5

Optimization

Optimization plays an increasingly important role in machine learning. For instance, many machine learning algorithms minimize a regularized risk functional:

$$\min_f \; J(f) := \lambda\Omega(f) + R_{\text{emp}}(f), \tag{5.1}$$

with the empirical risk

$$R_{\text{emp}}(f) := \frac{1}{m}\sum_{i=1}^{m} l(f(x_i), y_i). \tag{5.2}$$

Here x_i are the training instances and y_i are the corresponding labels. l the loss function measures the discrepancy between y and the predictions $f(x_i)$. Finding the optimal f involves solving an optimization problem.

This chapter provides a self-contained overview of some basic concepts and tools from optimization, especially geared towards solving machine learning problems. In terms of concepts, we will cover topics related to convexity, duality, and Lagrange multipliers. In terms of tools, we will cover a variety of optimization algorithms including gradient descent, stochastic gradient descent, Newton's method, and Quasi-Newton methods. We will also look at some specialized algorithms tailored towards solving Linear Programming and Quadratic Programming problems which often arise in machine learning problems.

5.1 Preliminaries

Minimizing an arbitrary function is, in general, very difficult, but if the objective function to be minimized is convex then things become considerably simpler. As we will see shortly, the key advantage of dealing with convex functions is that a local optima is also the global optima. Therefore, well developed tools exist to find the global minima of a convex function. Consequently, many machine learning algorithms are now formulated in terms of convex optimization problems. We briefly review the concept of convex sets and functions in this section.

5.1.1 Convex Sets

Definition 5.1 (Convex Set) *A subset C of $\mathbb{R}^n$ is said to be convex if $(1-\lambda)x + \lambda y \in C$ whenever $x \in C, y \in C$ and $0 < \lambda < 1$.*

Intuitively, what this means is that the line joining any two points x and y from the set C lies inside C (see Figure 5.1). It is easy to see (Exercise 5.1) that intersections of convex sets are also convex.

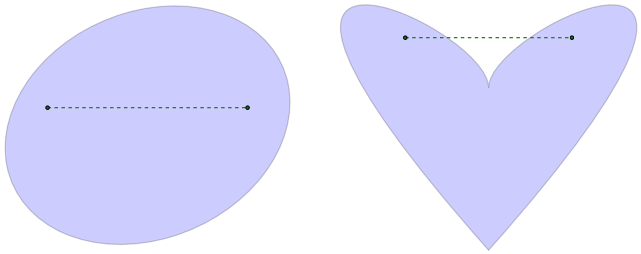

Fig. 5.1. The convex set (left) contains the line joining any two points that belong to the set. A non-convex set (right) does not satisfy this property.

A vector sum $\sum_i \lambda_i x_i$ is called a convex combination if $\lambda_i \geq 0$ and $\sum_i \lambda_i = 1$. Convex combinations are helpful in defining a convex hull:

Definition 5.2 (Convex Hull) *The convex hull, $\mathrm{conv}(X)$, of a finite subset $X = \{x_1, \ldots, x_n\}$ of $\mathbb{R}^n$ consists of all convex combinations of $x_1, \ldots, x_n$.*

5.1.2 Convex Functions

Let f be a real valued function defined on a set $X \subset \mathbb{R}^n$. The set

$$\{(x, \mu) : x \in X, \mu \in \mathbb{R}, \mu \geq f(x)\} \tag{5.3}$$

is called the *epigraph* of f. The function f is defined to be a convex function if its epigraph is a convex set in $\mathbb{R}^{n+1}$. An equivalent, and more commonly used, definition (Exercise 5.5) is as follows (see Figure 5.2 for geometric intuition):

Definition 5.3 (Convex Function) *A function f defined on a set X is called convex if, for any $x, x' \in X$ and any $0 < \lambda < 1$ such that $\lambda x + (1-\lambda)x' \in X$, we have*

$$f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda) f(x'). \tag{5.4}$$

A function f is called strictly *convex if*

$$f(\lambda x + (1-\lambda)x') < \lambda f(x) + (1-\lambda)f(x') \tag{5.5}$$

whenever $x \neq x'$.

In fact, the above definition can be extended to show that if f is a convex function and $\lambda_i \geq 0$ with $\sum_i \lambda_i = 1$ then

$$f\left(\sum_i \lambda_i x_i\right) \leq \sum_i \lambda_i f(x_i). \tag{5.6}$$

The above inequality is called the *Jensen's inequality* (problem).

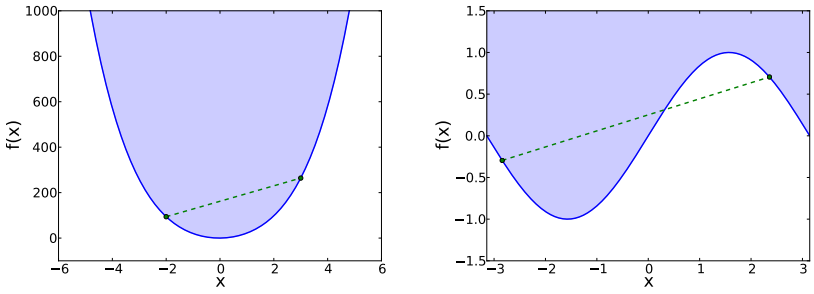

Fig. 5.2. A convex function (left) satisfies (5.4); the shaded region denotes its epigraph. A nonconvex function (right) does not satisfy (5.4).

If $f : X \to \mathbb{R}$ is differentiable, then f is convex if, and only if,

$$f(x') \geq f(x) + \langle x' - x, \nabla f(x) \rangle \text{ for all } x, x' \in X. \tag{5.7}$$

In other words, the first order Taylor approximation lower bounds the convex function universally (see Figure 5.4). Here and in the rest of the chapter $\langle x, y \rangle$ denotes the Euclidean dot product between vectors x and y, that is,

$$\langle x, y \rangle := \sum_i x_i y_i. \tag{5.8}$$

If f is twice differentiable, then f is convex if, and only if, its Hessian is positive semi-definite, that is,

$$\nabla^2 f(x) \succeq 0. \tag{5.9}$$

For twice differentiable strictly convex functions, the Hessian matrix is positive definite, that is, $\nabla^2 f(x) \succ 0$. We briefly summarize some operations which preserve convexity:

Addition	If f_1 and f_2 are convex, then $f_1 + f_2$ is also convex.
Scaling	If f is convex, then αf is convex for $\alpha > 0$.
Affine Transform	If f is convex, then $g(x) = f(Ax + b)$ for some matrix A and vector b is also convex.
Adding a Linear Function	If f is convex, then $g(x) = f(x) + \langle a, x \rangle$ for some vector a is also convex.
Subtracting a Linear Function	If f is convex, then $g(x) = f(x) - \langle a, x \rangle$ for some vector a is also convex.
Pointwise Maximum	If f_i are convex, then $g(x) = \max_i f_i(x)$ is also convex.
Scalar Composition	If $f(x) = h(g(x))$, then f is convex if a) g is convex, and h is convex, non-decreasing or b) g is concave, and h is convex, non-increasing.

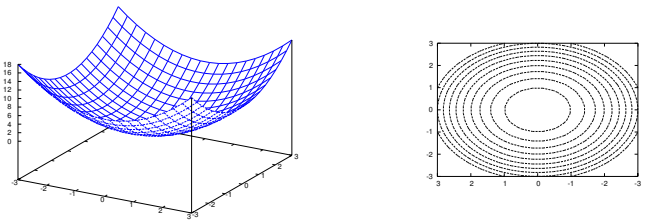

Fig. 5.3. Left: Convex Function in two variables. Right: the corresponding convex below-sets $\{x | f(x) \leq c\}$, for different values of c. This is also called a contour plot.

There is an intimate relation between convex functions and convex sets. For instance, the following lemma show that the *below sets* (level sets) of convex functions, sets for which $f(x) \leq c$, are convex.

Lemma 5.4 (Below-Sets of Convex Functions) *Denote by $f : X \to \mathbb{R}$ a convex function. Then the set*

$$X_c := \{x \,|\, x \in X \text{ and } f(x) \leq c\}, \text{ for all } c \in \mathbb{R}, \tag{5.10}$$

is convex.

Proof For any $x, x' \in X_c$, we have $f(x), f(x') \leq c$. Moreover, since f is convex, we also have

$$f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda) f(x') \leq c \text{ for all } 0 < \lambda < 1. \tag{5.11}$$

Hence, for all $0 < \lambda < 1$, we have $(\lambda x + (1-\lambda)x') \in X_c$, which proves the claim. Figure 5.3 depicts this situation graphically. ■

As we hinted in the introduction of this chapter, 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. In contrast, minimization of a convex objective function on a convex set exhibits exactly one *global* minimum. We now prove this property.

Theorem 5.5 (Minima on Convex Sets) *If the convex function $f : X \to \mathbb{R}$ attains its minimum, then the set of $x \in X$, for which the minimum value is attained, is a convex set. Moreover, if f is strictly convex, then this set contains a single element.*

Proof Denote by c the minimum of f on X. Then the set $X_c := \{x | x \in X \text{ and } f(x) \leq c\}$ is clearly convex.

If f is strictly convex, then for any two distinct $x, x' \in X_c$ and any $0 < \lambda < 1$ we have

$$f(\lambda x + (1-\lambda)x') < \lambda f(x) + (1-\lambda) f(x') = \lambda c + (1-\lambda)c = c,$$

which contradicts the assumption that f attains its minimum on X_c. Therefore X_c must contain only a single element. ■

As the following lemma shows, the minimum point can be characterized precisely.

Lemma 5.6 *Let $f : X \to \mathbb{R}$ be a differentiable convex function. Then x is a minimizer of f, if, and only if,*

$$\langle x' - x, \nabla f(x) \rangle \geq 0 \text{ for all } x'. \tag{5.12}$$

Proof To show the forward implication, suppose that x is the optimum but (5.12) does not hold, that is, there exists an x' for which

$$\langle x' - x, \nabla f(x) \rangle < 0.$$

Consider the line segment $z(\lambda) = (1-\lambda)x + \lambda x'$, with $0 < \lambda < 1$. Since X is convex, $z(\lambda)$ lies in X. On the other hand,

$$\frac{d}{d\lambda} f(z(\lambda))\Big|_{\lambda=0} = \langle x' - x, \nabla f(x) \rangle < 0,$$

which shows that for small values of λ we have $f(z(\lambda)) < f(x)$, thus showing that x is not optimal.

The reverse implication follows from (5.7) by noting that $f(x') \geq f(x)$, whenever (5.12) holds.

■

One way to ensure that (5.12) holds is to set $\nabla f(x) = 0$. In other words, minimizing a convex function is equivalent to finding a x such that $\nabla f(x) = 0$. Therefore, the first order conditions are both necessary and sufficient when minimizing a convex function.

5.1.3 Subgradients

So far, we worked with differentiable convex functions. The subgradient is a generalization of gradients appropriate for convex functions, including those which are not necessarily smooth.

Definition 5.7 (Subgradient) *Suppose x is a point where a convex function f is finite. Then a subgradient is the normal vector of any tangential supporting hyperplane of f at x. Formally μ is called a subgradient of f at x if, and only if,*

$$f(x') \geq f(x) + \langle x' - x, \mu \rangle \text{ for all } x'. \tag{5.13}$$

The set of all subgradients at a point is called the subdifferential, and is denoted by $\partial_x f(x)$. If this set is not empty then f is said to be *subdifferentiable at x*. On the other hand, if this set is a singleton then, the function is said to be *differentiable* at x. In this case we use $\nabla f(x)$ to denote the gradient of f. Convex functions are subdifferentiable everywhere in their domain. We now state some simple rules of subgradient calculus:

Addition	$\partial_x(f_1(x) + f_2(x)) = \partial_x f_1(x) + \partial_x f_2(x)$
Scaling	$\partial_x \alpha f(x) = \alpha \partial_x f(x)$, for $\alpha > 0$
Affine Transform	If $g(x) = f(Ax + b)$ for some matrix A and vector b, then $\partial_x g(x) = A^\top \partial_y f(y)$.
Pointwise Maximum	If $g(x) = \max_i f_i(x)$ then $\partial g(x) = \text{conv}(\partial_x f_{i'})$ where $i' \in \text{argmax}_i f_i(x)$.

The definition of a subgradient can also be understood geometrically. As illustrated by Figure 5.4, a differentiable convex function is always lower bounded by its first order Taylor approximation. This concept can be extended to non-smooth functions via subgradients, as Figure 5.5 shows.

By using more involved concepts, the proof of Lemma 5.6 can be extended to subgradients. In this case, minimizing a convex nonsmooth function entails finding a x such that $0 \in \partial f(x)$.

5.1.4 Strongly Convex Functions

When analyzing optimization algorithms, it is sometimes easier to work with strongly convex functions, which generalize the definition of convexity.

Definition 5.8 (Strongly Convex Function) *A convex function f is σ-strongly convex if, and only if, there exists a constant $\sigma > 0$ such that the function $f(x) - \frac{\sigma}{2}\|x\|^2$ is convex.*

The constant σ is called the modulus of strong convexity of f. If f is twice differentiable, then there is an equivalent, and perhaps easier, definition of strong convexity: f is strongly convex if there exists a σ such that

$$\nabla^2 f(x) \succeq \sigma I. \tag{5.14}$$

In other words, the smallest eigenvalue of the Hessian of f is *uniformly lower bounded* by σ everywhere. Some important examples of strongly convex functions include:

Example 5.1 (Squared Euclidean Norm) *The function $f(x) = \frac{\lambda}{2}\|x\|^2$ is λ-strongly convex.*

Example 5.2 (Negative Entropy) *Let $\Delta^n = \{x \text{ s.t. } \sum_i x_i = 1 \text{ and } x_i \geq 0\}$ be the n dimensional simplex, and $f : \Delta^n \to \mathbb{R}$ be the negative entropy:*

$$f(x) = \sum_i x_i \log x_i. \tag{5.15}$$

Then f is 1-strongly convex with respect to the $\|\cdot\|_1$ norm on the simplex (see Problem 5.7).

If f is a σ-strongly convex function then one can show the following properties (Exercise 5.8). Here x, x' are arbitrary and $\mu \in \partial f(x)$ and $\mu' \in \partial f(x')$.

$$f(x') \geq f(x) + \langle x' - x, \mu \rangle + \frac{\sigma}{2}\|x' - x\|^2 \tag{5.16}$$

$$f(x') \leq f(x) + \langle x' - x, \mu \rangle + \frac{1}{2\sigma}\|\mu' - \mu\|^2 \tag{5.17}$$

$$\langle x - x', \mu - \mu' \rangle \geq \sigma\|x - x'\|^2 \tag{5.18}$$

$$\langle x - x', \mu - \mu' \rangle \leq \frac{1}{\sigma}\|\mu - \mu'\|^2. \tag{5.19}$$

5.1.5 Convex Functions with Lipschitz Continous Gradient

A somewhat symmetric concept to strong convexity is the Lipschitz continuity of the gradient. As we will see later they are connected by Fenchel duality.

Definition 5.9 (Lipschitz Continuous Gradient) *A differentiable convex function f is said to have a Lipschitz continuous gradient, if there exists a constant $L > 0$, such that*

$$\left\|\nabla f(x) - \nabla f(x')\right\| \leq L \left\|x - x'\right\| \quad \forall x, x'. \tag{5.20}$$

As before, if f is twice differentiable, then there is an equivalent, and perhaps easier, definition of Lipschitz continuity of the gradient: f has a Lipschitz continuous gradient strongly convex if there exists a L such that

$$LI \succeq \nabla^2 f(x). \tag{5.21}$$

In other words, the largest eigenvalue of the Hessian of f is *uniformly upper bounded* by L everywhere. If f has a Lipschitz continuous gradient with modulus L, then one can show the following properties (Exercise 5.9).

$$f(x') \leq f(x) + \left\langle x' - x, \nabla f(x)\right\rangle + \frac{L}{2}\left\|x - x'\right\|^2 \tag{5.22}$$

$$f(x') \geq f(x) + \left\langle x' - x, \nabla f(x)\right\rangle + \frac{1}{2L}\left\|\nabla f(x) - \nabla f(x')\right\|^2 \tag{5.23}$$

$$\left\langle x - x', \nabla f(x) - \nabla f(x')\right\rangle \leq L\left\|x - x'\right\|^2 \tag{5.24}$$

$$\left\langle x - x', \nabla f(x) - \nabla f(x')\right\rangle \geq \frac{1}{L}\left\|\nabla f(x) - \nabla f(x')\right\|^2. \tag{5.25}$$

5.1.6 Fenchel Duality

The Fenchel conjugate of a function f is given by

$$f^*(x^*) = \sup_x \left\{\langle x, x^*\rangle - f(x)\right\}. \tag{5.26}$$

Even if f is not convex, the Fechel conjugate which is written as a supremum over linear functions is always convex. Some rules for computing Fenchel duals are summarized in Table 5.1.6. If f is convex and its epigraph (5.3) is a closed convex set, then $f^{**} = f$. If f and f^* are convex, then they satisfy the so-called Fenchel-Young inequality

$$f(x) + f^*(x^*) \geq \langle x, x^*\rangle \text{ for all } x, x^*. \tag{5.27}$$

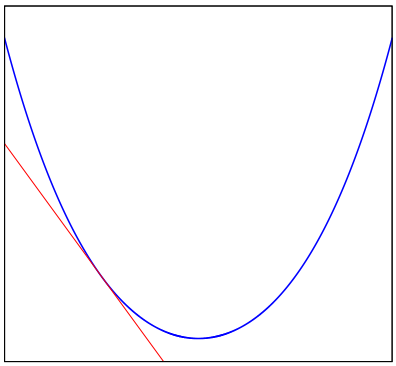

Fig. 5.4. A convex function is always lower bounded by its first order Taylor approximation. This is true even if the function is not differentiable (see Figure 5.5)

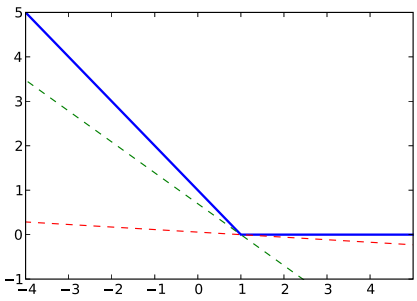

Fig. 5.5. Geometric intuition of a subgradient. The nonsmooth convex function (solid blue) is only subdifferentiable at the "kink" points. We illustrate two of its subgradients (dashed green and red lines) at a "kink" point which are tangential to the function. The normal vectors to these lines are subgradients. Observe that the first order Taylor approximations obtained by using the subgradients lower bounds the convex function.

This inequality becomes an equality whenever $x^* \in \partial f(x)$, that is,

$$f(x) + f^*(x^*) = \langle x, x^* \rangle \text{ for all } x \text{ and } x^* \in \partial f(x). \tag{5.28}$$

Strong convexity (Section 5.1.4) and Lipschitz continuity of the gradient

Table 5.1. *Rules for computing Fenchel Duals*

Scalar Addition	If $g(x) = f(x) + \alpha$ then $g^*(x^*) = f^*(x^*) - \alpha$.
Function Scaling	If $\alpha > 0$ and $g(x) = \alpha f(x)$ then $g^*(x^*) = \alpha f^*(x^*/\alpha)$.
Parameter Scaling	If $\alpha \neq 0$ and $g(x) = f(\alpha x)$ then $g^*(x^*) = f^*(x^*/\alpha)$
Linear Transformation	If A is an invertible matrix then $(f \circ A)^* = f^* \circ (A^{-1})^*$.
Shift	If $g(x) = f(x - x_0)$ then $g^*(x^*) = f^*(x^*) + \langle x^*, x_0 \rangle$.
Sum	If $g(x) = f_1(x) + f_2(x)$ then $g^*(x^*) = \inf \{f_1^*(x_1^*) + f_2^*(x_2^*)$ s.t. $x_1^* + x_2^* = x^*\}$.
Pointwise Infimum	If $g(x) = \inf f_i(x)$ then $g^*(x^*) = \sup_i f_i^*(x^*)$.

(Section 5.1.5) are related by Fenchel duality according to the following lemma, which we state without proof.

Lemma 5.10 (Theorem 4.2.1 and 4.2.2 [HUL93])

(i) *If f is σ-strongly convex, then f^* has a Lipschitz continuous gradient with modulus $\frac{1}{\sigma}$.*

(ii) *If f is convex and has a Lipschitz continuous gradient with modulus L, then f^* is $\frac{1}{L}$-strongly convex.*

Next we describe some convex functions and their Fenchel conjugates.

Example 5.3 (Squared Euclidean Norm) *Whenever $f(x) = \frac{1}{2}\|x\|^2$ we have $f^*(x^*) = \frac{1}{2}\|x^*\|^2$, that is, the squared Euclidean norm is its own conjugate.*

Example 5.4 (Negative Entropy) *The Fenchel conjugate of the negative entropy (5.15) is*

$$f^*(x^*) = \log \sum_i \exp(x_i^*).$$

5.1.7 Bregman Divergence

Let f be a differentiable convex function. The Bregman divergence defined by f is given by

$$\Delta_f(x, x') = f(x) - f(x') - \langle x - x', \nabla f(x') \rangle . \tag{5.29}$$

Also see Figure 5.6. Here are some well known examples.

Example 5.5 (Square Euclidean Norm) *Set $f(x) = \frac{1}{2}\|x\|^2$. Clearly, $\nabla f(x) = x$ and therefore*

$$\Delta_f(x, x') = \frac{1}{2}\|x\|^2 - \frac{1}{2}\|x'\|^2 - \langle x - x', x' \rangle = \frac{1}{2}\|x - x'\|^2 .$$

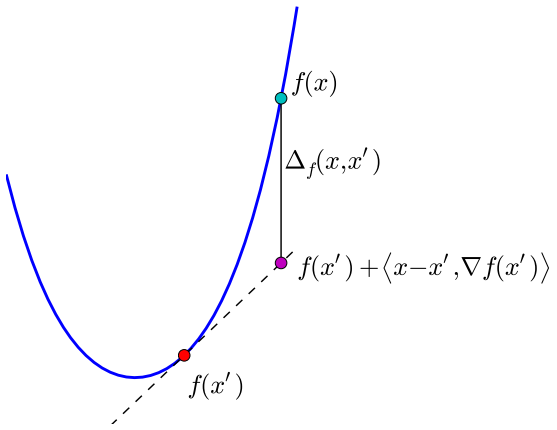

Fig. 5.6. $f(x)$ is the value of the function at x, while $f(x') + \langle x - x', \nabla f(x') \rangle$ denotes the first order Taylor expansion of f around x', evaluated at x. The difference between these two quantities is the Bregman divergence, as illustrated.

Example 5.6 (Relative Entropy) *Let f be the un-normalized entropy*

$$f(x) = \sum_i (x_i \log x_i - x_i) \,. \tag{5.30}$$

One can calculate $\nabla f(x) = \log x$, where $\log x$ is the component wise logarithm of the entries of x, and write the Bregman divergence

$$\begin{aligned} \Delta_f(x, x') &= \sum_i x_i \log x_i - \sum_i x_i - \sum_i x'_i \log x'_i + \sum_i x'_i - \langle x - x', \log x' \rangle \\ &= \sum_i \left(x_i \log \left(\frac{x_i}{x'_i} \right) + x'_i - x_i \right) . \end{aligned}$$

Example 5.7 (p-norm) *Let f be the square p-norm*

$$f(x) = \frac{1}{2} \|x\|_p^2 = \frac{1}{2} \left(\sum_i x_i^p \right)^{2/p} . \tag{5.31}$$

We say that the q-norm is dual to the p-norm whenever $\frac{1}{p}+\frac{1}{q}=1$. *One can verify (Problem 5.12) that the i-th component of the gradient* $\nabla f(x)$ *is*

$$\nabla_{x_i} f(x) = \frac{\operatorname{sign}(x_i)\,|x_i|^{p-1}}{\|x\|_p^{p-2}}. \tag{5.32}$$

The corresponding Bregman divergence is

$$\Delta_f(x,x') = \frac{1}{2}\|x\|_p^2 - \frac{1}{2}\|x'\|_p^2 - \sum_i (x_i - x_i')\frac{\operatorname{sign}(x_i')\,|x_i'|^{p-1}}{\|x'\|_p^{p-2}}.$$

The following properties of the Bregman divergence immediately follow:

- $\Delta_f(x,x')$ is convex in x.
- $\Delta_f(x,x') \geq 0$.
- Δ_f may not be symmetric, that is, in general $\Delta_f(x,x') \neq \Delta_f(x',x)$.
- $\nabla_x \Delta_f(x,x') = \nabla f(x) - \nabla f(x')$.

The next lemma establishes another important property.

Lemma 5.11 *The Bregman divergence (5.29) defined by a differentiable convex function* f *satisfies*

$$\Delta_f(x,y) + \Delta_f(y,z) - \Delta_f(x,z) = \langle \nabla f(z) - \nabla f(y), x - y\rangle\,. \tag{5.33}$$

Proof

$$\begin{aligned}\Delta_f(x,y) + \Delta_f(y,z) &= f(x) - f(y) - \langle x-y, \nabla f(y)\rangle + f(y) - f(z) - \langle y - z, \nabla f(z)\rangle \\ &= f(x) - f(z) - \langle x - y, \nabla f(y)\rangle - \langle y - z, \nabla f(z)\rangle \\ &= \Delta_f(x,z) + \langle \nabla f(z) - \nabla f(y), x - y\rangle\,.\end{aligned}$$

■

5.2 Unconstrained Smooth Convex Minimization

In this section we will describe various methods to minimize a smooth convex objective function.

5.2.1 Minimizing a One-Dimensional Convex Function

As a warm up let us consider the problem of minimizing a smooth one dimensional convex function $J : \mathbb{R} \to \mathbb{R}$ in the interval $[L, U]$. This seemingly

Algorithm 5.1 Interval Bisection

Input: L, U, precision ϵ
Set $t = 0$, $a_0 = L$ and $b_0 = U$
while $(b_t - a_t) \cdot J'(U) > \epsilon$ **do**
  if $J'(\frac{a_t+b_t}{2}) > 0$ **then**
    $a_{t+1} = a_t$ and $b_{t+1} = \frac{a_t+b_t}{2}$
  else
    $a_{t+1} = \frac{a_t+b_t}{2}$ and $b_{t+1} = b_t$
  end if
  $t = t + 1$
end while
Return: $\frac{a_t+b_t}{2}$

simple problem has many applications. As we will see later, many optimization methods find a direction of descent and minimize the objective function along this direction[1]; this subroutine is called a line search. Algorithm 5.1 depicts a simple line search routine based on interval bisection.

Before we show that Algorithm 5.1 converges, let us first derive an important property of convex functions of one variable. For a differentiable one-dimensional convex function J (5.7) reduces to

$$J(w) \geq J(w') + (w - w') \cdot J'(w'), \tag{5.34}$$

where $J'(w)$ denotes the gradient of J. Exchanging the role of w and w' in (5.34), we can write

$$J(w') \geq J(w) + (w' - w) \cdot J'(w). \tag{5.35}$$

Adding the above two equations yields

$$(w - w') \cdot (J'(w) - J'(w')) \geq 0. \tag{5.36}$$

If $w \geq w'$, then this implies that $J'(w) \geq J'(w')$. In other words, the gradient of a one dimensional convex function is monotonically non-decreasing.

Recall that minimizing a convex function is equivalent to finding w^* such that $J'(w^*) = 0$. Furthermore, it is easy to see that the interval bisection maintains the invariant $J'(a_t) < 0$ and $J'(b_t) > 0$. This along with the monotonicity of the gradient suffices to ensure that $w^* \in (a_t, b_t)$. Setting $w = w^*$ in (5.34), and using the monotonicity of the gradient allows us to

[1] If the objective function is convex, then the one dimensional function obtained by restricting it along the search direction is also convex (Exercise 5.10).

write for any $w' \in (a_t, b_t)$

$$J(w') - J(w^*) \leq (w' - w^*) \cdot J'(w') \leq (b_t - a_t) \cdot J'(U). \tag{5.37}$$

Since we halve the interval (a_t, b_t) at every iteration, it follows that $(b_t - a_t) = (U - L)/2^t$. Therefore

$$J(w') - J(w^*) \leq \frac{(U - L) \cdot J'(U)}{2^t}, \tag{5.38}$$

for all $w' \in (a_t, b_t)$. In other words, to find an ϵ-accurate solution, that is, $J(w') - J(w^*) \leq \epsilon$ we only need $\log(U - L) + \log J'(U) + \log(1/\epsilon) < t$ iterations. An algorithm which converges to an ϵ accurate solution in $O(\log(1/\epsilon))$ iterations is said to be linearly convergent.

For multi-dimensional objective functions, one cannot rely on the monotonicity property of the gradient. Therefore, one needs more sophisticated optimization algorithms, some of which we now describe.

5.2.2 Coordinate Descent

Coordinate descent is conceptually the simplest algorithm for minimizing a multidimensional smooth convex function $J : \mathbb{R}^n \to \mathbb{R}$. At every iteration select a coordinate, say i, and update

$$w_{t+1} = w_t - \eta_t e_i. \tag{5.39}$$

Here e_i denotes the i-th basis vector, that is, a vector with one at the i-th coordinate and zeros everywhere else, while $\eta_t \in \mathbb{R}$ is a non-negative scalar step size. One could, for instance, minimize the one dimensional convex function $J(w_t - \eta e_i)$ to obtain the stepsize η_t. The coordinates can either be selected cyclically, that is, $1, 2, \ldots, n, 1, 2, \ldots$ or greedily, that is, the coordinate which yields the maximum reduction in function value.

Even though coordinate descent can be shown to converge if J has a Lipschitz continuous gradient [LT92], in practice it can be quite slow. However, if a high precision solution is not required, as is the case in some machine learning applications, coordinate descent is often used because a) the cost per iteration is very low and b) the speed of convergence may be acceptable especially if the variables are loosely coupled.

5.2.3 Gradient Descent

Gradient descent (also widely known as steepest descent) is an optimization technique for minimizing multidimensional smooth convex objective functions of the form $J : \mathbb{R}^n \to \mathbb{R}$. The basic idea is as follows: Given a location

w_t at iteration t, compute the gradient $\nabla J(w_t)$, and update

$$w_{t+1} = w_t - \eta_t \nabla J(w_t), \tag{5.40}$$

where η_t is a scalar stepsize. See Algorithm 5.2 for details. Different variants of gradient descent depend on how η_t is chosen:

Exact Line Search: Since $J(w_t - \eta \nabla J(w_t))$ is a one dimensional convex function in η, one can use the Algorithm 5.1 to compute:

$$\eta_t = \underset{\eta}{\operatorname{argmin}}\, J(w_t - \eta \nabla J(w_t)). \tag{5.41}$$

Instead of the simple bisecting line search more sophisticated line searches such as the More-Thuente line search or the golden bisection rule can also be used to speed up convergence (see [NW99] Chapter 3 for an extensive discussion).

Inexact Line Search: Instead of minimizing $J(w_t - \eta \nabla J(w_t))$ we could simply look for a stepsize which results in sufficient decrease in the objective function value. One popular set of sufficient decrease conditions is the Wolfe conditions

$$J(w_{t+1}) \leq J(w_t) + c_1 \eta_t \langle \nabla J(w_t), w_{t+1} - w_t \rangle \text{ (sufficient decrease)} \tag{5.42}$$

$$\langle \nabla J(w_{t+1}), w_{t+1} - w_t \rangle \geq c_2 \langle \nabla J(w_t), w_{t+1} - w_t \rangle \text{ (curvature)} \tag{5.43}$$

with $0 < c_1 < c_2 < 1$ (see Figure 5.7). The Wolfe conditions are also called the Armijio-Goldstein conditions. If only sufficient decrease (5.42) alone is enforced, then it is called the Armijio rule.

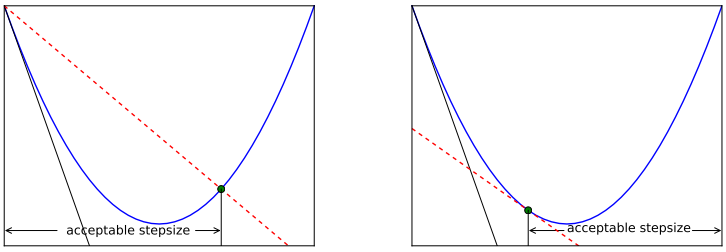

Fig. 5.7. The sufficient decrease condition (left) places an upper bound on the acceptable stepsizes while the curvature condition (right) places a lower bound on the acceptable stepsizes.

Algorithm 5.2 Gradient Descent

Input: Initial point w_0, gradient norm tolerance ϵ
Set $t = 0$
while $\|\nabla J(w_t)\| \geq \epsilon$ **do**
  $w_{t+1} = w_t - \eta_t \nabla J(w_t)$
  $t = t + 1$
end while
Return: w_t

Decaying Stepsize: Instead of performing a line search at every iteration, one can use a stepsize which decays according to a fixed schedule, for example, $\eta_t = 1/\sqrt{t}$. In Section 5.2.4 we will discuss the decay schedule and convergence rates of a generalized version of gradient descent.

Fixed Stepsize: Suppose J has a Lipschitz continuous gradient with modulus L. Using (5.22) and the gradient descent update $w_{t+1} = w_t - \eta_t \nabla J(w_t)$ one can write

$$J(w_{t+1}) \leq J(w_t) + \langle \nabla J(w_t), w_{t+1} - w_t \rangle + \frac{L}{2} \|w_{t+1} - w_t\| \tag{5.44}$$

$$= J(w_t) - \eta_t \|\nabla J(w_t)\|^2 + \frac{L\eta_t^2}{2} \|\nabla J(w_t)\|^2 . \tag{5.45}$$

Minimizing (5.45) as a function of η_t clearly shows that the upper bound on $J(w_{t+1})$ is minimized when we set $\eta_t = \frac{1}{L}$, which is the fixed stepsize rule.

Theorem 5.12 *Suppose J has a Lipschitz continuous gradient with modulus L. Then Algorithm 5.2 with a fixed stepsize $\eta_t = \frac{1}{L}$ will return a solution w_t with $\|\nabla J(w_t)\| \leq \epsilon$ in at most $O(1/\epsilon^2)$ iterations.*

Proof Plugging in $\eta_t = \frac{1}{L}$ and rearranging (5.45) obtains

$$\frac{1}{2L} \|\nabla J(w_t)\|^2 \leq J(w_t) - J(w_{t+1}) \tag{5.46}$$

Summing this inequality

$$\frac{1}{2L} \sum_{t=0}^{T} \|\nabla J(w_t)\|^2 \leq J(w_0) - J(w_T) \leq J(w_0) - J(w^*),$$

which clearly shows that $\|\nabla J(w_t)\| \to 0$ as $t \to \infty$. Furthermore, we can write the following simple inequality:

$$\|\nabla J(w_T)\| \leq \sqrt{\frac{2L(J(w_0) - J(w^*))}{T+1}}.$$

Solving for

$$\sqrt{\frac{2L(J(w_0) - J(w^*))}{T+1}} = \epsilon$$

shows that T is $O(1/\epsilon^2)$ as claimed. ■

If in addition to having a Lipschitz continuous gradient, if J is σ-strongly convex, then more can be said. First, one can translate convergence in $\|\nabla J(w_t)\|$ to convergence in function values. Towards this end, use (5.17) to write

$$J(w_t) \leq J(w^*) + \frac{1}{2\sigma} \|\nabla J(w_t)\|^2 .$$

Therefore, it follows that whenever $\|\nabla J(w_t)\| < \epsilon$ we have $J(w_t) - J(w^*) < \epsilon^2/2\sigma$. Furthermore, we can strengthen the rates of convergence.

Theorem 5.13 *Assume everything as in Theorem 5.12. Moreover assume that J is σ-strongly convex, and let $c := 1 - \frac{\sigma}{L}$. Then $J(w_t) - J(w^*) \leq \epsilon$ after at most*

$$\frac{\log((J(w_0) - J(w^*))/\epsilon)}{\log(1/c)} \tag{5.47}$$

iterations.

Proof Combining (5.46) with $\|\nabla J(w_t)\|^2 \geq 2\sigma(J(w_t) - J(w^*))$, and using the definition of c one can write

$$c(J(w_t) - J(w^*)) \geq J(w_{t+1}) - J(w^*).$$

Applying the above equation recursively

$$c^T(J(w_0) - J(w^*)) \geq J(w_T) - J(w^*).$$

Solving for

$$\epsilon = c^T(J(w_0) - J(w^*))$$

and rearranging yields (5.47). ■

When applied to practical problems which are not strongly convex gradient descent yields a low accuracy solution within a few iterations. However, as the iterations progress the method "stalls" and no further increase in accuracy is obtained because of the $O(1/\epsilon^2)$ rates of convergence. On the other hand, if the function is strongly convex, then gradient descent converges linearly, that is, in $O(\log(1/\epsilon))$ iterations. However, the number

of iterations depends inversely on $\log(1/c)$. If we approximate $\log(1/c) = -\log(1-\sigma/L) \approx \sigma/L$, then it shows that convergence depends on the ratio L/σ. This ratio is called the *condition number* of a problem. If the problem is well conditioned, *i.e.,* $\sigma \approx L$ then gradient descent converges extremely fast. In contrast, if $\sigma \ll L$ then gradient descent requires many iterations. This is best illustrated with an example: Consider the quadratic objective function

$$J(w) = \frac{1}{2} w^\top A w - bw, \tag{5.48}$$

where $A \in \mathbb{R}^{n \times n}$ is a symmetric positive definite matrix, and $b \in \mathbb{R}^n$ is any arbitrary vector.

Recall that a twice differentiable function is σ-strongly convex and has a Lipschitz continuous gradient with modulus L if and only if its Hessian satisfies $LI \succeq \nabla^2 J(w) \succeq \sigma I$ (see (5.14) and (5.21)). In the case of the quadratic function (5.48) $\nabla^2 J(w) = A$ and hence $\sigma = \lambda_{\min}$ and $L = \lambda_{\max}$, where $\lambda_{\min}$ (respectively $\lambda_{\max}$) denotes the minimum (respectively maximum) eigenvalue of A. One can thus change the condition number of the problem by varying the eigen-spectrum of the matrix A. For instance, if we set A to the $n \times n$ identity matrix, then $\lambda_{\max} = \lambda_{\min} = 1$ and hence the problem is well conditioned. In this case, gradient descent converges very quickly to the optimal solution. We illustrate this behavior on a two dimensional quadratic function in Figure 5.8 (right).

On the other hand, if we choose A such that $\lambda_{\max} \gg \lambda_{\min}$ then the problem (5.48) becomes ill-conditioned. In this case gradient descent exhibits zigzagging and slow convergence as can be seen in Figure 5.8 (left). Because of these shortcomings, gradient descent is not widely used in practice. A number of different algorithms we described below can be understood as explicitly or implicitly changing the condition number of the problem to accelerate convergence.

5.2.4 Mirror Descent

One way to motivate gradient descent is to use the following quadratic approximation of the objective function

$$Q_t(w) := J(w_t) + \langle \nabla J(w_t), w - w_t \rangle + \frac{1}{2}(w - w_t)^\top (w - w_t), \tag{5.49}$$

where, as in the previous section, $\nabla J(\cdot)$ denotes the gradient of J. Minimizing this quadratic model at every iteration entails taking gradients with

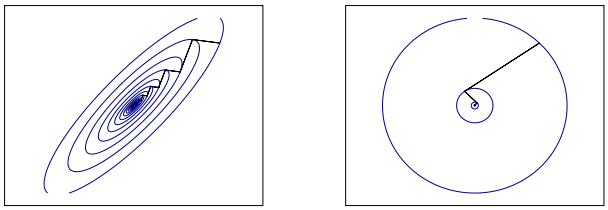

Fig. 5.8. Convergence of gradient descent with exact line search on two quadratic problems (5.48). The problem on the left is ill-conditioned, whereas the problem on the right is well-conditioned. We plot the contours of the objective function, and the steps taken by gradient descent. As can be seen gradient descent converges fast on the well conditioned problem, while it zigzags and takes many iterations to converge on the ill-conditioned problem.

respect to w and setting it to zero, which gives

$$w - w_t := -\nabla J(w_t). \tag{5.50}$$

Performing a line search along the direction $-\nabla J(w_t)$ recovers the familiar gradient descent update

$$w_{t+1} = w_t - \eta_t \nabla J(w_t). \tag{5.51}$$

The closely related mirror descent method replaces the quadratic penalty in (5.49) by a Bregman divergence defined by some convex function f to yield

$$Q_t(w) := J(w_t) + \langle \nabla J(w_t), w - w_t \rangle + \Delta_f(w, w_t). \tag{5.52}$$

Computing the gradient, setting it to zero, and using $\nabla_w \Delta_f(w, w_t) = \nabla f(w) - \nabla f(w_t)$, the minimizer of the above model can be written as

$$\nabla f(w) - \nabla f(w_t) = -\nabla J(w_t). \tag{5.53}$$

As before, by using a stepsize η_t the resulting updates can be written as

$$w_{t+1} = \nabla f^{-1}(\nabla f(w_t) - \eta_t \nabla J(w_t)). \tag{5.54}$$

It is easy to verify that choosing $f(\cdot) = \frac{1}{2}\|\cdot\|^2$ recovers the usual gradient descent updates. On the other hand if we choose f to be the un-normalized entropy (5.30) then $\nabla f(\cdot) = \log \cdot$ and therefore (5.54) specializes to

$$w_{t+1} = \exp(\log(w_t) - \eta_t \nabla J(w_t)) = w_t \exp(-\eta_t \nabla J(w_t)), \tag{5.55}$$

which is sometimes called the Exponentiated Gradient (EG) update.

Theorem 5.14 *Let J be a convex function and $J(w^*)$ denote its minimum value. The mirror descent updates (5.54) with a σ-strongly convex function f satisfy*

$$\frac{\Delta_f(w^*, w_1) + \frac{1}{2\sigma}\sum_t \eta_t^2 \left\|\nabla J(w_t)\right\|^2}{\sum_t \eta_t} \geq \min_t J(w_t) - J(w^*).$$

Proof Using the convexity of J (see (5.7)) and (5.54) we can write

$$\begin{aligned} J(w^*) &\geq J(w_t) + \langle w^* - w_t, \nabla J(w_t)\rangle \\ &\geq J(w_t) - \frac{1}{\eta_t}\left\langle w^* - w_t, f(w_{t+1}) - f(w_t)\right\rangle. \end{aligned}$$

Now applying Lemma 5.11 and rearranging

$$\Delta_f(w^*, w_t) - \Delta_f(w^*, w_{t+1}) + \Delta_f(w_t, w_{t+1}) \geq \eta_t(J(w_t) - J(w^*)).$$

Summing over $t = 1, \ldots, T$

$$\Delta_f(w^*, w_1) - \Delta_f(w^*, w_{T+1}) + \sum_t \Delta_f(w_t, w_{t+1}) \geq \sum_t \eta_t(J(w_t) - J(w^*)).$$

Noting that $\Delta_f(w^*, w_{T+1}) \geq 0$, $J(w_t) - J(w^*) \geq \min_t J(w_t) - J(w^*)$, and rearranging it follows that

$$\frac{\Delta_f(w^*, w_1) + \sum_t \Delta_f(w_t, w_{t+1})}{\sum_t \eta_t} \geq \min_t J(w_t) - J(w^*). \tag{5.56}$$

Using (5.17) and (5.54)

$$\Delta_f(w_t, w_{t+1}) \leq \frac{1}{2\sigma}\left\|\nabla f(w_t) - \nabla f(w_{t+1})\right\|^2 = \frac{1}{2\sigma}\eta_t^2\left\|\nabla J(w_t)\right\|^2. \tag{5.57}$$

The proof is completed by plugging in (5.57) into (5.56). ■

Corollary 5.15 *If J has a Lipschitz continuous gradient with modulus L, and the stepsizes η_t are chosen as*

$$\eta_t = \frac{\sqrt{2\sigma\Delta_f(w^*, w_1)}}{L}\frac{1}{\sqrt{t}} \text{ then} \tag{5.58}$$

$$\min_{1\leq t\leq T} J(w_t) - J(w^*) \leq L\sqrt{\frac{2\Delta_f(w^*, w_1)}{\sigma}}\frac{1}{\sqrt{T}}.$$

Proof Since ∇J is Lipschitz continuous

$$\min_{1\leq t\leq T} J(w_t) - J(w^*) \leq \frac{\Delta_f(w^*, w_1) + \frac{1}{2\sigma}\sum_t \eta_t^2 L^2}{\sum_t \eta_t}.$$

Plugging in (5.58) and using Problem 5.15

$$\min_{1\le t\le T} J(w_t) - J(w^*) \le L\sqrt{\frac{\Delta_f(w^*, w_1)}{2\sigma}}\frac{(1+\sum_t \frac{1}{t})}{\sum_t \frac{1}{\sqrt{t}}} \le L\sqrt{\frac{\Delta_f(w^*, w_1)}{2\sigma}}\frac{1}{\sqrt{T}}.$$

■

5.2.5 Conjugate Gradient

Let us revisit the problem of minimizing the quadratic objective function (5.48). Since $\nabla J(w) = Aw - b$, at the optimum $\nabla J(w) = 0$ (see Lemma 5.6) and hence

$$Aw = b. \tag{5.59}$$

In fact, the Conjugate Gradient (CG) algorithm was first developed as a method to solve the above linear system.

As we already saw, updating w along the negative gradient direction may lead to zigzagging. Therefore CG uses the so-called *conjugate directions*.

Definition 5.16 (Conjugate Directions) *Non zero vectors p_t and $p_{t'}$ are said to be conjugate with respect to a symmetric positive definite matrix A if $p_{t'}^\top A p_t = 0$ if $t \neq t'$.*

Conjugate directions $\{p_0, \ldots, p_{n-1}\}$ are linearly independent and form a basis. To see this, suppose the p_t's are not linearly independent. Then there exists non-zero coefficients σ_t such that $\sum_t \sigma_t p_t = 0$. The p_t's are conjugate directions, therefore $p_{t'}^\top A(\sum_t \sigma_t p_t) = \sum_t \sigma_t p_{t'}^\top A p_t = \sigma_{t'} p_{t'}^\top A p_{t'} = 0$ for all t'. Since A is positive definite this implies that $\sigma_{t'} = 0$ for all t', a contradiction.

As it turns out, the conjugate directions can be generated iteratively as follows: Starting with any $w_0 \in \mathbb{R}^n$ define $p_0 = -g_0 = b - Aw_0$, and set

$$\alpha_t = -\frac{g_t^\top p_t}{p_t^\top A p_t} \tag{5.60a}$$

$$w_{t+1} = w_t + \alpha_t p_t \tag{5.60b}$$

$$g_{t+1} = Aw_{t+1} - b \tag{5.60c}$$

$$\beta_{t+1} = \frac{g_{t+1}^\top A p_t}{p_t^\top A p_t} \tag{5.60d}$$

$$p_{t+1} = -g_{t+1} + \beta_{t+1} p_t \tag{5.60e}$$

The following theorem asserts that the p_t generated by the above procedure are indeed conjugate directions.

Theorem 5.17 *Suppose the t-th iterate generated by the conjugate gradient method (5.60) is not the solution of (5.59), then the following properties hold:*

$$\operatorname{span}\{g_0, g_1, \ldots, g_t\} = \operatorname{span}\{g_0, Ag_0, \ldots, A^t g_0\}. \tag{5.61}$$

$$\operatorname{span}\{p_0, p_1, \ldots, p_t\} = \operatorname{span}\{g_0, Ag_0, \ldots, A^t g_0\}. \tag{5.62}$$

$$p_j^\top g_t = 0 \text{ for all } j < t \tag{5.63}$$

$$p_j^\top A p_t = 0 \text{ for all } j < t. \tag{5.64}$$

Proof The proof is by induction. The induction hypothesis holds trivially at $t = 0$. Assuming that (5.61) to (5.64) hold for some t, we prove that they continue to hold for $t + 1$.

Step 1: We first prove that (5.63) holds. Using (5.60c), (5.60b) and (5.60a)

$$\begin{aligned} p_j^\top g_{t+1} &= p_j^\top (Aw_{t+1} - b) \\ &= p_j^\top (Aw_t + \alpha_t p_t - b) \\ &= p_j^\top \left(Aw_t - \frac{g_t^\top p_t}{p_t^\top A p_t} A p_t - b \right) \\ &= p_j^\top g_t - \frac{p_j^\top A p_t}{p_t^\top A p_t} g_t^\top p_t. \end{aligned}$$

For $j = t$, both terms cancel out, while for $j < t$ both terms vanish due to the induction hypothesis.

Step 2: Next we prove that (5.61) holds. Using (5.60c) and (5.60b)

$$g_{t+1} = Aw_{t+1} - b = Aw_t + \alpha_t A p_t - b = g_t + \alpha_t A p_t.$$

By our induction hypothesis, $g_t \in \operatorname{span}\{g_0, Ag_0, \ldots, A^t g_0\}$, while $Ap_t \in \operatorname{span}\{Ag_0, A^2 g_0, \ldots, A^{t+1} g_0\}$. Combining the two we conclude that $g_{t+1} \in \operatorname{span}\{g_0, Ag_0, \ldots, A^{t+1} g_0\}$. On the other hand, we already showed that g_{t+1} is orthogonal to $\{p_0, p_1, \ldots, p_t\}$. Therefore, $g_{t+1} \notin \operatorname{span}\{p_0, p_1, \ldots, p_t\}$. Thus our induction assumption implies that $g_{t+1} \notin \operatorname{span}\{g_0, Ag_0, \ldots, A^t g_0\}$. This allows us to conclude that $\operatorname{span}\{g_0, g_1, \ldots, g_{t+1}\} = \operatorname{span}\{g_0, Ag_0, \ldots, A^{t+1} g_0\}$.

Step 3 We now prove (5.64) holds. Using (5.60e)

$$p_{t+1}^\top A p_j = -g_{t+1}^\top A p_j + \beta_{t+1} p_t^\top A p_j.$$

By the definition of β_{t+1} (5.60d) the above expression vanishes for $j = t$. For $j < t$, the first term is zero because $Ap_j \in \text{span}\{p_0, p_1, \ldots, p_{j+1}\}$, a subspace orthogonal to g_{t+1} as already shown in Step 1. The induction hypothesis guarantees that the second term is zero.

Step 4 Clearly, (5.61) and (5.60e) imply (5.62). This concludes the proof. ■

A practical implementation of (5.60) requires two more observations: First, using (5.60e) and (5.63)

$$-g_t^\top p_t = g_t^\top g_t - \beta_t g_t^\top p_{t-1} = g_t^\top g_t.$$

Therefore (5.60a) simplifies to

$$\alpha_t = \frac{g_t^\top g_t}{p_t^\top A p_t}. \tag{5.65}$$

Second, using (5.60c) and (5.60b)

$$g_{t+1} - g_t = A(w_{t+1} - w_t) = \alpha_t A p_t.$$

But $g_t \in \text{span}\{p_0, \ldots, p_t\}$, a subspace orthogonal to g_{t+1} by (5.63). Therefore $g_{t+1}^\top A p_t = \frac{1}{\alpha_t}(g_{t+1}^\top g_{t+1})$. Substituting this back into (5.60d) and using (5.65) yields

$$\beta_{t+1} = \frac{g_{t+1}^\top g_{t+1}}{g_t^\top g_t}. \tag{5.66}$$

We summarize the CG algorithm in Algorithm 5.3. Unlike gradient descent whose convergence rates for minimizing the quadratic objective function (5.48) depend upon the condition number of A, as the following theorem shows, the CG iterates converge in at most n steps.

Theorem 5.18 *The CG iterates* (5.60) *converge to the minimizer of* (5.48) *after at most n steps.*

Proof Let w denote the minimizer of (5.48). Since the p_t's form a basis

$$w - w_0 = \sigma_0 p_0 + \ldots + \sigma_{n-1} p_{n-1},$$

for some scalars σ_t. Our proof strategy will be to show that the coefficients

Algorithm 5.3 Conjugate Gradient

Input: Initial point w_0, residual norm tolerance ϵ
Set $t = 0$, $g_0 = Aw_0 - b$, and $p_0 = -g_0$
while $\|Aw_t - b\| \geq \epsilon$ **do**
  $\alpha_t = \frac{g_t^\top g_t}{p_t^\top A p_t}$
  $w_{t+1} = w_t + \alpha_t p_t$
  $g_{t+1} = g_t + \alpha_t A p_t$
  $\beta_{t+1} = \frac{g_{t+1}^\top g_{t+1}}{g_t^\top g_t}$
  $p_{t+1} = -g_{t+1} + \beta_{t+1} p_t$
  $t = t + 1$
end while
Return: w_t

σ_t coincide with α_t defined in (5.60a). Towards this end premultiply with $p_t^\top A$ and use conjugacy to obtain

$$\sigma_t = \frac{p_t^\top A(w - w_0)}{p_t^\top A p_t}. \tag{5.67}$$

On the other hand, following the iterative process (5.60b) from w_0 until w_t yields

$$w_t - w_0 = \alpha_0 p_0 + \ldots + \alpha_{t-1} p_{t-1}.$$

Again premultiplying with $p_t^\top A$ and using conjugacy

$$p_t^\top A(w_t - w_0) = 0. \tag{5.68}$$

Substituting (5.68) into (5.67) produces

$$\sigma_t = \frac{p_t^\top A(w - w_t)}{p_t^\top A p_t} = -\frac{g_t^\top p_t}{p_t^\top A p_t}, \tag{5.69}$$

thus showing that $\sigma_t = \alpha_t$. ■

Observe that the g_{t+1} computed via (5.60c) is nothing but the gradient of $J(w_{t+1})$. Furthermore, consider the following one dimensional optimization problem:

$$\min_{\alpha \in \mathbb{R}} \phi_t(\alpha) := J(w_t + \alpha p_t).$$

Differentiating ϕ_t with respect to α

$$\phi_t'(\alpha) = p_t^\top (Aw_t + \alpha A p_t - b) = p_t^\top (g_t + \alpha A p_t).$$

The gradient vanishes if we set $\alpha = -\frac{g_t^\top p_t}{p_t^\top A p_t}$, which recovers (5.60a). In other words, every iteration of CG minimizes $J(w)$ along a conjugate direction p_t. Contrast this with gradient descent which minimizes $J(w)$ along the negative gradient direction g_t at every iteration.

It is natural to ask if this idea of generating conjugate directions and minimizing the objective function along these directions can be applied to general convex functions. The main difficulty here is that Theorems 5.17 and 5.18 do not hold. In spite of this, extensions of CG are effective even in this setting. Basically the update rules for g_t and p_t remain the same, but the parameters α_t and β_t are computed differently. Table 5.2 gives an overview of different extensions. See [NW99, Lue84] for details.

Table 5.2. *Non-Quadratic modifications of Conjugate Gradient Descent*

Generic Method	Compute Hessian $K_t := \nabla^2 J(w_t)$ and update α_t and β_t with $\alpha_t = -\frac{g_t^\top p_t}{p_t^\top K_t p_t}$ and $\beta_t = -\frac{g_{t+1}^\top K_t p_t}{p_t^\top K_t p_t}$
Fletcher-Reeves	Set $\alpha_t = \operatorname{argmin}_\alpha J(w_t + \alpha p_t)$ and $\beta_t = \frac{g_{t+1}^\top g_{t+1}}{g_t^\top g_t}$.
Polak-Ribière	Set $\alpha_t = \operatorname{argmin}_\alpha J(w_t + \alpha p_t)$, $y_t = g_{t+1} - g_t$, and $\beta_t = \frac{y_t^\top g_{t+1}}{g_t^\top g_t}$. In practice, Polak-Ribière tends to be better than Fletcher-Reeves.
Hestenes-Stiefel	Set $\alpha_t = \operatorname{argmin}_\alpha J(w_t + \alpha p_t)$, $y_t = g_{t+1} - g_t$, and $\beta_t = \frac{y_t^\top g_{t+1}}{y_t^\top p_t}$.

5.2.6 Higher Order Methods

Recall the motivation for gradient descent as the minimizer of the quadratic model

$$Q_t(w) := J(w_t) + \langle \nabla J(w_t), w - w_t \rangle + \frac{1}{2}(w - w_t)^\top (w - w_t),$$

The quadratic penalty in the above equation uniformly penalizes deviation from w_t in different dimensions. When the function is ill-conditioned one would intuitively want to penalize deviations in different directions differently. One way to achieve this is by using the Hessian, which results in the

Algorithm 5.4 Newton's Method

Input: Initial point w_0, gradient norm tolerance ϵ
Set $t = 0$
while $\|\nabla J(w_t)\| > \epsilon$ **do**
  Compute $p_t := -\nabla^2 J(w_t)^{-1} \nabla J(w_t)$
  Compute $\eta_t = \text{argmin}_\eta J(w_t + \eta p_t)$ *e.g.,* via Algorithm 5.1.
  $w_{t+1} = w_t + \eta_t p_t$
  $t = t + 1$
end while
Return: w_t

following second order Taylor approximation:

$$Q_t(w) := J(w_t) + \langle \nabla J(w_t), w - w_t \rangle + \frac{1}{2}(w - w_t)^\top \nabla^2 J(w_t)(w - w_t). \tag{5.70}$$

Of course, this requires that J be twice differentiable. We will also assume that J is strictly convex and hence its Hessian is positive definite and invertible. Minimizing Q_t by taking gradients with respect to w and setting it zero obtains

$$w - w_t := -\nabla^2 J(w_t)^{-1} \nabla J(w_t), \tag{5.71}$$

Since we are only minimizing a model of the objective function, we perform a line search along the descent direction (5.71) to compute the stepsize η_t, which yields the next iterate:

$$w_{t+1} = w_t - \eta_t \nabla^2 J(w_t)^{-1} \nabla J(w_t). \tag{5.72}$$

Details can be found in Algorithm 5.4.

Suppose w^* denotes the minimum of $J(w)$. We say that an algorithm exhibits quadratic convergence if the sequences of iterates $\{w_k\}$ generated by the algorithm satisfies:

$$\|w_{k+1} - w^*\| \leq C \|w_k - w_*\|^2 \tag{5.73}$$

for some constant $C > 0$. We now show that Newton's method exhibits quadratic convergence close to the optimum.

Theorem 5.19 (Quadratic convergence of Newton's Method) *Suppose J is twice differentiable, strongly convex, and the Hessian of J is bounded and Lipschitz continuous with modulus M in a neighborhood of the solution w^*. Furthermore, assume that $\left\|\nabla^2 J(w)^{-1}\right\| \leq N$. The iterations*

$w_{t+1} = w_t - \nabla^2 J(w_t)^{-1} \nabla J(w_t)$ converge quadratically to w^*, the minimizer of J.

Proof First notice that

$$\nabla J(w_t) - \nabla J(w^*) = \int_0^1 \nabla^2 J(w_t + t(w^* - w_t))(w_t - w^*) dt. \tag{5.74}$$

Next using the fact that $\nabla^2 J(w_t)$ is invertible and the gradient vanishes at the optimum ($\nabla J(w^*) = 0$), write

$$\begin{aligned} w_{t+1} - w^* &= w_t - w^* - \nabla^2 J(w_t)^{-1} \nabla J(w_t) \\ &= \nabla^2 J(w_t)^{-1} [\nabla^2 J(w_t)(w_t - w^*) - (\nabla J(w_t) - \nabla J(w^*))]. \end{aligned} \tag{5.75}$$

Using (5.75), (5.74), and the Lipschitz continuity of $\nabla^2 J$

$$\begin{aligned} &\left\| \nabla J(w_t) - \nabla J(w^*) - \nabla^2 J(w_t)(w_t - w^*) \right\| \\ &= \left\| \int_0^1 [\nabla^2 J(w_t + t(w_t - w^*)) - \nabla^2 J(w_t)](w_t - w^*) dt \right\| \\ &\leq \int_0^1 \left\| [\nabla^2 J(w_t + t(w_t - w^*)) - \nabla^2 J(w_t)] \right\| \left\| (w_t - w^*) \right\| dt \\ &\leq \left\| w_t - w^* \right\|^2 \int_0^1 M t \, dt = \frac{M}{2} \left\| w_t - w^* \right\|^2 . \end{aligned} \tag{5.76}$$

Finally use (5.75) and (5.76) to conclude that

$$\left\| w_{t+1} - w^* \right\| \leq \frac{M}{2} \left\| \nabla^2 J(w_t)^{-1} \right\| \left\| w_t - w^* \right\|^2 \leq \frac{NM}{2} \| w_t - w^* \|^2.$$

■

Newton's method as we described it suffers from two major problems. First, it applies only to twice differentiable, strictly convex functions. Second, it involves computing and inverting of the $n \times n$ Hessian matrix at every iteration, thus making it computationally very expensive. Although Newton's method can be extended to deal with positive semi-definite Hessian matrices, the computational burden often makes it unsuitable for large scale applications. In such cases one resorts to Quasi-Newton methods.

5.2.6.1 Quasi-Newton Methods

Unlike Newton's method, which computes the Hessian of the objective function at every iteration, quasi-Newton methods never compute the Hessian; they approximate it from past gradients. Since they do not require the objective function to be twice differentiable, quasi-Newton methods are much

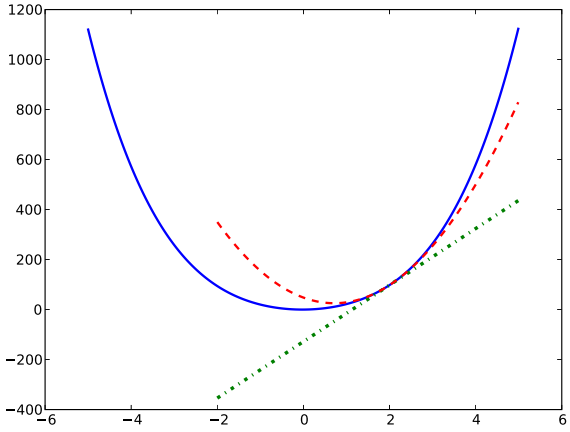

Fig. 5.9. The blue solid line depicts the one dimensional convex function $J(w) = w^4 + 20w^2 + w$. The green dotted-dashed line represents the first order Taylor approximation to $J(w)$, while the red dashed line represents the second order Taylor approximation, both evaluated at $w = 2$.

more widely applicable. They are widely regarded as the workhorses of smooth nonlinear optimization due to their combination of computational efficiency and good asymptotic convergence. The most popular quasi-Newton algorithm is BFGS, named after its discoverers Broyde, Fletcher, Goldfarb, and Shanno. In this section we will describe BFGS and its limited memory counterpart LBFGS.

Suppose we are given a smooth (not necessarily strictly) convex objective function $J : \mathbb{R}^n \to \mathbb{R}$ and a current iterate $w_t \in \mathbb{R}^n$. Just like Newton's method, BFGS forms a local quadratic model of the objective function, J:

$$Q_t(w) \;:=\; J(w_t) + \langle \nabla J(w_t), w - w_t \rangle + \frac{1}{2}(w - w_t)^\top H_t (w - w_t). \quad (5.77)$$

Unlike Newton's method which uses the Hessian to build its quadratic model (5.70), BFGS uses the matrix $H_t \succ 0$, which is a positive-definite *estimate* of the Hessian. A quasi-Newton direction of descent is found by minimizing $Q_t(w)$:

$$w - w_t = -H_t^{-1} \nabla J(w_t). \quad (5.78)$$

The stepsize $\eta_t > 0$ is found by a line search obeying the Wolfe conditions

(5.42) and (5.43). The final update is given by

$$w_{t+1} = w_t - \eta_t H_t^{-1} \nabla J(w_t). \tag{5.79}$$

Given w_{t+1} we need to update our quadratic model (5.77) to

$$Q_{t+1}(w) \;:=\; J(w_{t+1}) + \langle \nabla J(w_{t+1}), w - w_{t+1} \rangle + \frac{1}{2}(w - w_{t+1})^\top H_{t+1}(w - w_{t+1}). \tag{5.80}$$

When updating our model it is reasonable to expect that the gradient of Q_{t+1} should match the gradient of J at w_t and w_{t+1}. Clearly,

$$\nabla Q_{t+1}(w) \;=\; \nabla J(w_{t+1}) + H_{t+1}(w - w_{t+1}), \tag{5.81}$$

which implies that $\nabla Q_{t+1}(w_{t+1}) = \nabla J(w_{t+1})$, and hence our second condition is automatically satisfied. In order to satisfy our first condition, we require

$$\nabla Q_{t+1}(w_t) \;=\; \nabla J(w_{t+1}) + H_{t+1}(w_t - w_{t+1}) = \nabla J(w_t). \tag{5.82}$$

By rearranging, we obtain the so-called *secant equation*:

$$H_{t+1} s_t = y_t, \tag{5.83}$$

where $s_t := w_{t+1} - w_t$ and $y_t := \nabla J(w_{t+1}) - \nabla J(w_t)$ denote the most recent step along the optimization trajectory in parameter and gradient space, respectively. Since H_{t+1} is a positive definite matrix, pre-multiplying the secant equation by s_t yields the *curvature condition*

$$s_t^\top y_t > 0. \tag{5.84}$$

If the curvature condition is satisfied, then there are an infinite number of matrices H_{t+1} which satisfy the secant equation (the secant equation represents n linear equations, but the symmetric matrix H_{t+1} has $n(n+1)/2$ degrees of freedom). To resolve this issue we choose the closest matrix to H_t which satisfies the secant equation. The key insight of the BFGS comes from the observation that the descent direction computation (5.78) involves the inverse matrix $B_t := H_t^{-1}$. Therefore, we choose a matrix $B_{t+1} := H_{t+1}^{-1}$ such that it is close to B_t and also satisfies the secant equation:

$$\min_B \|B - B_t\| \tag{5.85}$$

$$\text{s. t. } B = B^\top \text{ and } B y_t = s_t. \tag{5.86}$$

If the matrix norm $\|\cdot\|$ is appropriately chosen [NW99], then it can be shown that

$$B_{t+1} = (\mathbf{1} - \rho_t s_t y_t^\top) B_t (\mathbf{1} - \rho_t y_t s_t^\top) + \rho_t s_t s_t^\top, \tag{5.87}$$

Algorithm 5.5 LBFGS

Input: Initial point w_0, gradient norm tolerance $\epsilon > 0$
Set $t = 0$ and $B_0 = I$
while $\|\nabla J(w_t)\| > \epsilon$ **do**
  $p_t = -B_t \nabla J(w_t)$
  Find η_t that obeys (5.42) and (5.43)
  $s_t = \eta_t p_t$
  $w_{t+1} = w_t + s_t$
  $y_t := \nabla J(w_{t+1}) - \nabla J(w_t)$
  if $t = 0$: $B_t := \frac{s_t^\top y_t}{y_t^\top y_t} I$
  $\rho_t = (s_t^\top y_t)^{-1}$
  $B_{t+1} = (I - \rho_t s_t y_t^\top) B_t (I - \rho_t y_t s_t^\top) + \rho_t s_t s_t^\top$
  $t = t + 1$
end while
Return: w_t

where $\rho_t := (y_t^\top s_t)^{-1}$. In other words, the matrix B_t is modified via an incremental rank-two update, which is very efficient to compute, to obtain B_{t+1}.

There exists an interesting connection between the BFGS update (5.87) and the Hestenes-Stiefel variant of Conjugate gradient. To see this assume that an exact line search was used to compute w_{t+1}, and therefore $s_t^\top \nabla J(w_{t+1}) = 0$. Furthermore, assume that $B_t = \mathbf{1}$, and use (5.87) to write

$$p_{t+1} = -B_{t+1} \nabla J(w_{t+1}) = -\nabla J(w_{t+1}) + \frac{y_t^\top \nabla J(w_{t+1})}{y_t^\top s_t} s_t, \tag{5.88}$$

which recovers the Hestenes-Stiefel update (see (5.60e) and Table 5.2).

Limited-memory BFGS (LBFGS) is a variant of BFGS designed for solving large-scale optimization problems where the $O(d^2)$ cost of storing and updating B_t would be prohibitively expensive. LBFGS approximates the quasi-Newton direction (5.78) directly from the last m pairs of s_t and y_t via a matrix-free approach. This reduces the cost to $O(md)$ space and time per iteration, with m freely chosen. Details can be found in Algorithm 5.5.

5.2.6.2 Spectral Gradient Methods

Although spectral gradient methods do not use the Hessian explicitly, they are motivated by arguments very reminiscent of the Quasi-Newton methods. Recall the update rule (5.79) and secant equation (5.83). Suppose we want

a very simple matrix which approximates the Hessian. Specifically, we want

$$H_{t+1} = \alpha_{t+1} I \tag{5.89}$$

where α_{t+1} is a scalar and I denotes the identity matrix. Then the secant equation (5.83) becomes

$$\alpha_{t+1} s_t = y_t. \tag{5.90}$$

In general, the above equation cannot be solved. Therefore we use the α_{t+1} which minimizes $\|\alpha_{t+1} s_t - y_t\|^2$ which yields the Barzilai-Borwein (BB) stepsize

$$\alpha_{t+1} = \frac{s_t^\top y_t}{s_t^\top s_t}. \tag{5.91}$$

As it turns out, α_{t+1} lies between the minimum and maximum eigenvalue of the average Hessian in the direction s_t, hence the name Spectral Gradient method. The parameter update (5.79) is now given by

$$w_{t+1} = w_t - \frac{1}{\alpha_t} \nabla J(w_t). \tag{5.92}$$

A practical implementation uses safeguards to ensure that the stepsize α_{t+1} is neither too small nor too large. Given $0 < \alpha_{\min} < \alpha_{\max} < \infty$ we compute

$$\alpha_{t+1} = \min\left(\alpha_{\max}, \max\left(\alpha_{\min}, \frac{s_t^\top y_t}{s_t^\top s_t}\right)\right). \tag{5.93}$$

One of the peculiar features of spectral gradient methods is their use of a non-monotone line search. In all the algorithms we have seen so far, the stepsize is chosen such that the objective function J decreases at every iteration. In contrast, non-monotone line searches employ a parameter $M \geq 1$ and ensure that the objective function decreases in every M iterations. Of course, setting $M = 1$ results in the usual monotone line search. Details can be found in Algorithm 5.6.

5.2.7 Bundle Methods

The methods we discussed above are applicable for minimizing smooth, convex objective functions. Some regularized risk minimization problems involve a non-smooth objective function. In such cases, one needs to use bundle methods. In order to lay the ground for bundle methods we first describe their precursor the cutting plane method [Kel60]. Cutting plane method is based on a simple observation: A convex function is bounded from below by

Algorithm 5.6 Spectral Gradient Method

Input: w_0, $M \geq 1$, $\alpha_{\max} > \alpha_{\min} > 0$, $\gamma \in (0,1)$, $1 > \sigma_2 > \sigma_1 > 0$, $\alpha_0 \in [\alpha_{\min}, \alpha_{\max}]$, and $\epsilon > 0$
Initialize: $t = 0$
while $\|\nabla J(w_t)\| > \epsilon$ **do**
  $\lambda = 1$
  while TRUE do
    $d_t = -\frac{1}{\alpha_t}\nabla J(w_t)$
    $w_+ = w_t + \lambda d_t$
    $\delta = \langle d_t, \nabla J(w_t)\rangle$
    if $J(w_+) \leq \min_{0\leq j\leq \min(t,M-1)} J(x_{t-j}) + \gamma\lambda\delta$ **then**
      $w_{t+1} = w_+$
      $s_t = w_{t+1} - w_t$
      $y_t = \nabla J(w_{t+1}) - \nabla J(w_t)$
      break
    else
      $\lambda_{\text{tmp}} = -\frac{1}{2}\lambda^2\delta/(J(w_+ - J(w_t) - \lambda\delta)$
      if $\lambda_{\text{tmp}} > \sigma_1$ and $\lambda_{\text{tmp}} < \sigma_2\lambda$ **then**
        $\lambda = \lambda_{\text{tmp}}$
      else
        $\lambda = \lambda/2$
      end if
    end if
  end while
  $\alpha_{t+1} = \min(\alpha_{\max}, \max(\alpha_{\min}, \frac{s_t^\top y_t}{s_t^\top s_t}))$
  $t = t + 1$
end while
Return: w_t

its linearization (*i.e.,* first order Taylor approximation). See Figures 5.4 and 5.5 for geometric intuition, and recall (5.7) and (5.13):

$$J(w) \geq J(w') + \left\langle w - w', s' \right\rangle \quad \forall w \text{ and } s' \in \partial J(w'). \tag{5.94}$$

Given subgradients $s_1, s_2, \ldots, s_t$ evaluated at locations $w_0, w_1, \ldots, w_{t-1}$, we can construct a tighter (piecewise linear) lower bound for J as follows (also see Figure 5.10):

$$J(w) \geq J_t^{\text{CP}}(w) := \max_{1\leq i\leq t} \{J(w_{i-1}) + \langle w - w_{i-1}, s_i\rangle\}. \tag{5.95}$$

Given iterates $\{w_i\}_{i=0}^{t-1}$, the cutting plane method minimizes J_t^{CP} to obtain the next iterate w_t:

$$w_t := \operatorname*{argmin}_w J_t^{\mathrm{CP}}(w). \tag{5.96}$$

This iteratively refines the piecewise linear lower bound J^{CP} and allows us to get close to the minimum of J (see Figure 5.10 for an illustration).

If w^* denotes the minimizer of J, then clearly each $J(w_i) \geq J(w^*)$ and hence $\min_{0 \leq i \leq t} J(w_i) \geq J(w^*)$. On the other hand, since $J \geq J_t^{\mathrm{CP}}$ it follows that $J(w^*) \geq J_t^{\mathrm{CP}}(w_t)$. In other words, $J(w^*)$ is sandwiched between $\min_{0 \leq i \leq t} J(w_i)$ and $J_t^{\mathrm{CP}}(w_t)$ (see Figure 5.11 for an illustration). The cutting plane method monitors the monotonically decreasing quantity

$$\epsilon_t := \min_{0 \leq i \leq t} J(w_i) - J_t^{\mathrm{CP}}(w_t), \tag{5.97}$$

and terminates whenever ϵ_t falls below a predefined threshold ϵ. This ensures that the solution $J(w_t)$ is ϵ optimum, that is, $J(w_t) \leq J(w^*) + \epsilon$.

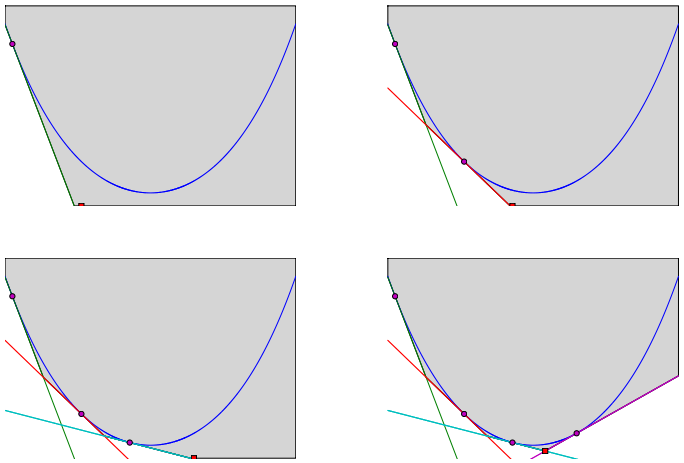

Fig. 5.10. A convex function (blue solid curve) is bounded from below by its linearizations (dashed lines). The gray area indicates the piecewise linear lower bound obtained by using the linearizations. We depict a few iterations of the cutting plane method. At each iteration the piecewise linear lower bound is minimized and a new linearization is added at the minimizer (red rectangle). As can be seen, adding more linearizations improves the lower bound.

Although cutting plane method was shown to be convergent [Kel60], it is

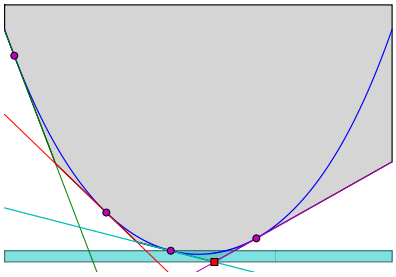

Fig. 5.11. A convex function (blue solid curve) with four linearizations evaluated at four different locations (magenta circles). The approximation gap ϵ_3 at the end of fourth iteration is indicated by the height of the cyan horizontal band *i.e.,* difference between lowest value of $J(w)$ evaluated so far and the minimum of $J_4^{\mathrm{CP}}(w)$ (red diamond).

well known (see *e.g.,* [LNN95, Bel05]) that it can be very slow when new iterates move too far away from the previous ones (*i.e.,* causing unstable "zig-zag" behavior in the iterates). In fact, in the worst case the cutting plane method might require exponentially many steps to converge to an ϵ optimum solution.

Bundle methods stabilize CPM by augmenting the piecewise linear lower (*e.g.,* $J_t^{\mathrm{CP}}(w)$ in (5.95)) with a prox-function (*i.e.,* proximity control function) which prevents overly large steps in the iterates [Kiw90]. Roughly speaking, there are 3 popular types of bundle methods, namely, *proximal* [Kiw90], *trust region* [SZ92], and *level set* [LNN95]. All three versions use $\frac{1}{2}\|\cdot\|^2$ as their prox-function, but differ in the way they compute the new iterate:

$$\text{proximal:} \quad w_t := \underset{w}{\operatorname{argmin}}\{\frac{\zeta_t}{2}\|w - \hat{w}_{t-1}\|^2 + J_t^{\mathrm{CP}}(w)\}, \tag{5.98}$$

$$\text{trust region:} \quad w_t := \underset{w}{\operatorname{argmin}}\{J_t^{\mathrm{CP}}(w) \mid \frac{1}{2}\|w - \hat{w}_{t-1}\|^2 \leq \kappa_t\}, \tag{5.99}$$

$$\text{level set:} \quad w_t := \underset{w}{\operatorname{argmin}}\{\frac{1}{2}\|w - \hat{w}_{t-1}\|^2 \mid J_t^{\mathrm{CP}}(w) \leq \tau_t\}, \tag{5.100}$$

where $\hat{w}_{t-1}$ is the current prox-center, and ζ_t, κ_t, and τ_t are positive trade-off parameters of the stabilization. Although (5.98) can be shown to be equivalent to (5.99) for appropriately chosen ζ_t and κ_t, tuning ζ_t is rather difficult while a trust region approach can be used for automatically tuning

κ_t. Consequently the trust region algorithm BT of [SZ92] is widely used in practice.

5.3 Constrained Optimization

So far our focus was on unconstrained optimization problems. Many machine learning problems involve constraints, and can often be written in the following canonical form:

$$\min_w \ J(w) \tag{5.101a}$$

$$\text{s. t. } \ c_i(w) \leq 0 \text{ for } i \in \mathcal{I} \tag{5.101b}$$

$$e_i(w) = 0 \text{ for } i \in \mathcal{E} \tag{5.101c}$$

where both c_i and e_i are convex functions. We say that w is feasible if and only if it satisfies the constraints, that is, $c_i(w) \leq 0$ for $i \in \mathcal{I}$ and $e_i(w) = 0$ for $i \in \mathcal{E}$.

Recall that w is the minimizer of an unconstrained problem if and only if $\|\nabla J(w)\| = 0$ (see Lemma 5.6). Unfortunately, when constraints are present one cannot use this simple characterization of the solution. For instance, the w at which $\|\nabla J(w)\| = 0$ may not be a feasible point. To illustrate, consider the following simple minimization problem (see Figure 5.12):

$$\min_w \ \frac{1}{2}w^2 \tag{5.102a}$$

$$\text{s. t. } \ 1 \leq w \leq 2. \tag{5.102b}$$

Clearly, $\frac{1}{2}w^2$ is minimized at $w = 0$, but because of the presence of the constraints, the minimum of (5.102) is attained at $w = 1$ where $\nabla J(w) = w$ is equal to 1. Therefore, we need other ways to detect convergence. In Section 5.3.1 we discuss some general purpose algorithms based on the concept of orthogonal projection. In Section 5.3.2 we will discuss Lagrange duality, which can be used to further characterize the solutions of constrained optimization problems.

5.3.1 Projection Based Methods

Suppose we are interested in minimizing a smooth convex function of the following form:

$$\min_{w \in \Omega} J(w), \tag{5.103}$$

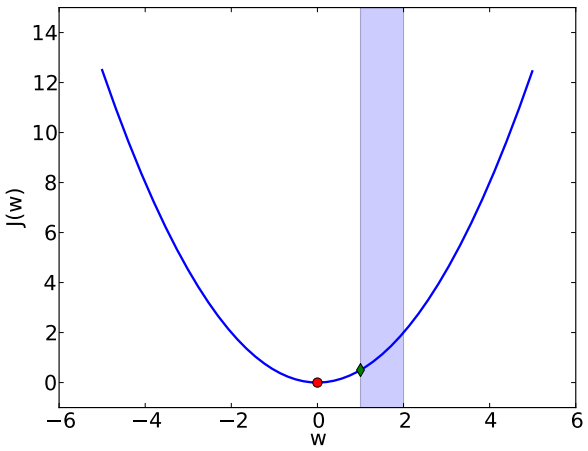

Fig. 5.12. The unconstrained minimum of the quadratic function $\frac{1}{2}w^2$ is attained at $w = 0$ (red circle). But, if we enforce the constraints $1 \leq w \leq 2$ (illustrated by the shaded area) then the minimizer is attained at $w = 1$ (green diamond).

where Ω is a convex feasible region. For instance, Ω may be described by convex functions c_i and e_i as in (5.101). The algorithms we describe in this section are applicable when Ω is a relatively simple set onto which we can compute an orthogonal projection. Given a point w' and a feasible region Ω, the orthogonal projection $P_\Omega(w')$ of w' on Ω is defined as

$$P_\Omega(w') := \operatorname*{argmin}_{w \in \Omega} \left\| w' - w \right\|^2 . \tag{5.104}$$

Geometrically speaking, $P_\Omega(w')$ is the closest point to w' in Ω. Of course, if $w' \in \Omega$ then $P_\Omega(w') = w'$.

We are interested in finding an approximate solution of (5.103), that is, a $w \in \Omega$ such that

$$J(w) - \min_{w \in \Omega} J(w) = J(w) - J^* \leq \epsilon, \tag{5.105}$$

for some pre-defined tolerance $\epsilon > 0$. Of course, J^* is unknown and hence the gap $J(w) - J^*$ cannot be computed in practice. Furthermore, as we showed in Section 5.3, for constrained optimization problems $\|\nabla J(w)\|$ does not vanish at the optimal solution. Therefore, we will use the following stopping

Algorithm 5.7 Basic Projection Based Method

Input: Initial point $w_0 \in \Omega$, and projected gradient norm tolerance $\epsilon > 0$
Initialize: $t = 0$
while $\|P_\Omega(w_t - \nabla J(w_t)) - w_t\| > \epsilon$ **do**
  Find direction of descent d_t
  $w_{t+1} = P_\Omega(w_t + \eta_t d_t)$
  $t = t + 1$
end while
Return: w_t

criterion in our algorithms

$$\|P_\Omega(w_t - \nabla J(w_t)) - w_t\| \leq \epsilon. \tag{5.106}$$

The intuition here is as follows: If $w_t - \nabla J(w_t) \in \Omega$ then $P_\Omega(w_t - \nabla J(w_t)) = w_t$ if, and only if, $\nabla J(w_t) = 0$, that is, w_t is the global minimizer of $J(w)$. On the other hand, if $w_t - \nabla J(w_t) \notin \Omega$ but $P_\Omega(w_t - \nabla J(w_t)) = w_t$, then the constraints are preventing us from making any further progress along the descent direction $-\nabla J(w_t)$ and hence we should stop.

The basic projection based method is described in Algorithm 5.7. Any unconstrained optimization algorithm can be used to generate the direction of descent d_t. A line search is used to find the stepsize η_t. The updated parameter $w_t - \eta_t d_t$ is projected onto Ω to obtain w_{t+1}. If d_t is chosen to be the negative gradient direction $-\nabla J(w_t)$, then the resulting algorithm is called the projected gradient method. One can show that the rates of convergence of gradient descent with various line search schemes is also preserved by projected gradient descent.

5.3.2 Lagrange Duality

Lagrange duality plays a central role in constrained convex optimization. The basic idea here is to augment the objective function (5.101) with a weighted sum of the constraint functions by defining the Lagrangian:

$$L(w, \alpha, \beta) = J(w) + \sum_{i \in \mathcal{I}} \alpha_i c_i(w) + \sum_{i \in \mathcal{E}} \beta_i e_i(w) \tag{5.107}$$

for $\alpha_i \geq 0$ and $\beta_i \in \mathbb{R}$. In the sequel, we will refer to α (respectively β) as the Lagrange multipliers associated with the inequality (respectively equality) constraints. Furthermore, we will call α and β dual feasible if and only if

$\alpha_i \geq 0$ and $\beta_i \in \mathbb{R}$. The Lagrangian satisfies the following fundamental property, which makes it extremely useful for constrained optimization.

Theorem 5.20 *The Lagrangian* (5.107) *of* (5.101) *satisfies*

$$\max_{\alpha \geq 0, \beta} L(w, \alpha, \beta) = \begin{cases} J(w) & \text{if } w \text{ is feasible} \\ \infty & \text{otherwise.} \end{cases}$$

In particular, if J^* *denotes the optimal value of* (5.101)*, then*

$$J^* = \min_w \max_{\alpha \geq 0, \beta} L(w, \alpha, \beta).$$

Proof First assume that w is feasible, that is, $c_i(w) \leq 0$ for $i \in \mathcal{I}$ and $e_i(w) = 0$ for $i \in \mathcal{E}$. Since $\alpha_i \geq 0$ we have

$$\sum_{i \in \mathcal{I}} \alpha_i c_i(w) + \sum_{i \in \mathcal{E}} \beta_i e_i(w) \leq 0, \tag{5.108}$$

with equality being attained by setting $\alpha_i = 0$ whenever $c_i(w) < 0$. Consequently,

$$\max_{\alpha \geq 0, \beta} L(w, \alpha, \beta) = \max_{\alpha \geq 0, \beta} J(w) + \sum_{i \in \mathcal{I}} \alpha_i c_i(w) + \sum_{i \in \mathcal{E}} \beta_i e_i(w) = J(w)$$

whenever w is feasible. On the other hand, if w is not feasible then either $c_{i'}(w) > 0$ or $e_{i'}(w) \neq 0$ for some i'. In the first case simply let $\alpha_{i'} \to \infty$ to see that $\max_{\alpha \geq 0, \beta} L(w, \alpha, \beta) \to \infty$. Similarly, when $e_{i'}(w) \neq 0$ let $\beta_{i'} \to \infty$ if $e_{i'}(w) > 0$ or $\beta_{i'} \to -\infty$ if $e_{i'}(w) < 0$ to arrive at the same conclusion. ■

If define the Lagrange dual function

$$D(\alpha, \beta) = \min_w L(w, \alpha, \beta), \tag{5.109}$$

for $\alpha \geq 0$ and β, then one can prove the following property, which is often called as *weak duality*.

Theorem 5.21 (Weak Duality) *The Lagrange dual function* (5.109) *satisfies*

$$D(\alpha, \beta) \leq J(w)$$

for all feasible w *and* $\alpha \geq 0$ *and* β*. In particular*

$$D^* := \max_{\alpha \geq 0, \beta} \min_w L(w, \alpha, \beta) \leq \min_w \max_{\alpha \geq 0, \beta} L(w, \alpha, \beta) = J^*. \tag{5.110}$$

Proof As before, observe that whenever w is feasible

$$\sum_{i\in\mathcal{I}} \alpha_i c_i(w) + \sum_{i\in\mathcal{E}} \beta_i e_i(w) \leq 0.$$

Therefore

$$D(\alpha,\beta) = \min_w L(w,\alpha,\beta) = \min_w J(w) + \sum_{i\in\mathcal{I}} \alpha_i c_i(w) + \sum_{i\in\mathcal{E}} \beta_i e_i(w) \leq J(w)$$

for all feasible w and $\alpha \geq 0$ and β. In particular, one can choose w to be the minimizer of (5.101) and $\alpha \geq 0$ and β to be maximizers of $D(\alpha,\beta)$ to obtain (5.110). ■

Weak duality holds for any arbitrary function, not-necessarily convex. When the objective function and constraints are convex, and certain technical conditions, also known as Slater's conditions hold, then we can say more.

Theorem 5.22 (Strong Duality) *Supposed the objective function f and constraints c_i for $i \in \mathcal{I}$ and e_i for $i \in \mathcal{E}$ in (5.101) are convex and the following constraint qualification holds:*

There exists a w such that $c_i(w) < 0$ for all $i \in \mathcal{I}$.

Then the Lagrange dual function (5.109) satisfies

$$D^* := \max_{\alpha\geq 0,\beta} \min_w L(w,\alpha,\beta) = \min_w \max_{\alpha\geq 0,\beta} L(w,\alpha,\beta) = J^*. \tag{5.111}$$

The proof of the above theorem is quite technical and can be found in any standard reference (*e.g.,* [BV04]). Therefore we will omit the proof and proceed to discuss various implications of strong duality. First note that

$$\min_w \max_{\alpha\geq 0,\beta} L(w,\alpha,\beta) = \max_{\alpha\geq 0,\beta} \min_w L(w,\alpha,\beta). \tag{5.112}$$

In other words, one can switch the order of minimization over w with maximization over α and β. This is called the *saddle point property* of convex functions.

Suppose strong duality holds. Given any $\alpha \geq 0$ and β such that $D(\alpha,\beta) > -\infty$ and a feasible w we can immediately write the *duality gap*

$$J(w) - J^* = J(w) - D^* \leq J(w) - D(\alpha,\beta),$$

where J^* and D^* were defined in (5.111). Below we show that if w^* is primal optimal and (α^*,β^*) are dual optimal then $J(w^*) - D(\alpha^*,\beta^*) = 0$. This provides a non-heuristic stopping criterion for constrained optimization: stop when $J(w) - D(\alpha,\beta) \leq \epsilon$, where ϵ is a pre-specified tolerance.

Suppose the primal and dual optimal values are attained at w^* and (α^*, β^*) respectively, and consider the following line of argument:

$$\begin{aligned} J(w^*) &= D(\alpha^*, \beta^*) && \text{(5.113a)} \\ &= \min_w J(w) + \sum_{i \in \mathcal{I}} \alpha_i^* c_i(w) + \sum_{i \in \mathcal{E}} \beta_i^* e_j(w) && \text{(5.113b)} \\ &\leq J(w^*) + \sum_{i \in \mathcal{I}} \alpha_i^* c_i(w^*) + \sum_{i \in \mathcal{E}} \beta_i^* e_i(w^*) && \text{(5.113c)} \\ &\leq J(w^*). && \text{(5.113d)} \end{aligned}$$

To write (5.113a) we used strong duality, while (5.113c) obtains by setting $w = w^*$ in (5.113c). Finally, to obtain (5.113d) we used the fact that w^* is feasible and hence (5.108) holds. Since (5.113) holds with equality, one can conclude that the following *complementary slackness condition*:

$$\sum_{i \in \mathcal{I}} \alpha_i^* c_i(w^*) + \sum_{i \in \mathcal{E}} \beta_i^* e_i(w^*) = 0.$$

In other words, $\alpha_i^* c_i(w^*) = 0$ or equivalently $\alpha_i^* = 0$ whenever $c_i(w) < 0$. Furthermore, since w^* minimizes $L(w, \alpha^*, \beta^*)$ over w, it follows that its gradient must vanish at w^*, that is,

$$\nabla J(w^*) + \sum_{i \in \mathcal{I}} \alpha_i^* \nabla c_i(w^*) + \sum_{i \in \mathcal{E}} \beta_i^* \nabla e_i(w^*) = 0.$$

Putting everything together, we obtain

$$\begin{aligned} c_i(w^*) &\leq 0 \quad \forall i \in \mathcal{I} && \text{(5.114a)} \\ e_j(w^*) &= 0 \quad \forall i \in \mathcal{E} && \text{(5.114b)} \\ \alpha_i^* &\geq 0 && \text{(5.114c)} \\ \alpha_i^* c_i(w^*) &= 0 && \text{(5.114d)} \\ \nabla J(w^*) + \sum_{i \in \mathcal{I}} \alpha_i^* \nabla c_i(w^*) + \sum_{i \in \mathcal{E}} \beta_i^* \nabla e_i(w^*) &= 0. && \text{(5.114e)} \end{aligned}$$

The above conditions are called the KKT conditions. If the primal problem is convex, then the KKT conditions are both necessary and sufficient. In other words, if $\hat{w}$ and $(\hat{\alpha}, \hat{\beta})$ satisfy (5.114) then $\hat{w}$ and $(\hat{\alpha}, \hat{\beta})$ are primal and dual optimal with zero duality gap. To see this note that the first two conditions show that $\hat{w}$ is feasible. Since $\alpha_i \geq 0$, $L(w, \alpha, \beta)$ is convex in w. Finally the last condition states that $\hat{w}$ minimizes $L(w, \hat{\alpha}, \hat{\beta})$. Since $\hat{\alpha}_i c_i(\hat{w}) = 0$ and

$e_j(\hat{w}) = 0$, we have

$$\begin{aligned} D(\hat{\alpha}, \hat{\beta}) &= \min_w L(w, \hat{\alpha}, \hat{\beta}) \\ &= J(\hat{w}) + \sum_{i=1}^{n} \hat{\alpha}_i c_i(\hat{w}) + \sum_{j=1}^{m} \hat{\beta}_j e_j(\hat{w}) \\ &= J(\hat{w}). \end{aligned}$$

5.3.3 Linear and Quadratic Programs

So far we discussed general constrained optimization problems. Many machine learning problems have special structure which can be exploited further. We discuss the implication of duality for two such problems.

5.3.3.1 Linear Programming

An optimization problem with a linear objective function and (both equality and inequality) linear constraints is said to be a linear program (LP). A canonical linear program is of the following form:

$$\min_w \quad c^\top w \tag{5.115a}$$

$$\text{s. t. } Aw = b, w \geq 0. \tag{5.115b}$$

Here w and c are n dimensional vectors, while b is a m dimensional vector, and A is a $m \times n$ matrix with $m < n$.

Suppose we are given a LP of the form:

$$\min_w \quad c^\top w \tag{5.116a}$$

$$\text{s. t. } Aw \geq b, \tag{5.116b}$$

we can transform it into a canonical LP by introducing non-negative slack variables

$$\min_{w,\xi} \quad c^\top w \tag{5.117a}$$

$$\text{s. t. } Aw - \xi = b, \xi \geq 0. \tag{5.117b}$$

Next, we split w into its positive and negative parts w^+ and w^- respectively by setting $w_i^+ = \max(0, w_i)$ and $w_i^- = \max(0, -w_i)$. Using these new

variables we rewrite (5.117) as

$$\min_{w^+, w^-, \xi} \quad \begin{bmatrix} c \\ -c \\ 0 \end{bmatrix}^\top \begin{bmatrix} w^+ \\ w^- \\ \xi \end{bmatrix} \tag{5.118a}$$

$$\text{s. t. } \begin{bmatrix} A & -A & -I \end{bmatrix} \begin{bmatrix} w^+ \\ w^- \\ \xi \end{bmatrix} = b, \begin{bmatrix} w^+ \\ w^- \\ \xi \end{bmatrix} \geq 0, \tag{5.118b}$$

thus yielding a canonical LP (5.115) in the variables w^+, w^- and ξ.

By introducing non-negative Lagrange multipliers α and β one can write the Lagrangian of (5.115) as

$$L(w, \beta, s) = c^\top w + \beta^\top (Aw - b) - \alpha^\top w. \tag{5.119}$$

Taking gradients with respect to the primal and dual variables and setting them to zero obtains

$$A^\top \beta - \alpha = c \tag{5.120a}$$

$$Aw = b \tag{5.120b}$$

$$\alpha^\top w = 0 \tag{5.120c}$$

$$w \geq 0 \tag{5.120d}$$

$$\alpha \geq 0. \tag{5.120e}$$

Condition (5.120c) can be simplified by noting that both w and α are constrained to be non-negative, therefore $\alpha^\top w = 0$ if, and only if, $\alpha_i w_i = 0$ for $i = 1, \ldots, n$.

Using (5.120a), (5.120c), and (5.120b) we can write

$$c^\top w = (A^\top \beta - \alpha)^\top w = \beta^\top A w = \beta^\top b.$$

Substituting this into (5.115) and eliminating the primal variable w yields the following dual LP

$$\max_{\alpha, \beta} \quad b^\top \beta \tag{5.121a}$$

$$\text{s.t. } A^\top \beta - \alpha = c, \alpha \geq 0. \tag{5.121b}$$

As before, we let $\beta^+ = \max(\beta, 0)$ and $\beta^- = \max(0, -\beta)$ and convert the

above LP into the following canonical LP

$$\max_{\alpha,\beta^+,\beta^-} \begin{bmatrix} b \\ -b \\ 0 \end{bmatrix}^\top \begin{bmatrix} \beta^+ \\ \beta^- \\ \alpha \end{bmatrix} \tag{5.122a}$$

$$\text{s.t. } \begin{bmatrix} A^\top & -A^\top & -I \end{bmatrix} \begin{bmatrix} \beta^+ \\ \beta^- \\ \alpha \end{bmatrix} = c, \begin{bmatrix} \beta^+ \\ \beta^- \\ \alpha \end{bmatrix} \geq 0. \tag{5.122b}$$

It can be easily verified that the primal-dual problem is symmetric; by taking the dual of the dual we recover the primal (Problem 5.17). One important thing to note however is that the primal (5.115) involves n variables and $n + m$ constraints, while the dual (5.122) involves $2m + n$ variables and $4m + 2n$ constraints.

5.3.3.2 Quadratic Programming

An optimization problem with a convex quadratic objective function and linear constraints is said to be a convex quadratic program (QP). The canonical convex QP can be written as follows:

$$\min_w \ \frac{1}{2} w^\top G x + w^\top d \tag{5.123a}$$

$$\text{s.t. } a_i^\top w = b_i \text{ for } i \in \mathcal{E} \tag{5.123b}$$

$$a_i^\top w \leq b_i \text{ for } i \in \mathcal{I} \tag{5.123c}$$

Here $G \succeq 0$ is a $n \times n$ positive semi-definite matrix, $\mathcal{E}$ and $\mathcal{I}$ are finite set of indices, while d and a_i are n dimensional vectors, and b_i are scalars.

As a warm up let us consider the arguably simpler equality constrained quadratic programs. In this case, we can stack the a_i into a matrix A and the b_i into a vector b to write

$$\min_w \ \frac{1}{2} w^\top G w + w^\top d \tag{5.124a}$$

$$\text{s.t. } Aw = b \tag{5.124b}$$

By introducing non-negative Lagrange multipliers β the Lagrangian of the above optimization problem can be written as

$$L(w, \beta) = \frac{1}{2} w^\top G w + w^\top d + \beta(Aw - b). \tag{5.125}$$

To find the saddle point of the Lagrangian we take gradients with respect

to w and β and set them to zero. This obtains

$$\begin{aligned} Gw + d + A^\top \beta &= 0 \\ Aw &= b. \end{aligned}$$

Putting these two conditions together yields the following linear system of equations

$$\begin{bmatrix} G & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} w \\ \beta \end{bmatrix} = \begin{bmatrix} -d \\ b \end{bmatrix}. \tag{5.126}$$

The matrix in the above equation is called the KKT matrix, and we can use it to characterize the conditions under which (5.124) has a unique solution.

Theorem 5.23 *Let Z be a $n \times (n-m)$ matrix whose columns form a basis for the null space of A, that is, $AZ = 0$. If A has full row rank, and the reduced-Hessian matrix $Z^\top GZ$ is positive definite, then there exists a unique pair (w^*, β^*) which solves (5.126). Furthermore, w^* also minimizes (5.124).*

Proof Note that a unique (w^*, β^*) exists whenever the KKT matrix is non-singular. Suppose this is not the case, then there exist non-zero vectors a and b such that

$$\begin{bmatrix} G & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = 0.$$

Since $Aa = 0$ this implies that a lies in the null space of A and hence there exists a u such that $a = Zu$. Therefore

$$\begin{bmatrix} Zu & 0 \end{bmatrix} \begin{bmatrix} G & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} Zu \\ 0 \end{bmatrix} = u^\top Z^\top GZu = 0.$$

Positive definiteness of $Z^\top GZ$ implies that $u = 0$ and hence $a = 0$. On the other hand, the full row rank of A and $A^\top b = 0$ implies that $b = 0$. In summary, both a and b are zero, a contradiction.

Let $w \neq w^*$ be any other feasible point and $\Delta w = w^* - w$. Since $Aw^* = Aw = b$ we have that $A\Delta w = 0$. Hence, there exists a non-zero u such that $\Delta w = Zu$. The objective function $J(w)$ can be written as

$$\begin{aligned} J(w) &= \frac{1}{2}(w^* - \Delta w)^\top G(w^* - \Delta w) + (w^* - \Delta w)^\top d \\ &= J(w^*) + \frac{1}{2}\Delta w^\top G \Delta w - (Gw^* + d)^\top \Delta w. \end{aligned}$$

First note that $\frac{1}{2}\Delta w^\top G \Delta w = \frac{1}{2} u^\top Z^\top GZu > 0$ by positive definiteness of the reduced Hessian. Second, since w^* solves (5.126) it follows that $(Gw^* +$

$d)^\top \Delta w = \beta^\top A \Delta w = 0$. Together these two observations imply that $J(w) > J(w^*)$. ■

If the technical conditions of the above theorem are met, then solving the equality constrained QP (5.124) is equivalent to solving the linear system (5.126). See [NW99] for a extensive discussion of algorithms that can be used for this task.

Next we turn our attention to the general QP (5.123) which also contains inequality constraints. The Lagrangian in this case can be written as

$$L(w, \beta) = \frac{1}{2} w^\top G w + w^\top d + \sum_{i \in \mathcal{I}} \alpha_i (a_i^\top w - b_i) + \sum_{i \in \mathcal{E}} \beta_i (a_i^\top w - b_i). \quad (5.127)$$

Let w^* denote the minimizer of (5.123). If we define the active set $\mathcal{A}(w^*)$ as

$$\mathcal{A}(w^*) = \left\{ i \text{ s.t. } i \in \mathcal{I} \text{ and } a_i^\top w^* = b_i \right\},$$

then the KKT conditions (5.114) for this problem can be written as

$$a_i^\top w - b_i < 0 \quad \forall i \in \mathcal{I} \setminus \mathcal{A}(w^*) \quad (5.128a)$$

$$a_i^\top w - b_i = 0 \quad \forall i \in \mathcal{E} \cup \mathcal{A}(w^*) \quad (5.128b)$$

$$\alpha_i^* \geq 0 \quad \forall i \in \mathcal{A}(w^*) \quad (5.128c)$$

$$G w^* + d + \sum_{i \in \mathcal{A}(w^*)} \alpha_i^* a_i + \sum_{i \in \mathcal{E}} \beta_i a_i = 0. \quad (5.128d)$$

Conceptually the main difficulty in solving (5.123) is in identifying the active set $\mathcal{A}(w^*)$. This is because $\alpha_i^* = 0$ for all $i \in \mathcal{I} \setminus \mathcal{A}(w^*)$. Most algorithms for solving (5.123) can be viewed as different ways to identify the active set. See [NW99] for a detailed discussion.

5.4 Stochastic Optimization

Recall that regularized risk minimization involves a data-driven optimization problem in which the objective function involves the summation of loss terms over a set of data to be modeled:

$$\min_f \; J(f) := \lambda \Omega(f) + \frac{1}{m} \sum_{i=1}^{m} l(f(x_i), y_i).$$

Classical optimization techniques must compute this sum in its entirety for each evaluation of the objective, respectively its gradient. As available data sets grow ever larger, such "batch" optimizers therefore become increasingly inefficient. They are also ill-suited for the incremental setting, where partial data must be modeled as it arrives.

Stochastic gradient-based methods, by contrast, work with gradient estimates obtained from small subsamples (mini-batches) of training data. This can greatly reduce computational requirements: on large, redundant data sets, simple stochastic gradient descent routinely outperforms sophisticated second-order batch methods by orders of magnitude.

The key idea here is that $J(w)$ is replaced by an instantaneous estimate J_t which is computed from a mini-batch of size k comprising of a subset of points (x_i^t, y_i^t) with $i = 1, \ldots, k$ drawn from the dataset:

$$J_t(w) = \lambda\Omega(w) + \frac{1}{k}\sum_{i=1}^{k} l(w, x_i^t, y_i^t). \tag{5.129}$$

Setting $k = 1$ obtains an algorithm which processes data points as they arrive.

5.4.1 Stochastic Gradient Descent

Perhaps the simplest stochastic optimization algorithm is Stochastic Gradient Descent (SGD). The parameter update of SGD takes the form:

$$w_{t+1} = w_t - \eta_t \nabla J_t(w_t). \tag{5.130}$$

If J_t is not differentiable, then one can choose an arbitrary subgradient from $\partial J_t(w_t)$ to compute the update. It has been shown that SGD asymptotically converges to the true minimizer of $J(w)$ if the stepsize η_t decays as $O(1/\sqrt{t})$. For instance, one could set

$$\eta_t = \sqrt{\frac{\tau}{\tau + t}}, \tag{5.131}$$

where $\tau > 0$ is a tuning parameter. See Algorithm 5.8 for details.

5.4.1.1 Practical Considerations

One simple yet effective rule of thumb to tune τ is to select a small subset of data, try various values of τ on this subset, and choose the τ that most reduces the objective function.

In some cases letting η_t to decay as $O(1/t)$ has been found to be more effective:

$$\eta_t = \frac{\tau}{\tau + t}. \tag{5.132}$$

The free parameter $\tau > 0$ can be tuned as described above. If $\Omega(w)$ is σ-strongly convex, then dividing the stepsize η_t by $\sigma\lambda$ yields good practical performance.

Algorithm 5.8 Stochastic Gradient Descent

Input: Maximum iterations T, batch size k, and τ
Set $t = 0$ and $w_0 = 0$
while $t < T$ **do**
  Choose a subset of k data points (x_i^t, y_i^t) and compute $\nabla J_t(w_t)$
  Compute stepsize $\eta_t = \sqrt{\frac{\tau}{\tau+t}}$
  $w_{t+1} = w_t - \eta_t \nabla J_t(w_t)$
  $t = t + 1$
end while
Return: w_T

5.5 Nonconvex Optimization

Our focus in the previous sections was on convex objective functions. Sometimes non-convex objective functions also arise in machine learning applications. These problems are significantly harder and tools for minimizing such objective functions are not as well developed. We briefly describe one algorithm which can be applied whenever we can write the objective function as a difference of two convex functions.

5.5.1 Concave-Convex Procedure

Any function with a bounded Hessian can be decomposed into the difference of two (non-unique) convex functions, that is, one can write

$$J(w) = f(w) - g(w), \tag{5.133}$$

where f and g are convex functions. Clearly, J is not convex, but there exists a reasonably simple algorithm namely the Concave-Convex Procedure (CCP) for finding a local minima of J. The basic idea is simple: In the t^{th} iteration replace g by its first order Taylor expansion at w_t, that is, $g(w_t) + \langle w - w_t, \nabla g(w_t) \rangle$ and minimize

$$J_t(w) = f(w) - g(w_t) - \langle w - w_t, \nabla g(w_t) \rangle . \tag{5.134}$$

Taking gradients and setting it to zero shows that J_t is minimized by setting

$$\nabla f(w_{t+1}) = \nabla g(w_t). \tag{5.135}$$

The iterations of CCP on a toy minimization problem is illustrated in Figure 5.13, while the complete algorithm listing can be found in Algorithm 5.9.

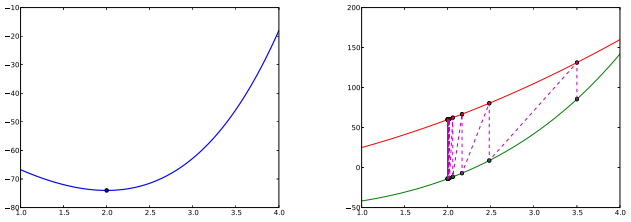

Fig. 5.13. Given the function on the left we decompose it into the difference of two convex functions depicted on the right panel. The CCP algorithm generates iterates by matching points on the two convex curves which have the same tangent vectors. As can be seen, the iterates approach the solution $x = 2.0$.

Algorithm 5.9 Concave-Convex Procedure

Input: Initial point w_0, maximum iterations T, convex functions f,g
Set $t = 0$
while $t < T$ **do**
  Set $w_{t+1} = \text{argmin}_w f(w) - g(w_t) - \langle w - w_t, \nabla g(w_t) \rangle$
  $t = t + 1$
end while
Return: w_T

Theorem 5.24 *Let J be a function which can be decomposed into a difference of two convex functions* e.g., (5.133). *The iterates generated by* (5.135) *monotically decrease J. Furthermore, the stationary point of the iterates is a local minima of J.*

Proof Since f and g are convex

$$f(w_t) \geq f(w_{t+1}) + \langle w_t - w_{t+1}, \nabla f(w_{t+1}) \rangle$$
$$g(w_{t+1}) \geq g(w_t) + \langle w_{t+1} - w_t, \nabla g(w_t) \rangle .$$

Adding the two inequalities, rearranging, and using (5.135) shows that $J(w_t) = f(w_t) - g(w_t) \geq f(w_{t+1}) - g(w_{t+1}) = J(w_{t+1})$, as claimed.

Let w^* be a stationary point of the iterates. Then $\nabla f(w^*) = \nabla g(w^*)$, which in turn implies that w^* is a local minima of J because $\nabla J(w^*) = 0$. ■

There are a number of extensions to CCP. We mention only a few in the passing. First, it can be shown that all instances of the EM algorithm (Section **??**) can be shown to be special cases of CCP. Second, the rate of con-

vergence of CCP is related to the eigenvalues of the positive semi-definite matrix $\nabla^2(f+g)$. Third, CCP can also be extended to solve constrained problems of the form:

$$\begin{aligned} \min_w \quad & f_0(w) - g_0(w) \\ \text{s.t.} \quad & f_i(w) - g_i(w) \leq c_i \text{ for } i = 1, \ldots, n. \end{aligned}$$

where, as before, f_i and g_i for $i = 0, 1, \ldots, n$ are assumed convex. At every iteration, we replace g_i by its first order Taylor approximation and solve the following constrained convex problem:

$$\begin{aligned} \min_w \quad & f_0(w) - g_0(w_t) + \langle w - w_t, \nabla g_0(w_t) \rangle \\ \text{s.t.} \quad & f_i(w) - g_i(w_t) + \langle w - w_t, \nabla g_i(w_t) \rangle \leq c_i \text{ for } i = 1, \ldots, n. \end{aligned}$$

5.6 Some Practical Advice

The range of optimization algorithms we presented in this chapter might be somewhat intimidating for the beginner. Some simple rules of thumb can alleviate this anxiety

Code Reuse: Implementing an efficient optimization algorithm correctly is both time consuming and error prone. Therefore, as far as possible use existing libraries. A number of high class optimization libraries both commercial and open source exist.

Unconstrained Problems: For unconstrained minimization of a smooth convex function LBFGS (Section 5.2.6.1 is the algorithm of choice. In many practical situations the spectral gradient method (Section 5.2.6.2) is also very competitive. It also has the added advantage of being easy to implement. If the function to be minimized is non-smooth then Bundle methods (Section 5.2.7) are to be preferred. Amongst the different formulations, the Bundle Trust algorithm tends to be quite robust.

Constrained Problems: For constrained problems it is very important to understand the nature of the constraints. Simple equality ($Ax = b$) and box ($l \leq x \leq u$) constraints are easier to handle than general non-linear constraints. If the objective function is smooth, the constraint set Ω is simple, and orthogonal projections P_Ω are easy to compute, then spectral projected gradient (Section 5.3.1) is the method of choice. If the optimization problem is a QP or an LP then specialized solvers tend to be much faster than general purpose solvers.

Large Scale Problems: If your parameter vector is high dimensional then consider coordinate descent (Section 5.2.2) especially if the one dimensional line search along a coordinate can be carried out efficiently. If the objective function is made up of a summation of large number of terms, consider stochastic gradient descent (Section 5.4.1). Although both these algorithms do not guarantee a very accurate solution, practical experience shows that for large scale machine learning problems this is rarely necessary.

Duality: Sometimes problems which are hard to optimize in the primal may become simpler in the dual. For instance, if the objective function is strongly convex but non-smooth, its Fenchel conjugate is smooth with a Lipschitz continuous gradient.

Problems

Problem 5.1 (Intersection of Convex Sets {1}) *If C_1 and C_2 are convex sets, then show that $C_1 \cap C_2$ is also convex. Extend your result to show that $\bigcap_{i=1}^{n} C_i$ are convex if C_i are convex.*

Problem 5.2 (Linear Transform of Convex Sets {1}) *Given a set $C \subset \mathbb{R}^n$ and a linear transform $A \in \mathbb{R}^{m \times n}$, define $AC := \{y = Ax : x \in C\}$. If C is convex then show that AC is also convex.*

Problem 5.3 (Convex Combinations {1}) *Show that a subset of $\mathbb{R}^n$ is convex if and only if it contains all the convex combination of its elements.*

Problem 5.4 (Convex Hull {2}) *Show that the convex hull, $\operatorname{conv}(X)$ is the smallest convex set which contains X.*

Problem 5.5 (Epigraph of a Convex Function {2}) *Show that a function satisfies Definition 5.3 if, and only if, its epigraph is convex.*

Problem 5.6 *Prove the Jensen's inequality (5.6).*

Problem 5.7 (Strong convexity of the negative entropy {3}) *Show that the negative entropy (5.15) is 1-strongly convex with respect to the $\|\cdot\|_1$ norm on the simplex. Hint: First show that $\phi(t) := (t-1)\log t - 2\frac{(t-1)^2}{t+1} \geq 0$ for all $t \geq 0$. Next substitute $t = x_i/y_i$ to show that*

$$\sum_i (x_i - y_i) \log \frac{x_i}{y_i} \geq \|x - y\|_1^2 .$$

Problem 5.8 (Strongly Convex Functions {2}) *Prove 5.16, 5.17, 5.18 and 5.19.*

Problem 5.9 (Convex Functions with Lipschitz Continuous Gradient {2}) *Prove 5.22, 5.23, 5.24 and 5.25.*

Problem 5.10 (One Dimensional Projection {1}) *If $f : \mathbb{R}^d \to \mathbb{R}$ is convex, then show that for an arbitrary x and p in $\mathbb{R}^d$ the one dimensional function $\Phi(\eta) := f(x + \eta p)$ is also convex.*

Problem 5.11 (Quasi-Convex Functions {2}) *In Section 5.1 we showed that the below-sets of a convex function $X_c := \{x \,|\, f(x) \leq c\}$ are convex. Give a counter-example to show that the converse is not true, that is, there exist non-convex functions whose below-sets are convex. This class of functions is called Quasi-Convex.*

Problem 5.12 (Gradient of the p-norm {1}) *Show that the gradient of the p-norm (5.31) is given by (5.32).*

Problem 5.13 *Derive the Fenchel conjugate of the following functions*

$$f(x) = \begin{cases} 0 & \text{if } x \in C \\ \infty & \text{otherwise.} \end{cases} \quad \text{where } C \text{ is a convex set}$$

$$f(x) = ax + b$$

$$f(x) = \frac{1}{2} x^\top A x \text{ where } A \text{ is a positive definite matrix}$$

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

$$f(x) = \exp(x)$$

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

Problem 5.14 (Convergence of gradient descent {2}) *Suppose J has a Lipschitz continuous gradient with modulus L. Then show that Algorithm 5.2 with an inexact line search satisfying the Wolfe conditions (5.42) and (5.43) will return a solution w_t with $\|\nabla J(w_t)\| \leq \epsilon$ in at most $O(1/\epsilon^2)$ iterations.*

Problem 5.15 *Show that*

$$\frac{1 + \sum_{t=1}^{T} \frac{1}{t}}{\sum_{t=1}^{T} \frac{1}{\sqrt{t}}} \leq \frac{1}{\sqrt{T}}$$

Problem 5.16 (Coordinate Descent for Quadratic Programming {2}) *Derive a projection based method which uses coordinate descent to generate directions of descent for solving the following box constrained QP:*

$$\min_{w \in \mathbb{R}^n} \frac{1}{2} w^\top Q w + c^\top w$$
$$s.t.\ l \leq w \leq u.$$

You may assume that Q is positive definite and l and u are scalars.

Problem 5.17 (Dual of a LP {1}) *Show that the dual of the LP (5.122) is (5.115). In other words, we recover the primal by computing the dual of the dual.*