6 Optimization

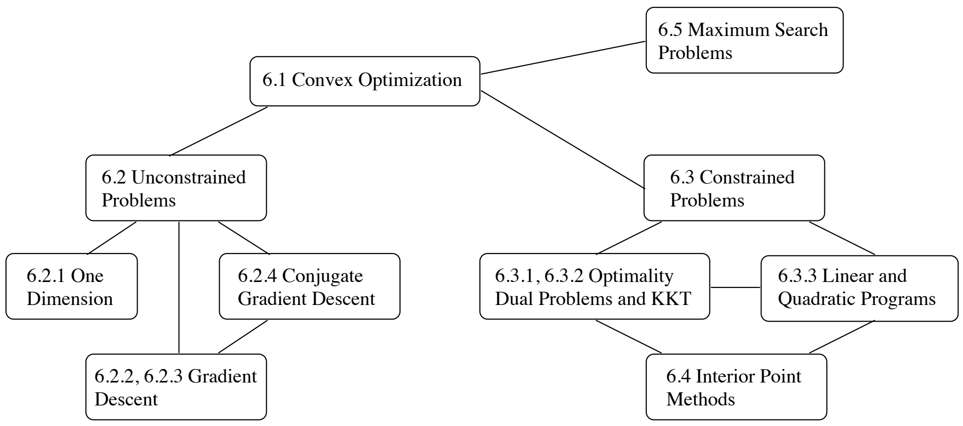
This chapter provides a self-contained overview of some of the basic tools needed to solve the optimization problems used in kernel methods. In particular, we will cover topics such as minimization of functions in one variable, convex minimization and maximization problems, duality theory, and statistical methods to solve optimization problems approximately.

The focus is noticeably different from the topics covered in works on optimization for Neural Networks, such as Backpropagation [575, 435, 298, 9] and its variants. In these cases, it is necessary to deal with non-convex problems exhibiting a large number of local minima, whereas much of the research on Kernel Methods and Mathematical Programming is focused on problems with global exact solutions. These boundaries may become less clear-cut in the future, but at the present time, methods for the solution of problems with unique optima appear to be sufficient for our purposes.

Overview

In Section 6.1, we explain general properties of convex sets and functions, and how the extreme values of such functions can be found. Next, we discuss practical algorithms to best minimize convex functions on unconstrained domains (Section 6.2). In this context, we will present techniques like interval cutting methods, Newton's method, gradient descent and conjugate gradient descent. Section 6.3 then deals with constrained optimization problems, and gives characterization results for optimal solutions. In this context, the Lagrange functions, primal and dual optimization problems, and the Kuhn-Tucker conditions are introduced. These concepts set the stage for Section 6.4, which presents an interior point algorithm for the solution of constrained convex optimization problems. In a sense, the final section (Section 6.5) is a departure from the previous topics, since it introduces the notion of randomization into the optimization procedures. The basic idea is that unless the exact optimal solution is required, statistical tools can speed up search maximization by orders of magnitude.

For a general overview, we recommend Section 6.1, and the first parts of Section 6.3, which explain the basic ideas underlying constrained optimization. The latter section is needed to understand the calculations which lead to the dual optimization problems in Support Vector Machines (Chapters 7 – 9). Section 6.4 is only intended for readers interested in practical implementations of optimization algorithms. In particular, Chapter 10 will require some knowledge of this section. Finally, Section 6.5 describes novel randomization techniques, which are needed in the sparse greedy methods of Section 10.2, 15.3, 16.4, and 18.4.3. Unconstrained optimization


6.1 Convex Optimization
6.5 Maximum Search Problems
6.2 Unconstrained Problems
6.3 Constrained Problems
6.2.1 One Dimension
6.2.4 Conjugate Gradient Descent
6.3.1, 6.3.2 Optimality Dual Problems and KKT
6.3.3 Linear and Quadratic Programs
6.2.2, 6.2.3 Gradient Descent
6.4 Interior Point Methods

problems (Section 6.2) are less common in this book and will only be required in the gradient descent methods of Section 10.6.1, and the Gaussian Process implementation methods of Section 16.4.

Prerequisites

The present chapter is intended as an introduction to the basic concepts of optimization. It is relatively self-contained, and requires only basic skills in linear algebra and multivariate calculus. Section 6.3 is somewhat more technical, Section 6.4 requires some additional knowledge of numerical analysis, and Section 6.5 assumes some knowledge of probability and statistics.

6.1 Convex Optimization

In the situations considered in this book, learning (or equivalently statistical estimation) implies the minimization of some risk functional such as $R_{\text{emp}}[f]$ or $R_{\text{reg}}[f]$ (cf. Chapter 4). While minimizing an arbitrary function on a (possibly not even compact) set of arguments can be a difficult task, and will most likely exhibit many local minima, minimization of a convex objective function on a convex set exhibits exactly one *global* minimum. We now prove this property.

Definition and Construction of Convex Sets and Functions

Definition 6.1 (Convex Set) *A set X in a vector space is called convex if for any $x, x' \in X$ and any $\lambda \in [0,1]$, we have*

$$\lambda x + (1-\lambda)x' \in X. \tag{6.1}$$

Definition 6.2 (Convex Function) *A function f defined on a set X (note that X need not be convex itself) is called convex if, for any $x, x' \in X$ and any $\lambda \in [0,1]$ such that $\lambda x + (1-\lambda)x' \in X$, we have*

$$f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda) f(x'). \tag{6.2}$$

A function f is called strictly *convex if for $x \neq x'$ and $\lambda \in (0,1)$ (6.2) is a strict inequality.*

There exist several ways to define convex sets. A convenient method is to define them via *below sets* of convex functions, such as the sets for which $f(x) \leq c$, for instance.

Lemma 6.3 (Convex Sets as Below-Sets) *Denote by $f : \mathcal{X} \to \mathbb{R}$ a convex function on a convex set $\mathcal{X}$. Then the set*

$$X := \{x | x \in \mathcal{X} \text{ and } f(x) \leq c\}, \text{ for some } c \in \mathbb{R}, \tag{6.3}$$

is convex.

Proof We must show condition (6.1). For any $x, x' \in \mathcal{X}$, we have $f(x), f(x') \leq c$. Moreover, since f is convex, we also have

$$f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda) f(x') \leq c \text{ for all } \lambda \in [0,1]. \tag{6.4}$$

Hence, for all $\lambda \in [0,1]$, we have $(\lambda x + (1-\lambda)x') \in X$, which proves the claim. Figure 6.1 depicts this situation graphically. ■

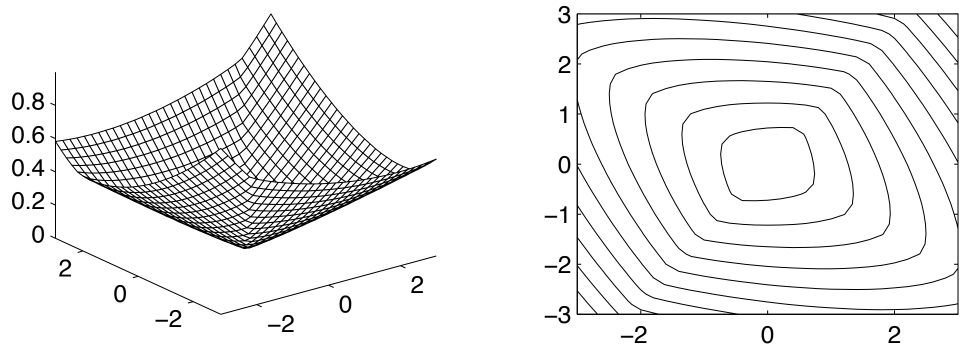


Figure 6.1 Left: Convex Function in two variables. Right: the corresponding convex level sets $\{x | f(x) \leq c\}$, for different values of c.

Intersections

Lemma 6.4 (Intersection of Convex Sets) *Denote by $X, X' \subset \mathcal{X}$ two convex sets. Then $X \cap X'$ is also a convex set.*

Proof Given any $x, x' \in X \cap X'$, then for any $\lambda \in [0,1]$, the point $x_\lambda := \lambda x + (1-\lambda)x'$ satisfies $x_\lambda \in X$ and $x_\lambda \in X'$, hence also $x_\lambda \in X \cap X'$. ■

See also Figure 6.2. Now we have the tools to prove the central theorem of this section.

Theorem 6.5 (Minima on Convex Sets) *If the convex function $f : \mathcal{X} \to \mathbb{R}$ has a minimum on a convex set $X \subset \mathcal{X}$, then its arguments $x \in \mathcal{X}$, for which the minimum value is attained, form a convex set. Moreover, if f is strictly convex, then this set will contain only one element.*

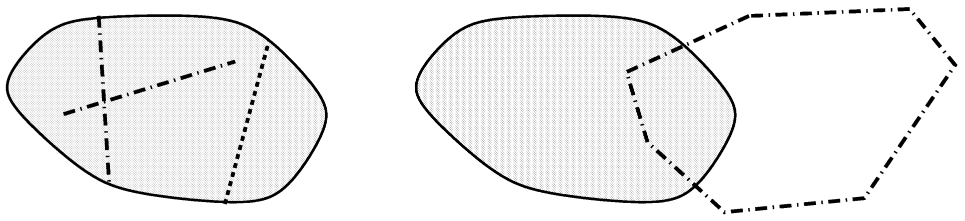


Figure 6.2 Left: a convex set; observe that lines with points in the set are fully contained inside the set. Right: the intersection of two convex sets is also a convex set.

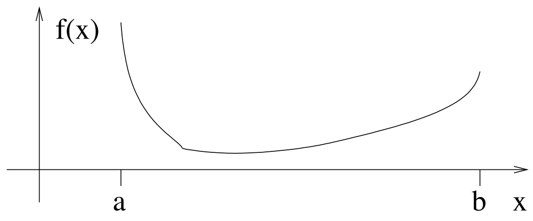


Figure 6.3 Note that the maximum of a convex function is obtained at the ends of the interval $[a, b]$.

Proof Denote by c the minimum of f on X. Then the set $X_m := \{x | x \in \mathcal{X} \text{ and } f(x) \leq c\}$ is clearly convex. In addition, $X_m \cap X$ is also convex, and $f(x) = c$ for all $x \in X_m \cap X$ (otherwise c would not be the minimum).

If f is strictly convex, then for any $x, x' \in X$, and in particular for any $x, x' \in X \cap X_m$, we have (for $x \neq x'$ and all $\lambda \in (0, 1)$),

$$f(\lambda x + (1-\lambda)x') < \lambda f(x) + (1-\lambda) f(x') = \lambda c + (1-\lambda)c = c. \tag{6.5}$$

This contradicts the assumption that $X_m \cap X$ contains more then one element. ■

Global Minima

A simple application of this theorem is in constrained convex minimization. Recall that the notation $[n]$, used below, is a shorthand for $\{1, \ldots, n\}$.

Corollary 6.6 (Constrained Convex Minimization) *Given the convex functions $f, c_1, \ldots, c_n$ on the convex set $\mathcal{X}$, the problem*

$$\begin{aligned} &\underset{x}{\text{minimize}} && f(x), \\ &\textit{subject to} && c_i(x) \leq 0 \textit{ for all } i \in [n], \end{aligned} \tag{6.6}$$

has as its solution a convex set, if a solution exists. This solution is unique if f is strictly convex.

Many problems in Mathematical Programming or Support Vector Machines can be cast into this formulation. This means either that they all have unique solutions (if f is strictly convex), or that all solutions are equally good and form a convex set (if f is merely convex).

We might ask what can be said about convex *maximization.* Let us analyze a simple case first: convex maximization on an interval.

Lemma 6.7 (Convex Maximization on an Interval) *Denote by f a convex function on $[a, b] \in \mathbb{R}$. Then the problem of maximizing f on $[a, b]$ has $f(a)$ and $f(b)$ as the only possible solutions.*

Maxima on Extreme Points

Proof Any $x \in [a, b]$ can be written as $\frac{b-x}{b-a}a + \left(1 - \frac{b-x}{b-a}\right)b$, and hence

$$f(x) \leq \frac{b-x}{b-a}f(a) + \left(1 - \frac{b-x}{b-a}\right)f(b) \leq \max(f(a), f(b)). \tag{6.7}$$

Therefore the maximum of f on $[a, b]$ is obtained on one of the points a, b. ∎

We will next show that the problem of convex *maximization* on a convex set is typically a hard problem, in the sense that the maximum can only be found at one of the extreme points of the constraining set. We must first introduce the notion of vertices of a set.

Definition 6.8 (Vertex of a Set) *A point $x \in X$ is a vertex of X if, for all $x' \in X$ with $x' \neq x$, and for all $\lambda < 0$, the point $\lambda x + (1 - \lambda)x' \notin X$.*

This definition implies, for instance, that in the case of X being an ℓ_2 ball, the vertices of X make up its surface. In the case of an ℓ_∞ ball, we have 2^n vertices in n dimensions, and for an ℓ_1 ball, we have only $2n$ of them. These differences will guide us in the choice of admissible sets of parameters for optimization problems (see, e.g., Section 14.4). In particular, there exists a connection between suprema on sets and their convex hulls. To state this link, however, we need to define the latter.

Definition 6.9 (Convex Hull) *Denote by X a set in a vector space. Then the convex hull $\operatorname{co} X$ is defined as*

$$\operatorname{co} X := \left\{ \bar{x} \,\middle|\, \bar{x} = \sum_{i=1}^{n} \alpha_i x_i \textit{ where } n \in \mathbb{N}, \alpha_i \geq 0 \textit{ and } \sum_{i=1}^{n} \alpha_i \leq 1 \right\}. \tag{6.8}$$

Theorem 6.10 (Suprema on Sets and their Convex Hulls) *Denote by X a set and by $\operatorname{co} X$ its convex hull. Then for a convex function f*

$$\sup\{f(x)|x \in X\} = \sup\{f(x)|x \in \operatorname{co} X\}. \tag{6.9}$$

Evaluating Convex Sets on Extreme Points

Proof Recall that the below set of convex functions is convex (Lemma 6.3), and that the below set of f with respect to $c = \sup\{f(x)|x \in X\}$ is by definition a superset of X. Moreover, due to its convexity, it is also a superset of $\operatorname{co} X$. ∎

This theorem can be used to replace search operations over sets X by subsets $X' \subset X$, which are considerably smaller, if the convex hull of the latter generates X. In particular, the vertices of convex sets are sufficient to reconstruct the whole set.

Theorem 6.11 (Convex Sets and Vertices) *A compact convex set is the convex hull of its vertices.*

Reconstructing Convex Sets from Vertices

The proof is slightly technical, and not central to the understanding of kernel methods. See Rockafellar [419, 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 $X = \operatorname{co}(|X)$. Figure 6.4 depicts the situation graphically. ■

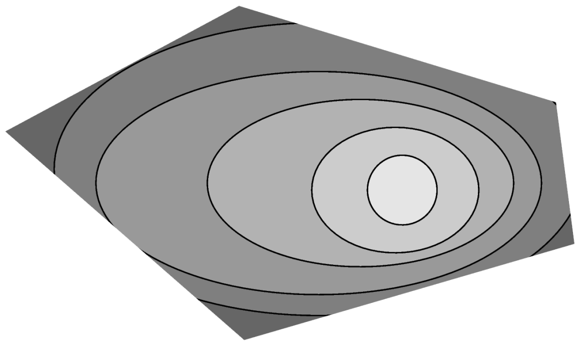


Figure 6.4 A convex function on a convex polyhedral set. Note that the minimum of this function is unique, and that the maximum can be found at one of the vertices of the constraining domain.

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.

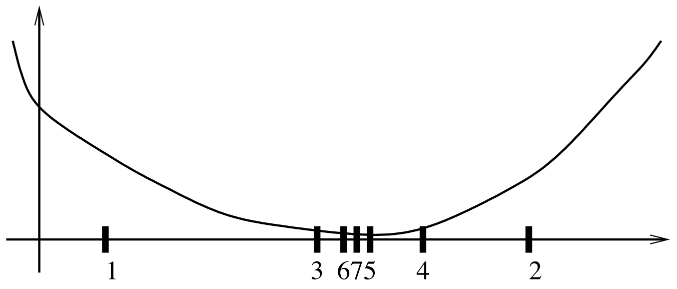


Figure 6.5 Interval Cutting Algorithm. The selection of points is ordered according to the numbers beneath (points 1 and 2 are the initial endpoints of the interval).

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 monotonic, and we can find a fast, simple algorithm (Algorithm 6.1) to solve our problem (see Figure 6.5).

Algorithm 6.1 Interval Cutting

```
Require: $a, b$, Precision $\epsilon$
  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

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*

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

Proof of Linear Convergence

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,

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

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.

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 e.g. [315] or Problem 6.3).

Newton's Method

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. 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.

Algorithm 6.2 Newton's Method

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

Quadratic Convergence

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.*

Proof For convenience, denote by x_n the value of x at the n-th 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

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

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

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

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 e.g. [163, 519, 315, 15, 152, 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 [315] 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

$$x_{n+1} = x_n - \gamma g_n \qquad (6.16)$$

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 monotically 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]

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

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.

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 (see e.g. [315]) that after a (possibly infinite) number of steps, gradient descent (see Algorithm 6.3) will converge.

Algorithm 6.3 Gradient Descent

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

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.

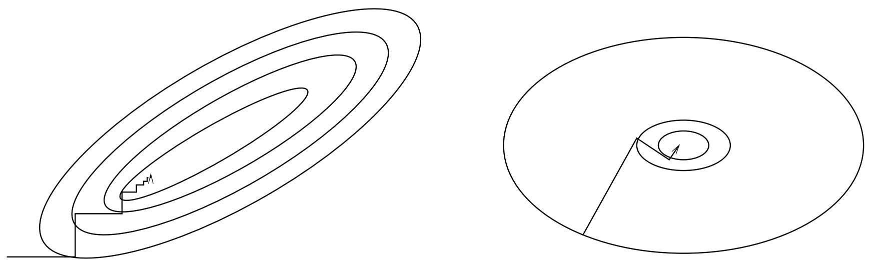


Figure 6.6 Left: Gradient descent takes a long time to converge, since the landscape of values of f forms a long and narrow valley, causing the algorithm to zig-zag along the walls of the valley. Right: due to the homogeneous structure of the minimum, the algorithm converges after very few iterations. 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 $f(x^*) = 0$. 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

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

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.

Lower Bound for Improvement

Theorem 6.16 (Kantorovich Inequality [262]) *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 $g \in \mathbb{R}^m$:*

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

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^*.

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 [262, 315] for more detail.

A brief calculation gives us the correct order of magnitude. Note that for any x, the quadratic term $x^\top Kx$ 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 [173, 12] or AdaLine [175], 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* [232, 407, 519]). 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

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

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

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

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

Linear Independence

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

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.

Optimality in Linear Space

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 $\mathfrak{X}_i := x_0 + \operatorname{span}\{v_1, \ldots, v_i\}$.*

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

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 $\mathfrak{X}_i$. 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 algorithm: 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.

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: $x_i$
```

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 $f(x) = \frac{1}{2} x^\top K x - c^\top x$, 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) $\operatorname{span}\{g_0, \ldots, g_i\} = \operatorname{span}\{v_0, \ldots, v_i\} = \operatorname{span}\{g_0, K g_0, \ldots, K^i g_0\}$.

(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 optimal solution within the linear manifold given by$x_0 + \mathrm{span}\{g_0, Kg_0, \ldots, K^{i-1}g_0\}$.

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 $\mathrm{span}\{g_0, \ldots, g_{i+1}\} = \mathrm{span}\{g_0, Kg_0, \ldots, K^{i+1}g_0\}$. Since $v_{i+1} = -g_{i+1} + \beta_i v_i$ the same statement holds for $\mathrm{span}\{v_0, \ldots, v_{i+1}\}$. 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.
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 optimal 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 an optimal solution on the linear manifold $x_0 + \mathrm{span}\{g_0, Kg_0, \ldots, K^{i-1}g_0\}$. 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 e.g. [315]

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 e.g. [163, 315, 519, 398] for details.

6.2.5 Predictor Corrector Methods

Increasing the Order

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 complicated 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 (e.g. for interior point programs, see Section 6.4). We might instead

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 [163]	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 [398]	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.

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 achived 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

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

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.

No Quadratic Residuals

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

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

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.

Algorithm 6.5 Predictor Corrector Method

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

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:

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

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

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

Before we start minimizing f, we have to discuss what optimality means in this case. Clearly $f'(x) = 0$ is too restrictive a condition. For instance, f' could point into a direction which is forbidden by the constraints c_i and e_i. Then we could have optimality, even though $f' \neq 0$. Let us analyze the situation in more detail.

6.3.1 Optimality Conditions

We start with optimality conditions for optimization problems which are independent of their differentiability. While it is fairly straightforward to state *sufficient* optimality conditions for arbitrary functions f and c_i, we will need convexity and "reasonably nice" constraints (see Lemma 6.23) to state *necessary* conditions. This is not a major concern, since for practical applications, the constraint qualification criteria are almost always satisfied, and the functions themselves are usually convex and differentiable. Much of the reasoning in this section follows [327], which should also be consulted for further references and detail.

Some of the most important sufficient criteria are the Kuhn-Tucker[4] saddlepoint conditions [294]. As indicated previously, they are independent of assumptions on convexity or differentiability of the constraints c_i or objective function f.

Lagrange Function

Theorem 6.21 (Kuhn-Tucker Saddlepoint Condition [294, 327]) *Assume an optimization problem of the form (6.37), where $f : \mathbb{R}^m \to \mathbb{R}$ and $c_i : \mathbb{R}^m \to \mathbb{R}$ for $i \in [n]$ are arbitrary functions, and a Lagrange function*

$$L(x, \alpha) := f(x) + \sum_{i=1}^{n} \alpha_i c_i(x) \text{ where } \alpha_i \geq 0. \tag{6.39}$$

If a pair of variables $(\bar{x}, \bar{\alpha})$ with $\bar{x} \in \mathbb{R}^n$ and $\bar{\alpha}_i \geq 0$ for all $i \in [n]$ exists, such that for all $x \in \mathbb{R}^m$ and $\alpha \in [0, \infty)^n$,

$$L(\bar{x}, \alpha) \leq L(\bar{x}, \bar{\alpha}) \leq L(x, \bar{\alpha}) \text{ (Saddlepoint)} \tag{6.40}$$

then $\bar{x}$ is an optimal solution to (6.37).

The parameters α_i are called Lagrange multipliers. As described in the later

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.
4. An earlier version is due to Karush [267]. This is why often one uses the abbreviation KKT (Karush-Kuhn-Tucker) rather than KT to denote the optimality conditions.

chapters, they will become the coefficients in the kernel expansion in Support Vector Machines.

Proof The proof follows [327]. Denote by $(\bar{x}, \bar{\alpha})$ a pair of variables satisfying (6.40). From the first inequality it follows that

$$\sum_{i=1}^{n} (\alpha_i - \bar{\alpha}_i) c_i(\bar{x}) \leq 0. \tag{6.41}$$

Since we are free to choose $\alpha_i \geq 0$, we can see (by setting all but one of the terms α_i to $\bar{\alpha}_i$ and the remaining one to $\alpha_i = \bar{\alpha}_i + 1$) that $c_i(x) \leq 0$ for all $i \in [n]$. This shows that $\bar{x}$ satisfies the constraints, i.e. it is feasible.

Additionally, by setting one of the α_i to 0, we see that $\bar{\alpha}_i c_i(\bar{x}) \geq 0$. The only way to satisfy this is by having

$$\bar{\alpha}_i c_i(\bar{x}) = 0 \text{ for all } i \in [n]. \tag{6.42}$$

Eq. (6.42) is often referred to as the Karush-Kuhn-Tucker (KKT) condition [267, 294]. Finally, combining (6.42) and $c_i(x) \leq 0$ with the second inequality in (6.40) yields $f(\bar{x}) \leq f(x)$ for all feasible x. This proves that $\bar{x}$ is optimal. ■

We can immediately extend Theorem 6.21 to accommodate equality constraints by splitting them into the conditions $e_i(x) \leq 0$ and $e_i(x) \geq 0$. We obtain:

Theorem 6.22 (Equality Constraints) *Assume an optimization problem of the form (6.38), where $f, c_i, e_j : \mathbb{R}^m \to \mathbb{R}$ for $i \in [n]$ and $j \in [n']$ are arbitrary functions, and a Lagrange function*

$$L(x, \alpha) := f(x) + \sum_{i=1}^{n} \alpha_i c_i(x) + \sum_{j=1}^{n'} \beta_j e_j(x) \text{ where } \alpha_i \geq 0 \text{ and } \beta_j \in \mathbb{R}. \tag{6.43}$$

If a set of variables $(\bar{x}, \bar{\alpha}, \bar{\beta})$ with $\bar{x} \in \mathbb{R}^m$, $\bar{\alpha} \in [0, \infty)$, and $\bar{\beta} \in \mathbb{R}^{n'}$ exists such that for all $x \in \mathbb{R}^m$, $\alpha \in [0, \infty)^n$, and $\beta \in \mathbb{R}^{n'}$,

$$L(\bar{x}, \alpha, \beta) \leq L(\bar{x}, \bar{\alpha}, \bar{\beta}) \leq L(x, \bar{\alpha}, \bar{\beta}), \tag{6.44}$$

then $\bar{x}$ is an optimal solution to (6.38).

Now we determine when the conditions of Theorem 6.21 are necessary. We will see that convexity and sufficiently "nice" constraints are needed for (6.40) to become a necessary condition. The following lemma (see [327]) describes three *constraint qualifications*, which will turn out to be exactly what we need.

Feasible Region

Lemma 6.23 (Constraint Qualifications) *Denote by $\mathcal{X} \subset \mathbb{R}^m$ a convex set, and by $c_1, \ldots, c_n : \mathcal{X} \to \mathbb{R}$ n convex functions defining a feasible region by*

$$X := \{x | x \in \mathcal{X} \text{ and } c_i(x) \leq 0 \text{ for all } i \in [n]\}. \tag{6.45}$$

Equivalence Between Constraint Qualifications

Then the following additional conditions on c_i are connected by (i) $\Longleftrightarrow$ (ii) and (iii) $\Longrightarrow$ (i).

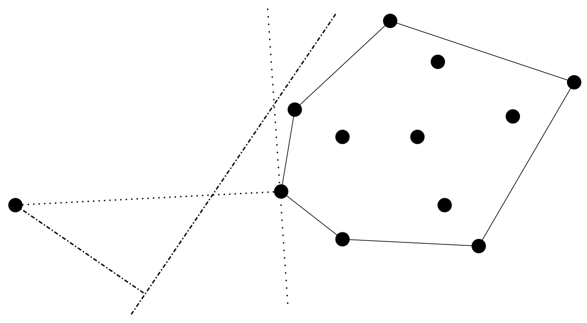


Figure 6.7 Two hyperplanes (and their normal vectors) separating the convex hull of a finite set of points from the origin.

(i) There exists an $x \in \mathcal{X}$ such that for all $i \in [n]$ $c_i(x) < 0$ (Slater's condition [487]).

(ii) For all nonzero $\alpha \in [0, \infty)^n$ there exists an $x \in \mathcal{X}$ such that $\sum_{i=1}^n \alpha_i c_i(x) \leq 0$ (Karlin's condition [265]).

(iii) The feasible region X contains at least two distinct elements, and there exists an $x \in X$ such that all c_i are strictly convex at x wrt. X (Strict constraint qualification).

The connection $(i) \Longleftrightarrow (ii)$ is also known as the Generalized Gordan Theorem [155]. The proof can be skipped if necessary. We need an auxiliary lemma which we state without proof (see [327, 419] for details).

Lemma 6.24 (Separating Hyperplane Theorem) *Denote by $X \in \mathbb{R}^m$ a convex set not containing the origin 0. Then there exists a hyperplane with normal vector $\alpha \in \mathbb{R}^m$ such that $\alpha^\top x > 0$ for all $x \in X$.*

See also Figure 6.7.

Proof of Lemma 6.23. We prove $\{(i) \Longleftrightarrow (ii)\}$ by showing $\{(i) \Longrightarrow (ii)\}$ and $\{\text{ not } (i) \Longrightarrow \text{not } (ii)\}$.

$(i) \Longrightarrow (ii)$ For a point $x \in X$ with $c_i(x) < 0$, for all $i \in [n]$ we have that $\alpha_i c_i(x) \geq 0$ implies $\alpha_i = 0$.

$\overline{(i)} \Longrightarrow \overline{(ii)}$ Assume that there is no x with $c_i(x) < 0$ for all $i \in [n]$. Hence the set

$$\Gamma := \{\gamma | \gamma \in \mathbb{R}^n \text{ and there exists some } x \in X \text{ with } \gamma_i > c_i(x) \text{ for all } i \in [n]\} \quad (6.46)$$

is convex and does not contain the origin. The latter follows directly from the assumption. For the former take $\gamma, \gamma' \in \Gamma$ and $\lambda \in (0, 1)$ to obtain

$$\lambda\gamma_i + (1 - \lambda)\gamma'_i > \lambda c_i(x) + (1 - \lambda)c_i(x') \geq c_i(\lambda x + (1 - \lambda)x'). \quad (6.47)$$

Now by Lemma 6.24, there exists some $\alpha \in \mathbb{R}^n$ such that $\alpha^\top \gamma \geq 0$ and $\|\alpha\|^2 = 1$ for all $\gamma \in \Gamma$. Since each of the γ_i for $\gamma \in \Gamma$ can be arbitrarily large (with respect to the other coordinates), we conclude $\alpha_i \geq 0$ for all $i \in [n]$.

Denote by $\delta := \inf_{x \in X} \sum_{i=1}^n \alpha_i c_i(x)$ and by $\delta' := \inf_{\gamma \in \Gamma} \alpha^\top \gamma$. One can see that by construction $\delta = \delta'$. By Lemma 6.24 α was chosen such that $\delta' \geq 0$, and hence

$\delta \geq 0$. This contradicts (ii), however, since it implies the existence of a suitable α with $\alpha_i c_i(x) \geq 0$ for all x.

$(iii) \Longrightarrow (i)$ Since X is convex we get for all c_i and for any $\lambda \in (0,1)$:

$$\lambda x + (1-\lambda)x' \in X \text{ and } 0 \geq \lambda c_i(x) + (1-\lambda)c_i(x') > c_i(\lambda x + (1-\lambda)x'). \tag{6.48}$$

This shows that $\lambda x + (1-\lambda)x'$ satisfies (i) and we are done. ■

We proved Lemma 6.23 as it provides us with a set of constraint qualifications (conditions on the constraints) that allow us to determine cases where the Kuhn-Tucker Saddlepoint conditions are both *necessary* and sufficient. This is important, since we will use the Kuhn-Tucker conditions to transform optimization problems into their duals, and solve the latter numerically. For this approach to be valid, however, we must ensure that we do not change the solvability of the optimization problem.

Theorem 6.25 (Necessary Kuhn-Tucker Conditions [294, 541, 265]) *Under the assumptions and definitions of Theorem 6.21 with the additional assumption that f and c_i are convex on the convex set $X \subseteq \mathbb{R}^m$ (containing the set of feasible solutions as a subset) and that c_i satisfy one of the constraint qualifications of Lemma 6.23, the saddlepoint criterion (6.40) is necessary for optimality.*

Proof Denote by $\bar{x}$ the solution to (6.37), and by X' the set

$$X' := X \cap \{x | x \in \mathcal{X} \text{ with } f(x) - f(\bar{x}) \leq 0 \text{ and } c_i(x) \leq 0 \text{ for all } i \in [n]\}. \tag{6.49}$$

By construction $\bar{x} \in X'$. Furthermore, there exists no $x' \in X'$ such that all inequality constraints including $f(x) - f(\bar{x})$ are satisfied as *strict* inequalities (otherwise $\bar{x}$ would not be optimal). In other words, X' violates Slater's conditions (i) of Lemma 6.23 (where both $(f(x) - f(\bar{x}))$ and $c(x)$ *together* play the role of $c_i(x)$), and thus also Karlin's conditions (ii). This means that there exists a nonzero vector $(\bar{\alpha}_0, \bar{\alpha}) \in \mathbb{R}^{n+1}$ with nonnegative entries such that

$$\bar{\alpha}_0(f(x) - f(\bar{x})) + \sum_{i=1}^n \bar{\alpha}_i c_i(x) \geq 0 \text{ for all } x \in \mathcal{X}. \tag{6.50}$$

In particular, for $x = \bar{x}$ we get $\sum_{i=1}^n \bar{\alpha}_i c_i(\bar{x}) \geq 0$. In addition, since $\bar{x}$ is a solution to (6.37), we have $c_i(\bar{x}) \leq 0$. Hence $\sum_{i=1}^n \bar{\alpha}_i c_i(\bar{x}) = 0$. This allows us to rewrite (6.50) as

$$\bar{\alpha}_0 f(x) + \sum_{i=1}^n \bar{\alpha}_i c_i(x) \geq \bar{\alpha}_0 f(\bar{x}) + \sum_{i=1}^n \bar{\alpha}_i c_i(\bar{x}). \tag{6.51}$$

This looks almost like the first inequality of (6.40), except for the $\bar{\alpha}_0$ term (which we will return to later). But let us consider the second inequality first.

Again, since $c_i(\bar{x}) \leq 0$ we have $\sum_{i=1}^n \alpha_i c_i(\bar{x}) \leq 0$ for all $\alpha_i \geq 0$. Adding $\bar{\alpha}_0 f(\bar{x})$

on both sides of the inequality and $\sum_{i=1}^n \bar{\alpha}_i c_i(\bar{x})$ on the RHS yields

$$\bar{\alpha}_0 f(\bar{x}) + \sum_{i=1}^n \bar{\alpha}_i c_i(\bar{x}) \geq \bar{\alpha}_0 f(\bar{x}) + \sum_{i=1}^n \alpha_i c_i(\bar{x}). \tag{6.52}$$

This is almost all we need for the first inequality of (6.40) .[5] If $\bar{\alpha}_0 > 0$ we can divide (6.51) and (6.52) by $\bar{\alpha}_0$ and we are done.

When $\bar{\alpha}_0 = 0$, then this implies the existence of $\bar{\alpha} \in \mathbb{R}^n$ with nonnegative entries satisfying $\sum_{i=1}^n \bar{\alpha}_i c_i(x) \geq 0$ for all $x \in X$. This contradicts Karlin's constraint qualification condition (ii), which allows us to rule out this case. ■

6.3.2 Duality and KKT-Gap

Now that we have formulated necessary and sufficient optimality conditions (Theorem 6.21 and 6.25) under quite general circumstances, let us put them to practical use for convex differentiable optimization problems. We first derive a more practically useful form of Theorem 6.21. Our reasoning is as follows: eq. (6.40) implies that $L(\bar{x}, \bar{\alpha})$ is a *saddlepoint* in terms of $(\bar{x}, \bar{\alpha})$. Hence, all we have to do is write the saddlepoint conditions in the form of derivatives.

Primal and Dual Feasibility

Theorem 6.26 (Kuhn-Tucker for Differentiable Convex Problems [294]) *An optimal solution to the optimization problem (6.37) with convex, differentiable f, c_i is given by $\bar{x}$, if there exists some $\bar{\alpha} \in \mathbb{R}^n$ with $\alpha_i \geq 0$ for all $i \in [n]$ such that the following conditions are satisfied:*

$$\partial_x L(\bar{x}, \bar{\alpha}) = \partial_x f(\bar{x}) + \sum_{i=1}^n \bar{\alpha}_i \partial_x c_i(\bar{x}) = 0 \text{ (Saddlepoint in } \bar{x}\text{)}, \tag{6.53}$$

$$\partial_{\alpha_i} L(\bar{x}, \bar{\alpha}) = c_i(\bar{x}) \leq 0 \text{ (Saddlepoint in } \bar{\alpha}\text{)}, \tag{6.54}$$

$$\sum_{i=1}^n \bar{\alpha}_i c_i(\bar{x}) = 0 \text{ (Vanishing KKT-Gap)}. \tag{6.55}$$

Proof The easiest way to prove Theorem 6.26 is to show that for any $x \in X$, we have $f(x) - f(\bar{x}) \geq 0$. Due to convexity we may linearize and obtain

$$\begin{aligned} f(x) - f(\bar{x}) &\geq (\partial_x f(\bar{x}))^\top (x - \bar{x}) && (6.56)\\ &= -\sum_{i=1}^n \bar{\alpha}_i \left(\partial_x c_i(\bar{x})\right)^\top (x - \bar{x}) && (6.57)\\ &\geq -\sum_{i=1}^n \bar{\alpha}_i (c_i(x) - c_i(\bar{x})) && (6.58) \end{aligned}$$

5. The two inequalities (6.51) and (6.52) are also known as the Fritz-John saddlepoint necessary optimality conditions [253], which play a similar role as the saddlepoint conditions for the Lagrangian (6.39) of Theorem 6.21.

$$= -\sum_{i=1}^{n} \bar{\alpha}_i c_i(x) \geq 0. \tag{6.59}$$

Here we used the convexity and differentiability of f to arrive at the rhs of (6.56) and (6.58). To obtain (6.57) we exploited the fact that at the saddlepoint $\partial_x f(\bar{x})$ can be replaced by the corresponding expansion in $\partial_x c_i(\bar{x})$; thus we used (6.53). Finally, for (6.59) we used the fact that the KKT gap vanishes at the optimum (6.55) and that the constraints are satisfied (6.54). ■

Optimization by Constraint Satisfaction

In other words, we may solve a convex optimization problem by finding $(\bar{x}, \bar{\alpha})$ that satisfy the conditions of Theorem 6.26. Moreover, these conditions, together with the constraint qualifications of Lemma 6.23, ensure necessity.

Note that we transformed the problem of minimizing functions into one of solving a set of equations, for which several numerical tools are readily available. This is exactly how interior point methods work (see Section 6.4 for details on how to implement them). Necessary conditions on the constraints similar to those discussed previously can also be formulated (see [327] for a detailed discussion).

The other consequence of Theorem 6.26, or rather of the definition of the Lagrangian $L(x, \alpha)$, is that we may bound $f(\bar{x}) = L(\bar{x}, \bar{\alpha})$ from above and below *without* explicit knowledge of $f(\bar{x})$.

Theorem 6.27 (KKT-Gap) *Assume an optimization problem of type (6.37), where both f and c_i are convex and differentiable. Denote by $\bar{x}$ its solution. Then for any set of variables (x, α) with $\alpha_i \geq 0$, and for all $i \in [n]$ satisfying*

$$\partial_x L(x, \alpha) = 0, \tag{6.60}$$

$$\partial_{\alpha_i} L(x, \alpha) \leq 0 \text{ for all } i \in [n], \tag{6.61}$$

Bounding the Error

we have

$$f(x) \geq f(\bar{x}) \geq f(x) + \sum_{i=1}^{m} \alpha_i c_i(x). \tag{6.62}$$

Strictly speaking, we only need differentiability of f and c_i at $\bar{x}$. However, since $\bar{x}$ is only known *after* the optimization problem has been solved, this is not a very useful condition.

Proof The first part of (6.62) follows from the fact that $x \in X$, so that x satisfies the constraints. Next note that $L(\bar{x}, \bar{\alpha}) = f(\bar{x})$ where $(\bar{x}, \bar{\alpha})$ denotes the saddlepoint of L. For the second part note that due to the saddlepoint condition (6.40), we have for any α with $\alpha_i \geq 0$,

$$f(\bar{x}) = L(\bar{x}, \bar{\alpha}) \geq L(\bar{x}, \alpha) \geq \inf_{x' \in X} L(x', \alpha). \tag{6.63}$$

The function $L(x', \alpha)$ is convex in x' since both f' and the constraints c_i are convex and all $\alpha_i \geq 0$. Therefore (6.60) implies that x minimizes $L(x', \alpha)$. This proves the second part of (6.63), which in turn proves the second inequality of (6.62). ■

Hence, no matter what algorithm we are using in order to solve (6.37), we may always use (6.62) to assess the proximity of the current set of parameters to the optimal solution. Clearly, the relative size of $\sum_{i=1}^n \alpha_i c_i(x)$ provides a useful stopping criterion for convex optimization algorithms.

Finally, another concept that is useful when dealing with optimization problems is that of *duality*. This means that for the *primal* minimization problem considered so far, which is expressed in terms of x, we can find a *dual* maximization problem in terms of α by computing the saddlepoint of the Lagrangian $L(x, \alpha)$, and eliminating the primal variables x. We thus obtain the following dual maximization problem from (6.37):

$$\begin{aligned} &\text{maximize} && L(x,\alpha) = f(x) + \sum_{i=1}^n \alpha_i c_i(x), \\ &\text{where} && (x,\alpha) \in Y := \left\{ (x,\alpha) \,\middle|\, \begin{array}{l} x \in X, \alpha_i \geq 0 \text{ for all } i \in [n] \\ \text{and } \partial_x L(x,\alpha) = 0 \end{array} \right\}. \end{aligned} \tag{6.64}$$

We state without proof a theorem guaranteeing the existence of a solution to (6.64).

Existence of Dual Solution

Theorem 6.28 (Wolfe [595]) *Recall the definition of X (6.45) and of the optimization problem (6.37). Under the assumptions that $\mathcal{X}$ is an open set, X satisfies one of the constraint qualifications of Lemma 6.23, and f, c_i are all convex and differentiable, there exists an $\bar{\alpha} \in \mathbb{R}^n$ such that $(\bar{x}, \bar{\alpha})$ solves the dual optimization problem (6.64) and in addition $L(\bar{x}, \bar{\alpha}) = f(\bar{x})$.*

In order to prove Theorem 6.28 we first have to show that some $(\bar{x}, \bar{\alpha})$ exists satisfying the Kuhn-Tucker conditions, and then use the fact that the KKT-Gap at the saddlepoint vanishes.

6.3.3 Linear and Quadratic Programs

Primal Linar Program

Let us analyze the notions of primal and dual objective functions in more detail by looking at linear and quadratic programs. We begin with a simple linear setting.[6]

$$\begin{aligned} &\underset{x}{\text{minimize}} && c^\top x \\ &\text{subject to} && Ax + d \leq 0 \end{aligned} \tag{6.65}$$

where $c, x \in \mathbb{R}^m$, $d \in \mathbb{R}^n$ and $A \in R^{n \times m}$, and where $Ax + d \leq 0$ is a shorthand for $\sum_{j=1}^m A_{ij} x_j + d_i \leq 0$ for all $i \in [n]$.

6. Note that we encounter a small clash of notation in (6.65), since c is used as a symbol for the loss function in the remainder of the book. This inconvenience is outweighed, however, by the advantage of remaining consistent with the standard literature (e.g., [327, 45, 543]) on optimization. The latter will allow the reader to read up on the subject without any need for cumbersome notational changes.

Unbounded and Infeasible Problems

It is far from clear that (6.65) always has a solution, or indeed a minimum. For instance, the set of x satisfying $Ax + d \leq 0$ might be empty, or it might contain rays going to infinity in directions where $c^\top x$ keeps increasing. Before we deal with this issue in more detail, let us compute the sufficient Kuhn-Tucker conditions for optimality, and the dual of (6.65). We may use (6.26) since (6.65) is clearly differentiable and convex. In particular we obtain:

Theorem 6.29 (Kuhn-Tucker Conditions for Linear Programs) *A sufficient condition for a solution to the linear program (6.65) to exist is that the following four conditions are satisfied for some $(x, \alpha) \in \mathbb{R}^{m+n}$ where $\alpha \geq 0$:*

$$\partial_x L(x,\alpha) = \partial_x \left[c^\top x + \alpha^\top (Ax + d)\right] = A^\top \alpha + c = 0, \tag{6.66}$$

$$\partial_\alpha L(x,\alpha) = Ax + d \leq 0, \tag{6.67}$$

$$\alpha^\top (Ax + d) = 0, \tag{6.68}$$

$$\alpha \geq 0. \tag{6.69}$$

Then the minimum is given by $c^\top x$.

Note that, depending on the choice of A and d, there may not always exist an x such that $Ax + d \leq 0$, in which case the constraint does not satisfy the conditions of Lemma 6.23. In this situation, no solution exists for (6.65). If a feasible x exists, however, then (projections onto lower dimensional subspaces aside) the constraint qualifications are satisfied on the feasible set, and the conditions above are necessary. See e.g. [315, 327, 543] for details.

Dual Linear Program

Next we may compute Wolfe's dual optimization problem by substituting (6.66) into $L(x, \alpha)$. Consequently, the primal variables x vanish, and we obtain a maximization problem in terms of α only:

$$\begin{array}{ll} \text{maximize} & d^\top \alpha, \\ \text{subject to} & A^\top \alpha + c = 0 \text{ and } \alpha \geq 0. \end{array} \tag{6.70}$$

Note that the number of variables and constraints has changed: we started with m variables and n constraints. Now we have n variables together with m equality constraints and n inequality constraints. While it is not yet completely obvious in the linear case, dualization may render optimization problems more amenable to numerical solution (the contrary may be true as well, though).

Primal Solution $\Leftrightarrow$ Dual Solution

What happens if a solution $\bar{x}$ to the primal problem (6.65) exists? In this case we know (since the Kuhn-Tucker conditions of Theorem 6.29 are necessary and sufficient) that there must be an $\bar{\alpha}$ solving the dual problem, since $L(x, \alpha)$ has a saddlepoint at $(\bar{x}, \bar{\alpha})$.

If no feasible point of the primal problem exists, there must exist, by (a small modification of) Lemma 6.23, some $\alpha \in \mathbb{R}^n$ with $\alpha \geq 0$ and at least one $\alpha_i > 0$ such that $\alpha^\top (Ax + d) > 0$ for all x. This means that for all x, the Lagrange function $L(x, \alpha)$ is unbounded from above, since we can make $\alpha^\top (Ax + d)$ arbitrarily large. Hence the dual optimization problem is unbounded. Using analogous reasoning, if the primal problem is unbounded, the dual problem is infeasible.

Table 6.2 Connections between primal and dual optimization problems.

Primal Optimization Problem (in x)	Dual Optimization Problem (in α)
solution exists	solution exists and is equal to primal solution
no solution exists	maximization problem is unbounded
minimization problem is unbounded	no solution exists
inequality constraint	inequality constraint
equality constraint	free variable
free variable	equality constraint

Let us see what happens if we dualize (6.70) one more time. First we need more Lagrange multipliers, since we have two sets of constraints. The equality constraints can be taken care of by an unbounded variable x' (see Theorem 6.22 for how to deal with equalities). For the inequalities $\alpha \geq 0$, we introduce a second Lagrange multiplier $y \in \mathbb{R}^n$. After some calculations and resubstitution into the corresponding Lagrange function, we get

Dual Dual Linear Program $\rightarrow$ Primal

$$\begin{aligned} &\text{maximize} && c^\top x', \\ &\text{subject to} && Ax' + d + y = 0 \text{ and } y \geq 0. \end{aligned} \tag{6.71}$$

We can remove $y \geq 0$ from the set of variables by transforming $Ax' + d + y$ into $Ax + d \leq 0$; thus we recover the primal optimization problem (6.65).[7] Table 6.2 gives an overview of the transformations and relations between primal and dual problems. Even though we only derived these relations for linear programs, this approach can also be applied to other convex differentiable settings such as quadratic programs [45].

We conclude this section by stating primal and dual optimization problems, and the sufficient Kuhn-Tucker conditions for convex quadratic optimization problems. To keep matters simple we only consider the following type of optimization problem (other problems can be rewritten in the same form; see Problem 6.11 for details):

Primal Quadratic Program

$$\begin{aligned} &\underset{x}{\text{minimize}} && \tfrac{1}{2} x^\top K x + c^\top x, \\ &\text{subject to} && Ax + d \leq 0. \end{aligned} \tag{6.72}$$

Here K is a strictly positive definite matrix, $x, c \in \mathbb{R}^m$, $A \in \mathbb{R}^{n \times m}$, and $d \in \mathbb{R}^n$. Note that this is clearly a differentiable convex optimization problem. To introduce a Lagrange function we need corresponding multipliers $\alpha \in \mathbb{R}^n$ with $\alpha \geq 0$. We obtain

$$L(x, \alpha) = \frac{1}{2} x^\top K x + c^\top x + \alpha^\top (Ax + d). \tag{6.73}$$

7. This finding is useful if we have to dualize twice in some optimization settings (see Chapter 10), since then we will be able to recover some of the primal variables without further calculations if the optimization algorithm provides us with both primal and dual variables.

Next we may apply Theorem 6.26 to obtain the Kuhn-Tucker conditions. They can be stated in analogy to (6.66)–(6.68) as

$$\partial_x L(x,\alpha) = \partial_x \left[c^\top x + \alpha^\top (Ax+d) + \frac{1}{2} x^\top K x \right] = Kx + A^\top \alpha + c = 0, \tag{6.74}$$

$$\partial_\alpha L(x,\alpha) = Ax + d \leq 0, \tag{6.75}$$

$$\alpha^\top (Ax+d) = 0, \tag{6.76}$$

$$\alpha \geq 0. \tag{6.77}$$

In order to compute the dual of (6.72), we have to eliminate x from (6.73) and write it as a function of α. We obtain

$$L(x,\alpha) = -\frac{1}{2} x^\top K x + \alpha^\top d \tag{6.78}$$

$$= -\frac{1}{2} \alpha^\top A^\top K^{-1} A \alpha + \left[d - c^\top K^{-1} A^\top \right] \alpha - \frac{1}{2} c^\top K^{-1} c. \tag{6.79}$$

In (6.78) we used (6.74) and (6.76) directly, whereas in order to eliminate x completely in (6.79) we solved (6.74) for $x = -K^{-1}(c + A^\top \alpha)$. Ignoring constant terms this leads to the dual quadratic optimization problem,

Dual Quadratic Program

$$\begin{array}{ll} \underset{\alpha}{\text{minimize}} & -\frac{1}{2} \alpha^\top A^\top K^{-1} A \alpha + \left[d - c^\top K^{-1} A^\top \right] \alpha, \\ \text{subject to} & \alpha \geq 0. \end{array} \tag{6.80}$$

The surprising fact about the dual problem (6.80) is that the constraints become significantly simpler than in the primal (6.72). Furthermore, if $n < m$, we also obtain a more compact representation of the quadratic term.

There is one aspect in which (6.80) differs from its linear counterpart (6.70): if we dualize (6.80) again, we do not recover (6.72) but rather a problem very similar in structure to (6.80). Dualizing (6.80) twice, however, we recover the dual itself (Problem 6.13 deals with this matter in more detail).

6.4 Interior Point Methods

Let us now have a look at simple, yet efficient optimization algorithms for constrained problems: interior point methods.

An interior point is a pair of variables (x, α) that satisfies both primal and dual constraints. As already mentioned before, finding a set of vectors $(\bar{x}, \bar{\alpha})$ that satisfy the Kuhn-Tucker conditions is sufficient to obtain an optimal solution in $\bar{x}$. Hence, all we have to do is devise an algorithm which solves (6.74)–(6.77), for instance, if we want to solve a quadratic program. We will focus on the quadratic case — the changes required for linear programs merely involve the removal of some variables, simplifying the equations. See Problem 6.14 and [543, 496] for details.

6.4.1 Sufficient Conditions for a Solution

Optimality as Constraint Satisfaction

We need a slight modification of (6.74)–(6.77) in order to achieve our goal: rather than the inequality (6.75), we are better off with an equality and a positivity constraint for an additional variable, i.e. we transform $Ax+d \leq 0$ into $Ax+d+\xi = 0$, where $\xi \geq 0$. Hence we arrive at the following system of equations:

$$\begin{aligned} Kx + A^\top \alpha + c &= 0 \quad \text{(Dual Feasibility)},\\ Ax + d + \xi &= 0 \quad \text{(Primal Feasibility)},\\ \alpha^\top \xi &= 0,\\ \alpha, \xi &\geq 0. \end{aligned} \tag{6.81}$$

Let us analyze the equations in more detail. We have three sets of variables: x, α, ξ. To determine the latter, we have an equal number of equations plus the positivity constraints on α, ξ. While the first two equations are linear and thus amenable to solution, e.g. by matrix inversion, the third equality $\alpha^\top \xi = 0$ has a small defect: given one variable, say α, we cannot solve it for ξ or vice versa. Furthermore, the last two constraints are not very informative either. So we need a strategy to improve this situation.

We use a primal-dual path-following algorithm, as proposed in [542], to solve this problem. Rather than requiring $\alpha^\top \xi = 0$ we modify it to become $\alpha_i \xi_i = \mu > 0$ for all $i \in [n]$, solve (6.81) for a given μ, and decrease μ to 0 as we go. The advantage of this strategy is that we may use a Newton-type predictor corrector algorithm (see Section 6.2.5) to update the parameters x, α, ξ, which exhibits the fast convergence of a second order method.

6.4.2 Solving the Equations

Linearized Constraints

For the moment, assume that we have suitable initial values of x, α, ξ, and μ with $\alpha, \xi > 0$. Linearization of the first three equations of (6.81), together with the modification $\alpha_i \xi_i = \mu$, yields (we expand x into $x + \Delta x$, etc.):

$$\begin{aligned} K\Delta x + A^\top \Delta\alpha &= -Kx - A^\top \alpha - c &&=: \rho_p,\\ A\Delta x + \Delta\xi &= -Ax - d - \xi &&=: \rho_d,\\ \alpha_i^{-1}\xi_i \Delta\alpha_i + \Delta\xi_i &= \mu\alpha_i^{-1} - \xi_i - \alpha_i^{-1}\Delta\alpha_i\Delta\xi_i &&=: \rho_{\text{KKT}\,i} \text{ for all } i \end{aligned} \tag{6.82}$$

Next we solve for $\Delta\xi_i$ to obtain what is commonly referred to as the *reduced* KKT system. For convenience we use $D := \operatorname{diag}(\alpha_1^{-1}\xi_1, \ldots, \alpha_n^{-1}\xi_n)$ as a shorthand;

$$\begin{bmatrix} K & A^\top \\ A & -D \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\alpha \end{bmatrix} = \begin{bmatrix} \rho_p \\ \rho_d - \rho_{\text{KKT}} \end{bmatrix}. \tag{6.83}$$

We apply a predictor-corrector method as in Section 6.2.5. The resulting matrix of the linear system in (6.83) is indefinite but of full rank, and we can solve (6.83) for $(\Delta x_{\text{Pred}}, \Delta\alpha_{\text{Pred}})$ by explicitly pivoting for individual entries (e.g. solve for Δx first and then substitute the result in to the second equality to obtain $\Delta\alpha$).

This gives us the *predictor* part of the solution. Next we have to correct for the linearization, which is conveniently achieved by updating ρ_{KKT} and solving (6.83) again to obtain the *corrector* values $(\Delta x_{\mathrm{Corr}}, \Delta\alpha_{\mathrm{Corr}})$. The value of $\Delta\xi$ is then obtained from (6.82).

Next, we have to make sure that the updates in α, ξ do not cause the estimates to violate their positivity constraints. This is done by shrinking the length of $(\Delta x, \Delta\alpha, \Delta\xi)$ by some factor $\lambda \geq 0$, such that

Update in x, α

$$\min\left(\frac{\alpha_1 + \lambda\Delta\alpha_1}{\alpha_1}, \ldots, \frac{\alpha_n + \lambda\Delta\alpha_n}{\alpha_n}, \frac{\xi_1 + \lambda\Delta\xi_1}{\xi_1}, \ldots, \frac{\xi_n + \lambda\Delta\xi_n}{\xi_n}\right) \geq \epsilon. \tag{6.84}$$

Of course, only the negative Δ terms pose a problem, since they lead the parameter values closer to 0, which may lead them into conflict with the positivity constraints. Typically [542, 498], we choose $\epsilon = 0.05$. In other words, the solution will not approach the boundaries in α, ξ by more than 95%. See Problem 6.15 for a formula to compute λ.

6.4.3 Updating μ

Next we have to update μ. Here we face the following dilemma: if we decrease μ too quickly, we will get bad convergence of our second order method, since the solution to the problem (which depends on the value of μ) moves too quickly away from our current set of parameters (x, α, ξ). On the other hand, we do not want to spend too much time solving an *approximation* of the unrelaxed ($\mu = 0$) Kuhn-Tucker conditions *exactly*. A good indication is how much the positivity constraints would be violated by the current update. Vanderbei [542] proposes the following update of μ:

Tightening the KKT Conditions

$$\mu = \frac{\alpha^\top\xi}{n}\left(\frac{1-\lambda}{10+\lambda}\right)^2. \tag{6.85}$$

The first term gives the average value of satisfaction of the condition $\alpha_i\xi_i = \mu$ after an update step. The second term allows us to decrease μ rapidly if good progress was made (small $(1-\lambda)^2$). Experimental evidence shows that it pays to be slightly more conservative, and to use the *predictor* estimates of α, ξ for (6.85) rather than the corresponding corrector terms.[8] This imposes little overhead for the implementation.

6.4.4 Initial Conditions and Stopping Criterion

To provide a complete algorithm, we have to consider two more things: a stopping criterion and a suitable start value. For the latter, we simply solve a regularized version of the initial reduced KKT system (6.83). This means that we replace K

8. In practice it is often useful to replace $(1-\lambda)$ by $(1+\epsilon-\lambda)$ for some small $\epsilon > 0$, in order to avoid $\mu = 0$.

by $K + \mathbf{1}$, use (x, α) in place of $\Delta x, \Delta\alpha$, and replace D by the identity matrix. Moreover, ρ_p and ρ_d are set to the values they would have if all variables had been set to 0 before, and finally ρ_{KKT} is set to 0. In other words, we obtain an initial guess of (x, α, ξ) by solving

Regularized KKT System

$$\begin{bmatrix} K + \mathbf{1} & A^\top \\ A & -\mathbf{1} \end{bmatrix} \begin{bmatrix} x \\ \alpha \end{bmatrix} = \begin{bmatrix} -c \\ -d \end{bmatrix}, \tag{6.86}$$

and $\xi = -Ax - d$. Since we have to ensure positivity of α, ξ, we simply replace

$$\alpha_i = \max(\alpha_i, 1) \text{ and } \xi_i = \max(\xi_i, 1). \tag{6.87}$$

This heuristic solves the problem of a suitable initial condition.

Regarding the stopping criterion, we recall Theorem 6.27, and in particular (6.62). Rather than obtaining bounds on the precision of *parameters*, we want to make sure that $f(x)$ is close to its optimal value $f(\bar{x})$. From (6.64) we know, provided the feasibility constraints are all satisfied, that the value of the dual objective function is given by $f(x) + \sum_{i=1}^n \alpha_i c_i(x)$. We may use the latter to bound the *relative* size of the gap between primal and dual objective function by

$$\operatorname{Gap}(x, \alpha) = \frac{2\left|f(x) - \left(f(x) + \sum\limits_{i=1}^n \alpha_i c_i(x)\right)\right|}{|f(x)| + \left|\left(f(x) + \sum\limits_{i=1}^n \alpha_i c_i(x)\right)\right|} \leq \frac{-\sum\limits_{i=1}^n \alpha_i c_i(x)}{\left|f(x) + \frac{1}{2}\sum\limits_{i=1}^n \alpha_i c_i(x)\right|}. \tag{6.88}$$

For the special case where $f(x) = \frac{1}{2}x^\top K x + c^\top x$ as in (6.72), we know by virtue of (6.73) that the size of the feasibility gap is given by $\alpha^\top \xi$, and therefore

$$\operatorname{Gap}(x, \alpha) = \frac{\alpha^\top \xi}{\left|\frac{1}{2}x^\top K x + c^\top x + \frac{1}{2}\alpha^\top \xi\right|}. \tag{6.89}$$

In practice, a small number is usually added to the denominator of (6.89) in order to avoid divisions by 0 in the first iteration. The quality of the solution is typically measured on a logarithmic scale by $-\log_{10} \operatorname{Gap}(x, \alpha)$, the number of significant figures.[9] We will come back to specific versions of such interior point algorithms in Chapter 10, and show how Support Vector Regression and Classification problems can be solved with them.

Number of Significant Figures

Primal-Dual path following methods are certainly not the only algorithms that can be employed for minimizing constrained quadratic problems. Other variants, for instance, are Barrier Methods [266, 45, 545], which minimize the unconstrained problem

$$f(x) + \mu \sum_{i=1}^n \log\left(-c_i(x)\right) \text{ for } \mu > 0. \tag{6.90}$$

9. Interior point codes are very precise. They usually achieve up to 8 significant figures, whereas iterative approximation methods do not normally exceed more than 3 significant figures on large optimization problems.

Active set methods have also been used with success in machine learning [350, 268]. These select subsets of variables x for which the constraints c_i are not active, i.e., where the we have a strict inequality, and solve the resulting restricted quadratic program, for instance by conjugate gradient descent. We will encounter subset selection methods in Chapter 10.

6.5 Maximum Search Problems

Approximations

In several cases the task of finding an optimal function for estimation purposes means finding the best element from a finite set, or sometimes finding an optimal subset from a finite set of elements. These are discrete (sometimes combinatorial) optimization problems which are not so easily amenable to the techniques presented in the previous two sections. Furthermore, many commonly encountered problems are computationally expensive if solved exactly. Instead, by using probabilistic methods, it is possible to find *almost* optimal approximate solutions. These probabilistic methods are the topic of the present section.

6.5.1 Random Subset Selection

Consider the following problem: given a set of m functions, say $M := \{f_1, \ldots, f_m\}$, and some criterion $Q[f]$, find the function $\hat{f}$ that maximizes $Q[f]$. More formally,

$$\hat{f} := \operatorname*{argmax}_{f \in M} Q[f]. \tag{6.91}$$

Clearly, unless we have additional knowledge about the values $Q[f_i]$, we have to compute all terms $Q[f_i]$ if we want to solve (6.91) exactly. This will cost $O(m)$ operations. If m is large, which is often the case in practical applications, this operation is too expensive. In sparse greedy approximation problems (Section 10.2) or in Kernel Feature Analysis (Section 14.4), m can easily be of the order of 10^5 or larger (here, m is the number of training patterns). Hence we have to look for cheaper *approximate* solutions.

The key idea is to pick a random subset $M' \subset M$ that is sufficiently large, and take the maximum over M' as an approximation of the maximum over M. Provided the distribution of the values of $Q[f_i]$ is "well behaved", i.e., there exists not a small fraction of $Q[f_i]$ whose values are significantly smaller or larger than the average, we will obtain a solution that is close to the optimum with high probability. To formalize these ideas, we need the following result.

Lemma 6.30 (Maximum of Random Variables) *Denote by ξ, ξ' two independent random variables on $\mathbb{R}$ with corresponding distributions $\mathrm{P}_\xi, \mathrm{P}_{\xi'}$ and distribution functions $F_\xi, F_{\xi'}$. Then the random variable $\bar{\xi} := \max(\xi, \xi')$ has the distribution function $F_{\bar{\xi}} = F_\xi \, F_{\xi'}$.*

Proof Note that for a random variable, the distribution function $F(\xi_0)$ is given by the probability $\mathrm{P}\{\xi \leq \xi_0\}$. Since ξ and ξ' are independent, we may write

$$\begin{aligned} F_{\bar{\xi}}(\bar{\xi}) = \mathrm{P}\left\{\max(\xi,\xi') \leq \bar{\xi}\right\} = \mathrm{P}\left\{\xi \leq \bar{\xi} \text{ and } \xi' \leq \bar{\xi}\right\} &= \mathrm{P}\left\{\xi \leq \bar{\xi}\right\}\mathrm{P}\left\{\xi' \leq \bar{\xi}\right\} \\ &= F_{\xi}(\bar{\xi})F_{\xi'}(\bar{\xi}), \end{aligned} \tag{6.92}$$

which proves the claim. ■

Distribution Over $\bar{\xi}$ is More Peaked

Repeated application of Lemma 6.30 leads to the following corollary.

Corollary 6.31 (Maximum Over Identical Random Variables) *Denote by $\xi_1, \ldots, \xi_{\tilde{m}}$ a set of $\tilde{m}$ independent and identically distributed (iid) random variables, with corresponding distribution function F_ξ. Then the random variable $\bar{\xi} := \max(\xi_1, \ldots, \xi_{\tilde{m}})$ has the distribution function $F_{\bar{\xi}}(\bar{\xi}) = \left(F_\xi(\bar{\xi})\right)^{\tilde{m}}$.*

In practice, the random variables ξ_i will be the values of $Q[f_i]$, where the f_i are drawn from the set M. If we draw them without replacement (i.e. none of the functions f_i appears twice), however, the values after each draw are dependent and we cannot apply Corollary 6.31 directly. Nonetheless, we can see that the maximum over draws *without* replacement will be larger than the maximum *with* replacement, since recurring observations can be understood as reducing the effective size of the set to be considered. Thus Corollary 6.31 gives us a *lower bound* on the value of the distribution function for draws without replacement. Moreover, for large m the difference between draws with and without replacement is small.

If the distribution of $Q[f_i]$ is known, we may use the distribution directly to determine the size $\tilde{m}$ of a subset to be used to find some $Q[f_i]$ that is almost as good as the solution to (6.91). In all other cases, we have to resort to assessing the *relative* quality of maxima over subsets. The following theorem tells us how.

Best Element of a Subset

Theorem 6.32 (Ranks on Random Subsets) *Denote by $M := \{x_1, \ldots, x_m\} \subset \mathbb{R}$ a set of cardinality m, and by $\tilde{M} \subset M$ a random subset of size $\tilde{m}$. Then the probability that* $\max \tilde{M}$ *is greater equal than n elements of M is at least $1 - \left(\frac{n}{m}\right)^{\tilde{m}}$.*

Proof We prove this by assuming the converse, namely that $\max \tilde{M}$ is smaller than $(m-n)$ elements of M. For $\tilde{m} = 1$ we know that this probability is $\frac{n}{m}$, since there are n elements to choose from. For $\tilde{m} > 1$, the probability is the one of choosing $\tilde{m}$ elements out of a subset M_{low} of n elements, rather than all m elements. Therefore we have that

$$\mathrm{P}(\tilde{M} \subset M_{\text{low}}) = \frac{\binom{n}{\tilde{m}}}{\binom{m}{\tilde{m}}} = \frac{n}{m} \cdot \frac{n-1}{m-1} \cdot \ldots \cdot \frac{n-\tilde{m}+1}{m-\tilde{m}+1} < \left(\frac{n}{m}\right)^{\tilde{m}}.$$

Consequently the probability that the maximum over $\tilde{M}$ will be larger than n elements of M is given by $1 - \mathrm{P}(\tilde{M} \subset M_{\text{low}}) \geq 1 - \left(\frac{n}{m}\right)^{\tilde{m}}$. ■

The practical consequence is that we may use $1 - \left(\frac{n}{m}\right)^{\tilde{m}}$ to compute the required size of a random subset to achieve the desired degree of approximation. If we want to obtain results in the $\frac{n}{m}$ percentile range with $1-\eta$ confidence, we must solve for $\tilde{m} = \frac{\log(1-\eta)}{\log n/m}$. To give a numerical example, if we desire values that are better than 95% of all other estimates with $1 - 0.05$ probability, then $\kappa = 59$ samples are sufficient. This $(95\%, 95\%, 59)$ rule is very useful in practice.[10] A similar method was used to speed up the process of boosting classifiers in the MadaBoost algorithm [136]. Furthermore, one could think whether it might not be useful to recycle old observations rather than computing all 59 values from scratch. If this can be done cheaply, and under some additional independence assumptions, subset selection methods can be improved further. For details see [409] who use the method in the context of memory management for operating systems.

6.5.2 Random Evaluation

Quite often, the evaluation of the term $Q[f]$ itself is rather time consuming, especially if $Q[f]$ is the sum of many (m, for instance) iid random variables. Again, we can speed up matters considerably by using probabilistic methods. The key idea is that averages over independent random variables are concentrated, which is to say that averages over subsets do not differ too much from averages over the whole set.

Approximating Sums by Partial Sums

Hoeffding's Theorem (Section 5.2) quantifies the size of the deviations between the expectation of a sum of random variables and their values at individual trials. We will use this to bound deviations between averages over sets and subsets. All we have to do is translate Theorem 5.1 into a statement regarding sample averages over different sample sizes. This can be readily constructed as follows:

Corollary 6.33 (Maximum Deviation Bounds for Empirical Means [489]) *Let $\xi_1, \ldots, \xi_m$ be iid bounded random variables, falling into the interval $[a, a+b]$ with probability one. Denote their average by $Q_m = \frac{1}{m}\sum_i \xi_i$. Furthermore, denote by $\xi_{s(1)}, \ldots, \xi_{s(\tilde{m})}$ with $\tilde{m} < m$ a subset of the same random variables (with $s : \{1, \ldots, \tilde{m}\} \to \{1, \ldots, m\}$ being an injective map, i.e. $s(i) = s(j)$ only if $i = j$), and $Q_{\tilde{m}} = \frac{1}{\tilde{m}}\sum_i \xi_{s(i)}$. Then for any $\varepsilon > 0$,*

Deviation of Subsets

$$\left.\begin{array}{l} \mathrm{P}\{Q_m - Q_{\tilde{m}} \geq \varepsilon\} \\ \mathrm{P}\{Q_{\tilde{m}} - Q_m \geq \varepsilon\} \end{array}\right\} \leq \exp\left(-\frac{2m\tilde{m}\varepsilon^2}{(m-\tilde{m})b^2}\right) = \exp\left(-2m\frac{\varepsilon^2}{b^2}\frac{\frac{\tilde{m}}{m}}{1-\frac{\tilde{m}}{m}}\right) \tag{6.93}$$

Proof By construction $\mathbf{E}[Q_m - Q_{\tilde{m}}] = 0$, since Q_m and $Q_{\tilde{m}}$ are both averages over sums of random variables drawn from the same distribution. Hence we only have to rewrite $Q_m - Q_{\tilde{m}}$ as an average over (different) random variables to apply Hoeffding's bound. Since all Q_i are identically distributed, we may pick the first $\tilde{m}$ random variables, without loss of generality. In other words, we assume that

10. During World War I tanks were often numbered in continuous increasing order. Unfortunately this "feature" allowed the enemy to estimate the number of tanks. How?

$s(i) = i$ for $i = 1, \ldots, \tilde{m}$. Then

$$Q_m - Q_{\tilde{m}} = \tfrac{1}{m}\sum_{i=1}^{m}\xi_i - \tfrac{1}{\tilde{m}}\sum_{i=1}^{\tilde{m}}\xi_i = \tfrac{1}{m}\sum_{i=1}^{\tilde{m}}\left(1 - \tfrac{m}{\tilde{m}}\right)\xi_i + \tfrac{1}{m}\sum_{i=\tilde{m}+1}^{m}\xi_i. \tag{6.94}$$

Thus we may split up $Q_m - Q_{\tilde{m}}$ into a sum of $\tilde{m}$ random variables with range $b_i = (\frac{m}{\tilde{m}} - 1)b$, and $m - \tilde{m}$ random variables with range $b_i = b$. We obtain

$$\sum_{i=1}^{m} b_i^2 = b^2\tilde{m}\left(\frac{m}{\tilde{m}} - 1\right)^2 + (m - \tilde{m})b^2 = b^2(m - \tilde{m})\frac{m}{\tilde{m}}. \tag{6.95}$$

Substituting this into (5.8) and noting that $Q_m - Q_{\tilde{m}} - \mathbf{E}\left[Q_m - Q_{\tilde{m}}\right] = Q_m - Q_{\tilde{m}}$ completes the proof. ■

For small $\frac{\tilde{m}}{m}$ the RHS in (6.93) reduces to $\exp\left(-\frac{2\tilde{m}\varepsilon^2}{b^2}\right)$. In other words, deviations on the subsample $\tilde{m}$ dominate the overall deviation of $Q_m - Q_{\tilde{m}}$ from 0. This allows us to compute a cutoff criterion for evaluating Q_m by computing only a subset of its terms.

Cutoff Criterion

We need only solve (6.93) for $\frac{\tilde{m}}{m}$. Hence, in order to ensure that $Q_{\tilde{m}}$ is within ε of Q_m with probability $1 - \eta$, we have to take a fraction $\frac{\tilde{m}}{m}$ of samples that satisfies

$$\frac{\frac{\tilde{m}}{m}}{1 - \frac{\tilde{m}}{m}} = \frac{b^2(\log 2 - \log \eta)}{2m\varepsilon^2} =: c, \text{ and therefore } \frac{\tilde{m}}{m} = \frac{c}{1 + c}. \tag{6.96}$$

The fraction $\frac{\tilde{m}}{m}$ can be small for large m, which is exactly the case where we need methods to speed up evaluation.

6.5.3 Greedy Optimization Strategies

Quite often the overall goal is not necessarily to find the single best element x_i from a set X to solve a problem, but to find a good subset $\tilde{X} \subset X$ of size $\tilde{m}$ according to some quality criterion $Q[\tilde{X}]$. Problems of this type include approximating a matrix by a subset of its rows and columns (Section 10.2), finding approximate solutions to Kernel Fisher Discriminant Analysis (Chapter 15) and finding a sparse solution to the problem of Gaussian Process Regression (Section 16.3.4). These all have a common structure:

Applications

(i) Finding an optimal set $\tilde{X} \subset X$ is quite often a combinatorial problem, or it even may be NP-hard, since it means selecting $\tilde{m} = |\tilde{X}|$ elements from a set of $m = |X|$ elements. There are $\binom{m}{\tilde{m}}$ different choices, which clearly prevents an exhaustive search over all of them. Additionally, the size of $\tilde{m}$ is often not known beforehand. Hence we need a fast approximate algorithm.

(ii) The evaluation of $Q[\tilde{X} \cup \{x_i\}]$ is inexpensive, provided $Q[\tilde{X}]$ has been computed before. This indicates that an iterative algorithm can be useful.

(iii) The value of $Q[X]$, or equivalently how well we would do by taking the whole set X, can be bounded efficiently by using $Q[\tilde{X}]$ (or some by-products of the computation of $Q[\tilde{M}]$) without actually computing $Q[X]$.

(iv) The set of functions X is typically very large (i.e. more than 10^5 elements), yet the individual improvements by f_i via $Q[\tilde{X} \cup \{x_i\}]$ do not differ too much, meaning that specific $x_{\tilde{\imath}}$ for which $Q[\tilde{X} \cup \{x_{\tilde{\imath}}\}]$ deviate by a large amount from the rest of $Q[\tilde{X} \cup \{x_i\}]$ do not exist.

Iterative Enlargement of $\tilde{X}$

In this case we may use a sparse greedy algorithm to find close to optimal solutions among the remaining $X \backslash \tilde{X}$ elements. This combines the idea of an iterative enlargement of $\tilde{X}$ by one more element at a time (which is feasible since we can compute $Q[\tilde{X} \cup \{f_i\}]$ cheaply) with the idea that we need not consider all f_i as possible candidates for the enlargement. This uses the reasoning in Section 6.5.1 combined with the fact that the distribution of the improvements is not too long tailed (cf. (iv)). The overall strategy is described in Algorithm 6.6.

Algorithm 6.6 Sparse Greedy Algorithm

```
Require: Set of functions $X$, Precision $\epsilon$, Criterion $Q[\cdot]$
  Set $\tilde{X} = \emptyset$
  repeat
    Choose random subset $X'$ of size $m'$ from $X \backslash \tilde{X}$.
    Pick $\hat{x} = \mathrm{argmax}_{x \in X'} Q[X' \cup \{x\}]$
    $X' = X' \cup \{\hat{x}\}$
    If needed, (re)compute bound on $Q[X]$.
  until $Q[\tilde{X}] + \epsilon \geq$ Bound on $Q[X]$
Output: $\tilde{X}, Q[\tilde{X}]$
```

Problems 6.9 and 6.10 contain more examples of sparse greedy algorithms.

6.6 Summary

This chapter gave an overview of different optimization methods, which form the basic toolbox for solving the problems arising in learning with kernels. The main focus was on convex and differentiable problems, hence the overview of properties of convex sets and functions defined on them.

The key insights in Section 6.1 are that *convex sets* can be defined by *level sets of convex functions* and that convex optimization problems have *one global minimum.* Furthermore, the fact that the solutions of convex maximization over polyhedral sets can be found on the vertices will prove useful in some unsupervised learning applications (Section 14.4).

Basic tools for unconstrained problems (Section 6.2) include interval cutting methods, the Newton method, Conjugate Gradient descent, and Predictor-Corrector methods. These techniques are often used as building blocks to solve more advanced constrained optimization problems.

Since constrained minimization is a fairly complex topic, we only presented a selection of fundamental results, such as necessary and sufficient conditions in the general case of nonlinear programming. The Kuhn-Tucker conditions for differentiable convex functions then followed immediately from the previous reasoning. The main results are dualization, meaning the transformation of optimization problems via the Lagrange function mechanism into possibly simpler problems, and that optimality properties can be estimated via the KKT gap (Theorem 6.27).

Interior point algorithms are practical applications of the duality reasoning; these seek to find a solution to optimization problems by satisfying the Kuhn-Tucker optimality conditions. Here we were able to employ some of the concepts introduced at an earlier stage, such as predictor corrector methods and numerical ways of finding roots of equations. These algorithms are robust tools to find optimal solutions on moderately sized problems ($10^3 - 10^4$ examples). Larger problems require decomposition methods, to be discussed in Section 10.4, or randomized methods. The chapter concluded with an overview of randomized methods for maximizing functions or finding the best subset of elements. These techniques are useful once datasets are so large that we cannot reasonably hope to find exact solutions to optimization problems. The random subset selection strategy and sparse greedy methods are examples of such algorithms.

6.7 Problems

6.1 (Level Sets •) *Given the function $f : \mathbb{R}^2 \to \mathbb{R}$ with $f(x) := |x_1|^p + |x_2|^p$, for which p do we obtain a convex function?*

Now consider the sets $\{x | f(x) \leq c\}$ for some $c > 0$. Can you give an explicit parameterization of the boundary of the set? Is it easier to deal with this parameterization? Can you find other examples (see also [474] and Chapter 8 for details)?

6.2 (Convex Hulls •) *Show that for any set X, its convex hull $\operatorname{co} X$ is convex. Furthermore, show that $\operatorname{co} X = X$ if X is convex.*

6.3 (Method of False Position [315] •••) *Given a unimodal (posessing one minimum) differentiable function $f : \mathbb{R} \to \mathbb{R}$, develop a quadratic method for minimizing f.*

Hint: Recall the Newton method. There we used $f''(x)$ to make a quadratic approximation of f. Two values of $f'(x)$ are also sufficient to obtain this information, however.

What happens if we may only use f? What does the iteration scheme look like? See Figure 6.8 for a hint.

6.4 (Convex Minimization in one Variable ••) *Denote by f a convex function on $[a, b]$. Show that the algorithm below finds the minimum of f. What is the* rate *of convergence in x to $\operatorname{argmin}_x f(x)$? Can you obtain a bound in $f(x)$ wrt. $\min_x f(x)$?*

```
input  $a, b, f$ and threshold $\varepsilon$
```

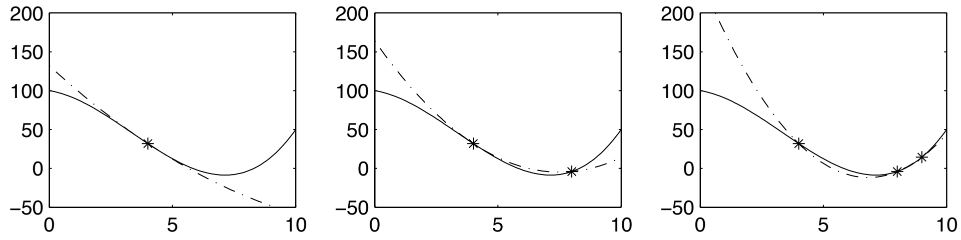


Figure 6.8 From left to right: Newton method, method of false position, quadratic interpolation through 3 points. Solid line: $f(x)$, dash-dotted line: interpolation.

```
$x_1 = a, x_2 = \frac{a+b}{2}, x_3 = b$ and compute $f(x_1), f(x_2), f(x_3)$
repeat
  if $x_3 - x_2 > x_2 - x_1$ then
    $x_4 = \frac{x_2+x_3}{2}$ and compute $f(x_4)$
  else
    $x_4 = \frac{x_1+x_2}{2}$ and compute $f(x_4)$
  end if
  Keep the two points closest to the point with the minimum value of $f(x_i)$ and
  rename them such that $x_1 < x_2 < x_3$.
until $x_3 - x_1 \geq \varepsilon$
```

6.5 (Newton Method in $\mathbb{R}^d$ ••) *Extend the Newton method to functions on $\mathbb{R}^d$. What does the iteration rule look like? Under which conditions does the algorithm converge? Do you have to extend Theorem 6.13 to prove convergence?*

6.6 (Rewriting Quadratic Functionals •) *Given a function*

$$f(x) = x^\top Q x + c^\top x + d, \tag{6.97}$$

rewrite it into the form of (6.18). Give explicit expressions for $x^* = \operatorname{argmin}_x f(x)$ and the difference in the additive constants.

6.7 (Kantorovich Inequality [262] •••) *Prove Theorem 6.16. Hint: note that without loss of generality we may require $\|x\|^2 = 1$. Second, perform a transformation of coordinates into the eigensystem of K. Finally, note that in the new coordinate system we are dealing with convex combinations of eigenvalues λ_i and $\frac{1}{\lambda_i}$. First show (6.24) for only two eigenvalues. Then argue that only the largest and smallest eigenvalues matter.*

6.8 (Random Subsets •) *Generate m random numbers drawn uniformly from the interval $[0, 1]$. Plot their distribution function. Plot the distribution of maxima of subsets of random numbers. What can you say about the distribution of the maxima? What happens if you draw randomly from the Laplace distribution, with density $p(\xi) = e^{-\xi}$ (for $\xi \geq 0$)?*

6.9 (Matching Pursuit [324] ••) *Denote by $f_1, \ldots, f_M$ a set of functions $\mathcal{X} \to \mathbb{R}$, by $\{x_1, \ldots, x_m\} \subset \mathcal{X}$ a set of locations and by $\{y_1, \ldots, y_m\} \subset \mathcal{Y}$ a set of corresponding observations.*

Design a sparse greedy algorithm that finds a linear combination of functions $f := \sum_i \alpha_i f_i$ minimizing the squared loss between $f(x_i)$ and y_i.

6.10 (Reduced Set Approximation [456] ••) *Let $f(x) = \sum_{i=1}^m \alpha_i k(x_i, x)$ be a kernel expansion in a Reproducing Kernel Hilbert Space $\mathcal{H}_k$ (see Section 2.2.4). Give a sparse greedy algorithm that finds an approximation to f in $\mathcal{H}_k$ by using fewer terms. See also Chapter 18 for more detail.*

6.11 (Equality Constraints in LP and QP ••) *Find the dual optimization problem and the necessary Kuhn-Tucker conditions for the following optimization problem:*

$$\begin{aligned} &\underset{x}{\text{minimize}} && c^\top x, \\ &\textit{subject to} && Ax + b \leq 0, \\ & && Cx + d = 0, \end{aligned} \tag{6.98}$$

where $c, x \in \mathbb{R}^m$, $b \in \mathbb{R}^n$, $d \in \mathbb{R}^{n'}$, $A \in R^{n \times m}$ and $C \in \mathbb{R}^{n'}$. Hint: split up the equality constraints into two inequality constraints. Note that you may combine the two Lagrange multipliers again to obtain a free variable. Derive the corresponding conditions for

$$\begin{aligned} &\underset{x}{\text{minimize}} && \tfrac{1}{2} x^\top K x + c^\top x, \\ &\textit{subject to} && Ax + b \leq 0, \\ & && Cx + d = 0, \end{aligned} \tag{6.99}$$

where K is a strictly positive definite matrix.

6.12 (Semidefinite Quadratic Parts •••) *How do you have to change the dual of (6.99) if K does not have full rank? Is it better not to dualize in this case? Do the Kuhn-Tucker conditions still hold?*

6.13 (Dual Problems of Quadratic Programs ••) *Denote by P a quadratic optimization problem of type (6.72) and by $(\cdot)^D$ the dualization operation. Prove that the following is true,*

$$((P^D)^D)^D = P^D \text{ and } (((P^D)^D)^D)^D = (P^D)^D, \tag{6.100}$$

where in general $(P^D)^D \neq P$. Hint: use (6.80). Caution: you have to check whether $KA^\top$ has full rank.

6.14 (Interior Point Equations for Linear Programs [318] •••) *Derive the interior point equations for linear programs. Hint: use the expansions for the quadratic programs and note that the reduced KKT system has only a diagonal term where we had K before.*

How does the complexity of the problem scale with the size of A?

6.15 (Update Step in Interior Point Codes •) *Show that the maximum value of λ satisfying (6.84) can be found by*

$$\frac{1}{\lambda} = \max\left(1, (\epsilon - 1)^{-1} \min_{i \in [n]} \frac{\Delta \alpha_i}{\alpha_i}, (\epsilon - 1)^{-1} \min_{i \in [n]} \frac{\Delta \xi_i}{\xi_i}\right). \tag{6.101}$$