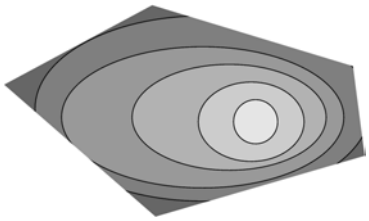

Figure 6.4 A convex function on a convex polyhedral set. Note that the minimum of this function is unique, and that the maximum can be found at one of the vertices of the constraining domain.

Reconstructing Convex Sets from Vertices

The proof is slightly technical, and not central to the understanding of kernel methods. See Rockafellar [435, Chapter 18] for details, along with further theorems on convex functions. We now proceed to the second key theorem in this section.

Theorem 6.12 (Maxima of Convex Functions on Convex Compact Sets) *Denote by X a compact convex set in $\mathcal{X}$, by $|X$ the vertices of X, and by f a convex function on X. Then*

$$\sup\{f(x)|x \in X\} = \sup\{f(x)|x \in |X\}. \tag{6.10}$$

Proof Application of Theorem 6.10 and Theorem 6.11 proves the claim, since under the assumptions made on X, we have $X = \mathrm{co}(|X)$. Figure 6.4 depicts the situation graphically. ■

6.2 Unconstrained Problems

After the characterization and uniqueness results (Theorem 6.5, Corollary 6.6, and Lemma 6.7) of the previous section, we will now study numerical techniques to obtain minima (or maxima) of convex optimization problems. While the choice of algorithms is motivated by applicability to kernel methods, the presentation here is not problem specific. For details on implementation, and descriptions of applications to learning problems, see Chapter 10.

6.2.1 Functions of One Variable

We begin with the easiest case, in which f depends on only one variable. Some of the concepts explained here, such as the interval cutting algorithm and Newton's method, can be extended to the multivariate setting (see Problem 6.5). For the sake of simplicity, however, we limit ourselves to the univariate case.

Assume we want to minimize $f : \mathbb{R} \to \mathbb{R}$ on the interval $[a, b] \subset \mathbb{R}$. If we cannot make any further assumptions regarding f, then this problem, as simple as it may seem, cannot be solved numerically.

Continuous Differentiable Functions

If f is *differentiable*, the problem can be reduced to finding $f'(x) = 0$ (see Problem 6.4 for the general case). If in addition to the previous assumptions, f is convex, then f' is nondecreasing, and we can find a fast, simple algorithm (Algorithm

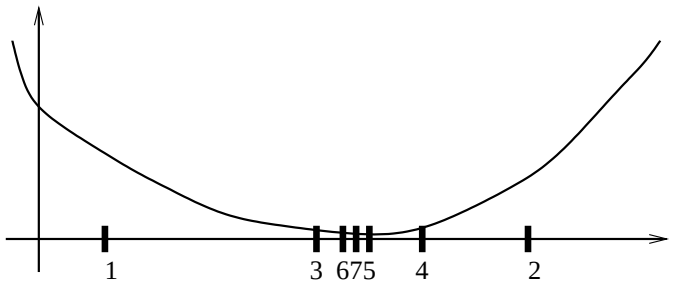

Figure 6.5 Interval Cutting Algorithm. The selection of points is ordered according to the numbers beneath (points 1 and 2 are the initial endpoints of the interval).

Algorithm 6.1 Interval Cutting

Require: a, b, Precision ϵ
  Set $A = a, B = b$
  repeat
    if $f'\left(\frac{A+B}{2}\right) > 0$ **then**
      $B = \frac{A+B}{2}$
    else
      $A = \frac{A+B}{2}$
    end if
  until $(B - A)\min(|f'(A)|, |f'(B)|) \leq \epsilon$
Output: $x = \frac{A+B}{2}$

Interval Cutting

6.1) to solve our problem (see Figure 6.5).

This technique works by halving the size of the interval that contains the minimum x^* of f, since it is always guaranteed by the selection criteria for B and A that $x^* \in [A, B]$. We use the following Taylor series expansion to determine the stopping criterion.

Theorem 6.13 (Taylor Series) *Denote by $f : \mathbb{R} \to \mathbb{R}$ a function that is d times differentiable. Then for any $x, x' \in \mathbb{R}$, there exists a ξ with $|\xi| \leq |x - x'|$, such that*

$$f(x') = \sum_{i=0}^{d-1} \frac{1}{i!} f^{(i)}(x)(x' - x)^i + \frac{\xi^d}{d!} f^{(d)}(x + \xi). \tag{6.11}$$

Now we may apply (6.11) to the stopping criterion of Algorithm 6.1. We denote by x^* the minimum of $f(x)$. Expanding f around $f(x^*)$, we obtain for some $\xi_A \in [A - x^*, 0]$ that $f(A) = f(x^*) + \xi_A f'(x^* + \xi_A)$, and therefore,

$$|f(A) - f(x^*)| = |\xi_A||f'(x^* + \xi_A)| \leq (B - A)|f'(A)|.$$

Proof of Linear Convergence

Taking the minimum over $\{A, B\}$ shows that Algorithm 6.1 stops once f is ϵ-close to its minimal value. The convergence of the algorithm is *linear* with constant 0.5, since the intervals $[A, B]$ for possible x^* are halved at each iteration.

Algorithm 6.2 Newton's Method

Require: x_0, Precision ϵ
  Set $x = x_0$
  repeat
    $x = x - \frac{f'(x)}{f''(x)}$
  until $|f'(x)| \leq \epsilon$
Output: x

In constructing the interval cutting algorithm, we in fact wasted most of the information obtained in evaluating f' at each point, by only making use of the sign of f. In particular, we could fit a parabola to f and thereby obtain a method that converges more rapidly. If we are only allowed to use f and f', this leads to the *Method of False Position* (see [334] or Problem 6.3).

Moreover, if we may compute the second derivative as well, we can use (6.11) to obtain a quadratic approximation of f and use the latter to find the minimum of f.

Newton's Method

This is commonly referred to as *Newton's method* (see Section 16.4.1 for a practical application of the latter to classification problems). We expand $f(x)$ around x_0;

$$f(x) \approx f(x_0) + (x - x_0)f'(x_0) + \frac{(x - x_0)^2}{2} f''(x_0). \tag{6.12}$$

Minimization of the expansion (6.12) yields

$$x = x_0 - \frac{f'(x_0)}{f''(x_0)}. \tag{6.13}$$

Hence, we hope that if the approximation (6.12) is good, we will obtain an algorithm with fast convergence (Algorithm 6.2). Let us analyze the situation in more detail. For convenience, we state the result in terms of $g := f'$, since finding a zero of g is equivalent to finding a minimum of f.

Theorem 6.14 (Convergence of Newton Method) *Let $g : \mathbb{R} \to \mathbb{R}$ be a twice continuously differentiable function, and denote by $x^* \in \mathbb{R}$ a point with $g'(x^*) \neq 0$ and $g(x^*) = 0$. Then, provided x_0 is sufficiently close to x^*, the sequence generated by (6.13) will converge to x^* at least quadratically.*

Quadratic Convergence

Proof For convenience, denote by x_n the value of x at the nth iteration. As before, we apply Theorem 6.13. We now expand $g(x^*)$ around x_n. For some $\xi \in [0, x^* - x_n]$, we have

$$g(x_n) = g(x_n) - g(x^*) = g(x_n) - \left[g(x_n) + g'(x_n)(x^* - x_n) + \frac{\xi^2}{2} g''(x_n) \right], \tag{6.14}$$

and therefore by substituting (6.14) into (6.13),

$$x_{n+1} - x^* = x_n - x^* - \frac{g(x_n)}{g'(x_n)} = \xi^2 \frac{g''(x_n)}{2g'(x_n)}. \tag{6.15}$$

Since by construction $|\xi| \leq |x_n - x^*|$, we obtain a quadratically convergent algorithm in $|x_n - x^*|$, provided that $\left| (x_n - x^*) \frac{g''(x_n)}{2g'(x_n)} \right| < 1$. ∎

Region of Convergence

In other words, if the Newton method converges, it converges more rapidly than interval cutting or similar methods. We cannot guarantee beforehand that we are really in the region of convergence of the algorithm. In practice, if we apply the Newton method and find that it converges, we know that the solution has converged to the minimizer of f. For more information on optimization algorithms for unconstrained problems see [173, 530, 334, 15, 159, 45].

Line Search

In some cases we will not know an upper bound on the size of the interval to be analyzed for the presence of minima. In this situation we may, for instance, start with an initial guess of an interval, and if no minimum can be found strictly *inside* the interval, enlarge it, say by doubling its size. See [334] for more information on this matter. Let us now proceed to a technique which is quite popular (albeit not always preferable) in machine learning.

6.2.2 Functions of Several Variables: Gradient Descent

Gradient descent is one of the simplest optimization techniques to implement for minimizing functions of the form $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X}$ may be $\mathbb{R}^N$, or indeed any set on which a gradient may be defined and evaluated. In order to avoid further complications we assume that the gradient $f'(x)$ exists and that we are able to compute it.

Direction of Steepest Descent

The basic idea is as follows: given a location x_n at iteration n, compute the gradient $g_n := f'(x_n)$, and update

$$x_{n+1} = x_n - \gamma g_n \tag{6.16}$$

such that the decrease in f is maximal over all $\gamma > 0$. For the final step, one of the algorithms from Section 6.2.1 can be used. It is straightforward to show that $f(x_n)$ is a monotonically decreasing series, since at each step the line search updates x_{n+1} in such a way that $f(x_{n+1}) < f(x_n)$. Such a value of γ must exist, since (again by Theorem 6.13) we may expand $f(x_n + \gamma g_n)$ in terms of γ around x_n, to obtain[1]

$$f(x_n - \gamma g_n) = f(x_n) - \gamma \|g_n\|^2 + O(\gamma^2). \tag{6.17}$$

As usual $\|\cdot\|$ is the Euclidean norm. For small γ the linear contribution in the Taylor expansion will be dominant, hence for some $\gamma > 0$ we have $f(x_n - \gamma g_n) < f(x_n)$. It can be shown [334] that after a (possibly infinite) number of steps, gradient descent (see Algorithm 6.3) will converge.

Problems of Convergence

In spite of this, the performance of gradient descent is far from optimal. Depending on the shape of the landscape of values of f, gradient descent may take a long time to converge. Figure 6.6 shows two examples of possible convergence behavior of the gradient descent algorithm.

1. To see that Theorem 6.13 applies in (6.17), note that $f(x_n + \gamma g_n)$ is a mapping $\mathbb{R} \to \mathbb{R}$ when viewed as a function of γ.

Algorithm 6.3 Gradient Descent

Require: x_0, Precision ϵ
  $n = 0$
  repeat
    Compute $g = f'(x_n)$
    Perform line search on $f(x_n - \gamma g)$ for optimal γ.
    $x_{n+1} = x_n - \gamma g$
    $n = n + 1$
  until $\|f'(x_n)\| \leq \epsilon$
Output: x_n

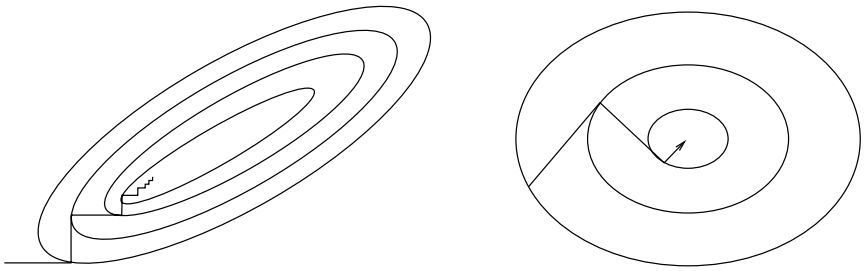

Figure 6.6 Left: Gradient descent takes a long time to converge, since the landscape of values of f forms a long and narrow valley, causing the algorithm to zig-zag along the walls of the valley. Right: due to the homogeneous structure of the minimum, the algorithm converges after very few iterations. Note that in both cases, the next direction of descent is *orthogonal* to the previous one, since line search provides the optimal step length.

6.2.3 Convergence Properties of Gradient Descent

Let us analyze the convergence properties of Algorithm 6.3 in more detail. To keep matters simple, we assume that f is a quadratic function, i.e.

$$f(x) = \frac{1}{2}(x - x^*)^\top K(x - x^*) + c_0, \tag{6.18}$$

where K is a positive definite symmetric matrix (cf. Definition 2.4) and c_0 is constant.[2] This is clearly a convex function with minimum at x^*, and $f(x^*) = c_0$. The gradient of f is given by

$$g := f'(x) = K(x - x^*). \tag{6.19}$$

To find the update of the steepest descent we have to minimize

$$f(x - \gamma g) = \frac{1}{2}(x - \gamma g - x^*)K(x - \gamma g - x^*) = \frac{1}{2}\gamma^2 g^\top K g - \gamma g^\top g. \tag{6.20}$$

2. Note that we may rewrite (up to a constant) any convex quadratic function $f(x) = x^\top K x + c^\top x + d$ in the form (6.18), simply by expanding f around its minimum value x^*.

By minimizing (6.20) for γ, the update of steepest descent is given explicitly by

$$x_{n+1} = x_n - \frac{g^\top g}{g^\top K g} g. \tag{6.21}$$

Improvement per Step

Substituting (6.21) into (6.18) and subtracting the terms $f(x_n)$ and $f(x_{n+1})$ yields the following improvement after an update step

$$\begin{aligned} f(x_n) - f(x_{n+1}) &= (x_n - x^*)^\top K \frac{g^\top g}{g^\top K g} g - \frac{1}{2}\left(\frac{g^\top g}{g^\top K g}\right)^2 g^\top K g \\ &= \frac{1}{2}\frac{(g^\top g)^2}{g^\top K g} = f(x_n)\left[\frac{(g^\top g)^2}{(g^\top K g)(g^\top K^{-1} g)}\right]. \end{aligned} \tag{6.22}$$

Thus the relative improvement per iteration depends on the value of $t(g) := \frac{(g^\top g)^2}{(g^\top K g)(g^\top K^{-1} g)}$. In order to give performance guarantees we have to find a lower bound for $t(g)$. To this end we introduce the *condition* of a matrix.

Definition 6.15 (Condition of a Matrix) *Denote by K a matrix and by $\lambda_{\max}$ and $\lambda_{\min}$ its largest and smallest singular values (or eigenvalues if they exist) respectively. The condition of a matrix is defined as*

$$\text{cond } K := \frac{\lambda_{\max}}{\lambda_{\min}}. \tag{6.23}$$

Clearly, as cond K decreases, different directions are treated in a more homogeneous manner by $x^\top K x$. In particular, note that smaller cond K correspond to less elliptic contours in Figure 6.6. Kantorovich proved the following inequality which allows us to connect the condition number with the convergence behavior of gradient descent algorithms.

Theorem 6.16 (Kantorovich Inequality [278]) *Denote by $K \in \mathbb{R}^{m \times m}$ (typically the kernel matrix) a strictly positive definite symmetric matrix with largest and smallest eigenvalues $\lambda_{\max}$ and $\lambda_{\min}$. Then the following inequality holds for any $g \in \mathbb{R}^m$:*

Lower Bound for Improvement

$$\frac{(g^\top g)^2}{(g^\top K g)(g^\top K^{-1} g)} \geq \frac{4\lambda_{\min}\lambda_{\max}}{(\lambda_{\min} + \lambda_{\max})^2} \geq \frac{1}{\text{cond } K}. \tag{6.24}$$

We typically denote by g the gradient of f. The second inequality follows immediately from Definition 6.15; the proof of the first inequality is more technical, and is not essential to the understanding of the situation. See Problem 6.7 and [278, 334] for more detail.

A brief calculation gives us the correct order of magnitude. Note that for any x, the quadratic term $x^\top K x$ is bounded from above by $\lambda_{\max}\|x\|^2$, and likewise $x^\top K^{-1} x \leq \lambda_{\min}^{-1}\|x\|^2$. Hence we bound the relative improvement $t(g)$ (as defined below (6.22)) by $1/(\text{cond } K)$ which is almost as good as the second term in (6.24) (the latter can be up to a factor of 4 better for $\lambda_{\min} \ll \lambda_{\max}$).

This means that gradient descent methods perform poorly if some of the eigenvalues of K are very small in comparison with the largest eigenvalue, as is usually the case with matrices generated by positive definite kernels (and as sometimes

desired for learning theoretical reasons); see Chapter 4 for details. This is one of the reasons why many gradient descent algorithms for training Support Vector Machines, such as the Kernel AdaTron [183, 12] or AdaLine [185], exhibit poor convergence. Section 10.6.1 deals with these issues, and sets up the gradient descent directions both in the Reproducing Kernel Hilbert Space $\mathcal{H}$ and in coefficient space $\mathbb{R}^m$.

6.2.4 Functions of Several Variables: Conjugate Gradient Descent

Let us now look at methods that are better suited to minimizing convex functions. Again, we start with quadratic forms. The key problem with gradient descent is that the quotient between the smallest and the largest eigenvalue can be very large, which leads to slow convergence. Hence, one possible technique is to *rescale* $\mathcal{X}$ by some matrix M such that the condition of $K \in \mathbb{R}^{m\times m}$ in this rescaled space, which is to say the condition of $M^\top KM$, is much closer to 1 (in numerical analysis this is often referred to as *preconditioning* [247, 423, 530]). In addition, we would like to focus first on the largest eigenvectors of K.

A key tool is the concept of *conjugate directions*. The basic idea is that rather than using the metric of the normal dot product $x^\top x' = x^\top \mathbf{1} x'$ ($\mathbf{1}$ is the unit matrix) we use the metric imposed by K, i.e. $x^\top Kx'$, to guide our algorithm, and we introduce an equivalent notion of orthogonality with respect to the new metric.

Definition 6.17 (Conjugate Directions) *Given a symmetric matrix $K \in \mathbb{R}^{m\times m}$, any two vectors $v, v' \in \mathbb{R}^m$ are called K-orthogonal if $v^\top Kv' = 0$.*

Likewise, we can introduce notions of a basis and of linear independence with respect to K. The following theorem establishes the necessary identities.

Theorem 6.18 (Orthogonal Decompositions in K) *Denote by $K \in \mathbb{R}^{m\times m}$ a strictly positive definite symmetric matrix and by $v_1, \ldots, v_m$ a set of mutually K-orthogonal and nonzero vectors. Then the following properties hold:*

(i) The vectors $v_1, \ldots, v_m$ form a basis.

(ii) Any $x \in \mathbb{R}^m$ can be expanded in terms of v_i by

$$x = \sum_{i=1}^{m} v_i \frac{v_i^\top Kx}{v_i^\top Kv_i}. \tag{6.25}$$

In particular, for any $y = Kx$, we can find x by

$$x = \sum_{i=1}^{m} v_i \frac{v_i^\top y}{v_i^\top Kv_i}. \tag{6.26}$$

Linear Independence

Proof (i) Since we have m vectors in $\mathbb{R}^m$, all we have to show is that the vectors v_i are linearly independent. Assume that there exist some $\alpha_i \in \mathbb{R}$ such that $\sum_{i=1}^m \alpha_i v_i =$

0. Then due to K-orthogonality, we have

$$0 = v_j^\top K \left[\sum_{i=1}^m \alpha_i v_i \right] = \sum_{i=1}^m \alpha_i v_j^\top K v_i = \alpha_j v_j^\top K v_j \text{ for all } j. \tag{6.27}$$

Hence $\alpha_j = 0$ for all j. This means that all v_j are linearly independent.

(ii) The vectors $\{v_1, \ldots, v_m\}$ form a basis. Therefore we may expand any $x \in \mathbb{R}^m$ as a linear combination of v_j, i.e. $x = \sum_{i=1}^m \alpha_i v_i$. Consequently we can expand $v_j^\top K x$ in terms of $v_j^\top K v_i$, and we obtain

$$v_j^\top K x = v_j^\top K \left[\sum_{i=1}^m \alpha_i v_i \right] = \alpha_j v_j^\top K v_j. \tag{6.28}$$

Basis Expansion

Solving for α_j proves the claim.

(iii) Let $y = Kx$. Since the vectors v_i form a basis, we can expand x in terms of α_i. Substituting this definition into (6.28) proves (6.26). ∎

The practical consequence of this theorem is that, provided we know a set of K-orthogonal vectors v_i, we can solve the linear equation $y = Kx$ via (6.26). Furthermore, we can also use it to minimize quadratic functions of the form $f(x) = \frac{1}{2} x^\top K x - c^\top x$. The following theorem tells us how.

Theorem 6.19 (Deflation Method) *Denote by $v_1, \ldots, v_m$ a set of mutually K-orthogonal vectors for a strictly positive definite symmetric matrix $K \in \mathbb{R}^{m \times m}$. Then for any $x_0 \in \mathbb{R}^m$ the following method finds x_i that minimize $f(x) = \frac{1}{2} x^\top K x - c^\top x$ in the linear manifold $\mathcal{X}_i := x_0 + \operatorname{span}\{v_1, \ldots, v_i\}$.*

Optimality in Linear Space

$$x_i := x_{i-1} - v_i \frac{g_{i-1}^\top v_i}{v_i^\top K v_i} \text{ where } g_{i-1} = f'(x_{i-1}) \text{ for all } i > 0. \tag{6.29}$$

Proof We use induction. For $i = 0$ the statement is trivial, since the linear manifold consists of only one point.

Assume that the statement holds for i. Since f is convex, we only need prove that the gradient of $f(x_i)$ is orthogonal to $\operatorname{span}\{v_1, \ldots, v_i\}$. In that case no further improvement can be gained on the linear manifold $\mathcal{X}_i$. It suffices to show that for all $j \leq i + 1$,

$$0 = v_j^\top g_i. \tag{6.30}$$

Gradient Descent in Rescaled Space

Additionally, we may expand x_{i+1} to obtain

$$v_j^\top g_i = v_j^\top \left[K x_{i-1} - c - K v_i \frac{g_{i-1}^\top v_i}{v_i^\top K v_i} \right] = v_j^\top g_{i-1} - (g_{i-1}^\top v_i) \frac{v_j^\top K v_i}{v_i^\top K v_i}. \tag{6.31}$$

For $j = i$ both terms cancel out. For $j < i$ both terms vanish due to the induction assumption. Since the vectors v_j form a basis $\mathcal{X}_m = \mathbb{R}^m$, x_m is a minimizer of f. ∎

In a nutshell, Theorem 6.19 already contains the Conjugate Gradient descent al-

Algorithm 6.4 Conjugate Gradient Descent

Require: x_0
  Set $i = 0$
  $g_0 = f'(x_0)$
  $v_0 = g_0$
  repeat
    $x_{i+1} = x_i + \alpha_i v_i$ where $\alpha_i = -\frac{g_i^\top v_i}{v_i^\top K v_i}$
    $g_{i+1} = f'(x_{i+1})$
    $v_{i+1} = -g_{i+1} + \beta_i v_i$ where $\beta_i = \frac{g_{i+1}^\top K v_i}{v_i^\top K v_i}$.
    $i = i + 1$
  until $g_i = 0$
Output: x_i

gorithm: in each step we perform gradient descent with respect to one of the K-orthogonal vectors v_i, which means that after n steps we will reach the minimum. We still lack a method to obtain such a K-orthogonal basis of vectors v_i. It turns out that we can get the latter directly from the gradients g_i. Algorithm 6.4 describes the procedure.

All we have to do is prove that Algorithm 6.4 actually does what it is required to do, namely generate a K-orthogonal set of vectors v_i, and perform deflation in the latter. To achieve this, the v_i are obtained by an orthogonalization procedure akin to Gram-Schmidt orthogonalization.

Theorem 6.20 (Conjugate Gradient) *Assume we are given a quadratic convex function $f(x) = \frac{1}{2}x^\top Kx - c^\top x$, to which we apply conjugate gradient descent for minimization purposes. Then algorithm 6.4 is a deflation method, and unless $g_i = 0$, we have for every $0 \leq i \leq m$,*

(i) $\operatorname{span}\{g_0, \ldots, g_i\} = \operatorname{span}\{v_0, \ldots, v_i\} = \operatorname{span}\{g_0, Kg_0, \ldots, K^i g_0\}$.

(ii) The vectors v_i are K-orthogonal.

(iii) The equations in Algorithm 6.4 for α_i and β_i can be replaced by $\alpha_i = \frac{g_i^\top g_i}{v_i^\top K v_i}$ and $\beta_i = \frac{g_{i+1}^\top g_{i+1}}{g_i^\top g_i}$.

(iv) After i steps, x_i is the solution in the manifold $x_0 + \operatorname{span}\{g_0, Kg_0, \ldots, K^{i-1}g_0\}$.

Proof **(i) and (ii)** We use induction. For $i = 0$ the statements trivially hold since $v_0 = g_0$. For i note that by construction (see Algorithm 6.4) $g_{i+1} = Kx_{i+1} - c = g_i + \alpha_i K v_i$, hence $\operatorname{span}\{g_0, \ldots, g_{i+1}\} = \operatorname{span}\{g_0, Kg_0, \ldots, K^{i+1}g_0\}$. Since $v_{i+1} = -g_{i+1} + \beta_i v_i$ the same statement holds for $\operatorname{span}\{v_0, \ldots, v_{i+1}\}$. Moreover, the vectors g_i are linearly independent or 0 due to Theorem 6.19.
Finally $v_j^\top K v_{i+1} = -v_j^\top K g_{i+1} + \beta_i v_j^\top K v_i = 0$, since for $j = i$ both terms cancel out, and for $j < i$ both terms individually vanish (due to Theorem 6.19 and (i)).

(iii) We have $-g_i^\top v_i = g_i^\top g_i - \beta_{i-1} g_i^\top v_{i-1} = g_i^\top g_i$, since the second term vanishes due to Theorem 6.19. This proves the result for α_i.

Table 6.1 Non-quadratic modifications of conjugate gradient descent.

Generic Method	Compute Hessian $K_i := f''(x_i)$ and update α_i, β_i with $\alpha_i = -\frac{g_i^\top v_i}{v_i^\top K_i v_i}$ $\beta_i = \frac{g_{i+1}^\top K_i v_i}{v_i^\top K_i v_i}$ This requires calculation of the Hessian at each iteration.
Fletcher-Reeves [173]	Find α_i via a line search and use Theorem 6.20 (iii) for β_i $\alpha_i = \operatorname{argmin}_\alpha f(x_i + \alpha v_i)$ $\beta_i = \frac{g_{i+1}^\top g_{i+1}}{g_i^\top g_i}$
Polak-Ribiere [414]	Find α_i via a line search $\alpha_i = \operatorname{argmin}_\alpha f(x_i + \alpha v_i)$ $\beta_i = \frac{(g_{i+1}-g_i)^\top g_{i+1}}{g_i^\top g_i}$ Experimentally, Polak-Ribiere tends to be better than Fletcher-Reeves.

For β_i note that $g_{i+1}^\top K v_i = \alpha_i^{-1} g_{i+1}^\top (g_{i+1} - g_i) = \alpha_i^{-1} g_{i+1}^\top g_{i+1}$. Substitution of the value of α_i proves the claim.

(iv) Again, we use induction. At step $i = 1$ we compute the solution within the space spanned by g_0. ∎

We conclude this section with some remarks on the optimality of conjugate gradient descent algorithms, and how they can be extended to arbitrary convex functions.

Space of Largest Eigenvalues

Due to Theorems 6.19 and 6.20, we can see that after i iterations, the conjugate gradient descent algorithm finds a solution on the linear manifold $x_0 + \operatorname{span}\{g_0, Kg_0, \ldots, K^{i-1}g_0\}$. This means that the solutions will be mostly aligned with the largest eigenvalues of K, since after multiple application of K to any arbitrary vector g_0, the largest eigenvectors dominate. Nonetheless, the algorithm here is significantly cheaper than computing the eigenvalues of K, and subsequently minimizing f in the subspace corresponding to the largest eigenvalues. For more detail see [334]

Nonlinear Extensions

In the case of general convex functions, the assumptions of Theorem 6.20 are no longer satisfied. In spite of this, conjugate gradient descent has proven to be effective even in these situations. Additionally, we have to account for some modifications. Basically, the update rules for g_i and v_i remain unchanged but the parameters α_i and β_i are computed differently. Table 6.1 gives an overview of different methods. See [173, 334, 530, 414] for details.

6.2.5 Predictor Corrector Methods

As we go to higher order Taylor expansions of the function f to be minimized (or set to zero), the corresponding numerical methods become increasingly com-

Increasing the Order

plicated to implement, and require an ever increasing number of parameters to be estimated or computed. For instance, a quadratic expansion of a multivariate function $f : \mathbb{R}^m \to \mathbb{R}$ requires $m \times m$ terms for the quadratic part (the Hessian), whereas the linear part (the gradient) can be obtained by computing m terms. Since the quadratic expansion is only an approximation for most non-quadratic functions, this is wasteful (for interior point programs, see Section 6.4). We might instead be able to achieve roughly the same goal without computing the quadratic term explicitly, or more generally, obtain the performance of higher order methods without actually implementing them.

This can in fact be achieved using predictor-corrector methods. These work by computing a tentative update $x_i \to x_{i+1}^{\text{pred}}$ (predictor step), then using x_{i+1}^{pred} to account for higher order changes in the objective function, and finally obtaining a *corrected* value x_{i+1}^{corr} based on these changes. A simple example illustrates the method. Assume we want to find the solution to the equation

Predictor Corrector Methods for Quadratic Equations

$$f(x) = 0 \text{ where } f(x) = f_0 + ax + \frac{1}{2}bx^2. \tag{6.32}$$

We assume $a, b, f_0, x \in \mathbb{R}$. Exact solution of (6.32) requires taking a square root. Let us see whether we can find an approximate method that avoids this (in general b will be an $m \times m$ matrix, so this is a worthwhile goal). The predictor corrector approach works as follows: first solve

$$f_0 + ax = 0 \text{ and hence } x^{\text{pred}} = -\frac{f_0}{a}. \tag{6.33}$$

Second, substitute x^{pred} into the nonlinear parts of (6.32) to obtain

$$f_0 + ax^{\text{corr}} + \frac{1}{2}b\left(\frac{f_0}{a}\right)^2 = 0 \text{ and hence } x^{\text{corr}} = -\frac{f_0}{a}\left(1 + \frac{1}{2}\frac{bf_0}{a^2}\right). \tag{6.34}$$

No Quadratic Residuals

Comparing x^{pred} and x^{corr}, we see that $\frac{1}{2}\frac{bf_0}{a^2}$ is the correction term that takes the effect of the changes in x into account.

Since neither of the two values (x^{pred} or x^{corr}) will give us the exact solution to $f(x) = 0$ in just one step, it is worthwhile having a look at the errors of both approaches.

$$f(x^{\text{pred}}) = \frac{1}{2}\frac{bf_0^2}{a^2} \text{ and } f(x^{\text{corr}}) = 2\frac{f^2(x^{\text{pred}})}{f_0} + \frac{f^3(x^{\text{pred}})}{f_0^2}. \tag{6.35}$$

We can check that if $\frac{bf_0}{a^2} \leq 2 - 2\sqrt{2}$, the corrector estimate will be better than the predictor one. As our initial estimate f_0 decreases, this will be the case. Moreover, we can see that $f(x^{\text{corr}})$ only contains terms in x that are of higher order than quadratic. This means that even though we did not solve the quadratic form explicitly, we eliminated all corresponding terms.

The general scheme is described in Algorithm 6.5. It is based on the assumption that $f(x + \xi)$ can be split up into

$$f(x + \xi) = f(x) + f_{\text{simple}}(\xi, x) + T(\xi, x), \tag{6.36}$$

Algorithm 6.5 Predictor Corrector Method

Require: x_0, Precision ϵ
  Set $i = 0$
  repeat
    Expand f into $f(x_i) + f_{\text{simple}}(\xi, x_i) + T(\xi, x_i)$.
Predictor Solve $f(x_i) + f_{\text{simple}}(\xi^{\text{pred}}, x_i) = 0$ for ξ^{pred}.
Corrector Solve $f(x_i) + f_{\text{simple}}(\xi^{\text{corr}}, x_i) + T(\xi^{\text{pred}}, x_i) = 0$ for ξ^{corr}.
    $x_{i+1} = x_i + \xi^{\text{corr}}$.
    $i = i + 1$.
  until $|f(x_i)| \leq \epsilon$
Output: x_i

where $f_{\text{simple}}(\xi, x)$ contains the simple, possibly low order, part of f, and $T(\xi, x)$ the higher order terms, such that $f_{\text{simple}}(0, x) = T(0, x) = 0$. While in the previous example we introduced higher order terms into f that were not present before (f is only quadratic), usually such terms will already exist anyway. Hence the corrector step will just eliminate additional lower order terms without too much additional error in the approximation.
We will encounter such methods for instance in the context of interior point algorithms (Section 6.4), where we have to solve a set of quadratic equations.

6.3 Constrained Problems

After this digression on unconstrained optimization problems, let us return to constrained optimization, which makes up the main body of the problems we will have to deal with in learning (e.g., quadratic or general convex programs for Support Vector Machines). Typically, we have to deal with problems of type (6.6). For convenience we repeat the problem statement:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & c_i(x) \leq 0 \text{ for all } i \in [n]. \end{array} \tag{6.37}$$

Here f and c_i are convex functions and $n \in \mathbb{N}$. In some cases[3], we additionally have *equality* constraints $e_j(x) = 0$ for some $j \in [n']$. Then the optimization problem can be written as

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x), \\ \text{subject to} & c_i(x) \leq 0 \text{ for all } i \in [n], \\ & e_j(x) = 0 \text{ for all } j \in [n']. \end{array} \tag{6.38}$$

3. Note that it is common practice in Support Vector Machines to write c_i as positivity constraints by using concave functions. This can be fixed by a sign change, however.