log x$ is an example of a self-concordant function which is not minimally self-concordant, since $(1/20)f$ is also self-concordant.)

Practical importance of self-concordance

We have already observed that Newton's method works in general very well for strongly convex objective functions. We can justify this vague statement empirically, and also using the classical analysis of Newton's method, which yields a complexity bound, but one that depends on several constants that are almost always unknown.

For self-concordant functions we can say somewhat more. We have a complexity bound that is completely explicit, and does not depend on any unknown constants. Empirical studies suggest that this bound can be tightened considerably, but its general form, a small constant plus a multiple of $f(x^{(0)}) - p^\star$, seems to predict, at least crudely, the number of Newton steps required to minimize an approximately minimally self-concordant function.

It is not yet clear whether self-concordant functions are in practice more easily minimized by Newton's method than non-self-concordant functions. (It is not even clear how one would make this statement precise.) At the moment, we can say that self-concordant functions are a class of functions for which we can say considerably more about the complexity of Newton's method than is the case for non-self-concordant functions.

9.7 Implementation

In this section we discuss some of the issues that arise in implementing an unconstrained minimization algorithm. We refer the reader to appendix C for more details on numerical linear algebra.

9.7.1 Pre-computation for line searches

In the simplest implementation of a line search, $f(x + t\Delta x)$ is evaluated for each value of t in the same way that $f(z)$ is evaluated for any $z \in \mathbf{dom}\, f$. But in some cases we can exploit the fact that f (and its derivatives, in an exact line search) are to be evaluated at many points along the ray $\{x + t\Delta x \mid t \geq 0\}$ to reduce the total computational effort. This usually requires some pre-computation, which is often on the same order as computing f at any point, after which f (and its derivatives) can be computed more efficiently along the ray.

Suppose that $x \in \mathbf{dom}\, f$ and $\Delta x \in \mathbf{R}^n$, and define $\tilde{f}$ as f restricted to the line or ray determined by x and Δx, *i.e.*, $\tilde{f}(t) = f(x + t\Delta x)$. In a backtracking line search we must evaluate $\tilde{f}$ for several, and possibly many, values of t; in an exact line search method we must evaluate $\tilde{f}$ and one or more derivatives at a number of values of t. In the simple method described above, we evaluate $\tilde{f}(t)$ by first forming $z = x + t\Delta x$, and then evaluating $f(z)$. To evaluate $\tilde{f}'(t)$, we form $z = x + t\Delta x$, then evaluate $\nabla f(z)$, and then compute $\tilde{f}'(t) = \nabla f(z)^T \Delta x$. In some representative examples below we show how $\tilde{f}$ can be computed at a number of values of t more efficiently.

Composition with an affine function

A very general case in which pre-computation can speed up the line search process occurs when the objective has the form $f(x) = \phi(Ax + b)$, where $A \in \mathbf{R}^{p\times n}$, and ϕ is easy to evaluate (for example, separable). To evaluate $\tilde{f}(t) = f(x + t\Delta x)$ for k values of t using the simple approach, we form $A(x + t\Delta x) + b$ for each value of t (which costs $2kpn$ flops), and then evaluate $\phi(A(x + t\Delta x) + b)$ for each value of t. This can be done more efficiently by first computing $Ax + b$ and $A\Delta x$ ($4pn$ flops), then forming $A(x + t\Delta x) + b$ for each value of t using

$$A(x + t\Delta x) + b = (Ax + b) + t(A\Delta x),$$

which costs $2kp$ flops. The total cost, keeping only the dominant terms, is $4pn+2kp$ flops, compared to $2kpn$ for the simple method.

Analytic center of a linear matrix inequality

Here we give an example that is more specific, and more complete. We consider the problem (9.6) of computing the analytic center of a linear matrix inequality, *i.e.*, minimizing $\log\det F(x)^{-1}$, where $x \in \mathbf{R}^n$ and $F : \mathbf{R}^n \to \mathbf{S}^p$ is affine. Along the line through x with direction Δx we have

$$\tilde{f}(t) = \log\det(F(x + t\Delta x))^{-1} = -\log\det(A + tB)$$

where

$$A = F(x), \qquad B = \Delta x_1 F_1 + \cdots + \Delta x_n F_n \in \mathbf{S}^p.$$

Since $A \succ 0$, it has a Cholesky factorization $A = LL^T$, where L is lower triangular and nonsingular. Therefore we can express $\tilde{f}$ as

$$\tilde{f}(t) = -\log\det\left(L(I + tL^{-1}BL^{-T})L^T\right) = -\log\det A - \sum_{i=1}^{p} \log(1 + t\lambda_i) \qquad (9.59)$$

where $\lambda_1, \ldots, \lambda_p$ are the eigenvalues of $L^{-1}BL^{-T}$. Once these eigenvalues are computed, we can evaluate $\tilde{f}(t)$, for any t, with $4p$ simple arithmetic computations, by using the formula on the right hand side of (9.59). We can evaluate $\tilde{f}'(t)$ (and similarly, any higher derivative) in $4p$ operations, using the formula

$$\tilde{f}'(t) = -\sum_{i=1}^{p} \frac{\lambda_i}{1 + t\lambda_i}.$$

Let us compare the two methods for carrying out a line search, assuming that we need to evaluate $f(x + t\Delta x)$ for k values of t. In the simple method, for each value of t we form $F(x+t\Delta x)$, and then evaluate $f(x+t\Delta x)$ as $-\log\det F(x+t\Delta x)$. For example, we can find the Cholesky factorization of $F(x + t\Delta x) = LL^T$, and then evaluate

$$-\log\det F(x + t\Delta x) = -2\sum_{i=1}^{p} \log L_{ii}.$$

The cost is np^2 to form $F(x + t\Delta x)$, plus $(1/3)p^3$ for the Cholesky factorization. Therefore the total cost of the line search is

$$k(np^2 + (1/3)p^3) = knp^2 + (1/3)kp^3.$$

Using the method outlined above, we first form A, which costs np^2, and factor it, which costs $(1/3)p^3$. We also form B (which costs np^2), and $L^{-1}BL^{-T}$, which costs $2p^3$. The eigenvalues of this matrix are then computed, at a cost of about $(4/3)p^3$ flops. This pre-computation requires a total of $2np^2 + (11/3)p^3$ flops. After finishing this pre-computation, we can now evaluate $\tilde{f}(t)$ for each value of t at a cost of $4p$ flops. The total cost is then

$$2np^2 + (11/3)p^3 + 4kp.$$

Assuming k is small compared to $p(2n + (11/3)p)$, this means the entire line search can be carried out at an effort comparable to simply evaluating f. Depending on the values of k, p, and n, the savings over the simple method can be as large as order k.

9.7.2 Computing the Newton step

In this section we briefly describe some of the issues that arise in implementing Newton's method. In most cases, the work of computing the Newton step Δx_{nt}

dominates the work involved in the line search. To compute the Newton step $\Delta x_{\rm nt}$, we first evaluate and form the Hessian matrix $H = \nabla^2 f(x)$ and the gradient $g = \nabla f(x)$ at x. Then we solve the system of linear equations $H\Delta x_{\rm nt} = -g$ to find the Newton step. This set of equations is sometimes called the *Newton system* (since its solution gives the Newton step) or the *normal equations*, since the same type of equation arises in solving a least-squares problem (see §9.1.1).

While a general linear equation solver can be used, it is better to use methods that take advantage of the symmetry and positive definiteness of H. The most common approach is to form the Cholesky factorization of H, *i.e.*, to compute a lower triangular matrix L that satisfies $LL^T = H$ (see §C.3.2). We then solve $Lw = -g$ by forward substitution, to obtain $w = -L^{-1}g$, and then solve $L^T\Delta x_{\rm nt} = w$ by back substitution, to obtain

$$\Delta x_{\rm nt} = L^{-T}w = -L^{-T}L^{-1}g = -H^{-1}g.$$

We can compute the Newton decrement as $\lambda^2 = -\Delta x_{\rm nt}^T g$, or use the formula

$$\lambda^2 = g^T H^{-1} g = \|L^{-1}g\|_2^2 = \|w\|_2^2.$$

If a dense (unstructured) Cholesky factorization is used, the cost of the forward and back substitution is dominated by the cost of the Cholesky factorization, which is $(1/3)n^3$ flops. The total cost of computing the Newton step $\Delta x_{\rm nt}$ is thus $F+(1/3)n^3$ flops, where F is the cost of forming H and g.

It is often possible to solve the Newton system $H\Delta x_{\rm nt} = -g$ more efficiently, by exploiting special structure in H, such as band structure or sparsity. In this context, 'structure of H' means structure that is the same for all x. For example, when we say that 'H is tridiagonal' we mean that for every $x \in \mathbf{dom}\, f$, $\nabla^2 f(x)$ is tridiagonal.

Band structure

If the Hessian H is banded with bandwidth k, *i.e.*, $H_{ij} = 0$ for $|i-j| > k$, then the banded Cholesky factorization can be used, as well as banded forward and back substitutions. The cost of computing the Newton step $\Delta x_{\rm nt} = -H^{-1}g$ is then $F + nk^2$ flops (assuming $k \ll n$), compared to $F + (1/3)n^3$ for a dense factorization and substitution method.

The Hessian band structure condition

$$\nabla^2 f(x)_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j} = 0 \quad \text{for} \quad |i-j| > k,$$

for all $x \in \mathbf{dom}\, f$, has an interesting interpretation in terms of the objective function f. Roughly speaking it means that in the objective function, each variable x_i couples nonlinearly only to the $2k+1$ variables x_j, $j = i-k, \ldots, i+k$. This occurs when f has the partial separability form

$$f(x) = \psi_1(x_1, \ldots, x_{k+1}) + \psi_2(x_2, \ldots, x_{k+2}) + \cdots + \psi_{n-k}(x_{n-k}, \ldots, x_n),$$

where $\psi_i : \mathbf{R}^{k+1} \to \mathbf{R}$. In other words, f can be expressed as a sum of functions of k consecutive variables.

Example 9.9 Consider the problem of minimizing $f : \mathbf{R}^n \to \mathbf{R}$, which has the form

$$f(x) = \psi_1(x_1, x_2) + \psi_2(x_2, x_3) + \cdots + \psi_{n-1}(x_{n-1}, x_n),$$

where $\psi_i : \mathbf{R}^2 \to \mathbf{R}$ are convex and twice differentiable. Because of this form, the Hessian $\nabla^2 f$ is tridiagonal, since $\partial^2 f/\partial x_i \partial x_j = 0$ for $|i - j| > 1$. (And conversely, if the Hessian of a function is tridiagonal for all x, then it has this form.)

Using Cholesky factorization and forward and back substitution algorithms for tridiagonal matrices, we can solve the Newton system for this problem in order n flops. This should be compared to order n^3 flops, if the special form of f were not exploited.

Sparse structure

More generally we can exploit sparsity of the Hessian H in solving the Newton system. This sparse structure occurs whenever each variable x_i is nonlinearly coupled (in the objective) to only a few other variables, or equivalently, when the objective function can be expressed as a sum of functions, each depending on only a few variables, and each variable appearing in only a few of these functions.

To solve $H\Delta x = -g$ when H is sparse, a sparse Cholesky factorization is used to compute a permutation matrix P and lower triangular matrix L for which

$$H = PLL^T P^T.$$

The cost of this factorization depends on the particular sparsity pattern, but is often far smaller than $(1/3)n^3$, and an empirical complexity of order n (for large n) is not uncommon. The forward and back substitution are very similar to the basic method without the permutation. We solve $Lw = -P^T g$ using forward substitution, and then solve $L^T v = w$ by back substitution to obtain

$$v = L^{-T} w = -L^{-T} L^{-1} P^T g.$$

The Newton step is then $\Delta x = Pv$.

Since the sparsity pattern of H does not change as x varies (or more precisely, since we only exploit sparsity that does not change with x) we can use the same permutation matrix P for each of the Newton steps. The step of determining a good permutation matrix P, which is called the *symbolic factorization* step, can be done once, for the whole Newton process.

Diagonal plus low rank

There are many other types of structure that can be exploited in solving the Newton system $H\Delta x_{\text{nt}} = -g$. Here we briefly describe one, and refer the reader to appendix C for more details. Suppose the Hessian H can be expressed as a diagonal matrix plus one of low rank, say, p. This occurs when the objective function f has the special form

$$f(x) = \sum_{i=1}^{n} \psi_i(x_i) + \psi_0(Ax + b) \qquad (9.60)$$

where $A \in \mathbf{R}^{p \times n}$, $\psi_1, \ldots, \psi_n : \mathbf{R} \to \mathbf{R}$, and $\psi_0 : \mathbf{R}^p \to \mathbf{R}$. In other words, f is a separable function, plus a function that depends on a low dimensional affine function of x.

To find the Newton step Δx_{nt} for (9.60) we must solve the Newton system $H\Delta x_{\mathrm{nt}} = -g$, with

$$H = D + A^T H_0 A.$$

Here $D = \mathbf{diag}(\psi_1''(x_1), \ldots, \psi_n''(x_n))$ is diagonal, and $H_0 = \nabla^2 \psi_0(Ax + b)$ is the Hessian of ψ_0. If we compute the Newton step without exploiting the structure, the cost of solving the Newton system is $(1/3)n^3$ flops.

Let $H_0 = L_0 L_0^T$ be the Cholesky factorization of H_0. We introduce the temporary variable $w = L_0^T A \Delta x_{\mathrm{nt}} \in \mathbf{R}^p$, and express the Newton system as

$$D\Delta x_{\mathrm{nt}} + A^T L_0 w = -g, \qquad w = L_0^T A \Delta x_{\mathrm{nt}}.$$

Substituting $\Delta x_{\mathrm{nt}} = -D^{-1}(A^T L_0 w + g)$ (from the first equation) into the second equation, we obtain

$$(I + L_0^T A D^{-1} A^T L_0) w = -L_0^T A D^{-1} g, \tag{9.61}$$

which is a system of p linear equations.

Now we proceed as follows to compute the Newton step Δx_{nt}. First we compute the Cholesky factorization of H_0, which costs $(1/3)p^3$. We then form the dense, positive definite symmetric matrix appearing on the lefthand side of (9.61), which costs $2p^2 n$. We then solve (9.61) for w using a Cholesky factorization and a back and forward substitution, which costs $(1/3)p^3$ flops. Finally, we compute Δx_{nt} using $\Delta x_{\mathrm{nt}} = -D^{-1}(A^T L_0 w + g)$, which costs $2np$ flops. The total cost of computing Δx_{nt} is (keeping only the dominant term) $2p^2 n$ flops, which is far smaller than $(1/3)n^3$ for $p \ll n$.

Bibliography

Dennis and Schnabel [DS96] and Ortega and Rheinboldt [OR00] are two standard references on algorithms for unconstrained minimization and nonlinear equations. The result on quadratic convergence, assuming strong convexity and Lipschitz continuity of the Hessian, is attributed to Kantorovich [Kan52]. Polyak [Pol87, §1.6] gives some insightful comments on the role of convergence results that involve unknown constants, such as the results derived in §9.5.3.

Self-concordant functions were introduced by Nesterov and Nemirovski [NN94]. All our results in §9.6 and exercises 9.14–9.20 can be found in their book, although often in a more general form or with different notation. Renegar [Ren01] gives a concise and elegant presentation of self-concordant functions and their role in the analysis of primal-dual interior-point algorithms. Peng, Roos, and Terlaky [PRT02] study interior-point methods from the viewpoint of *self-regular functions*, a class of functions that is similar, but not identical, to self-concordant functions.

References for the material in §9.7 are given at the end of appendix C.