9.1 Introduction

Given the computational burden associated with computing the empirical risk minimizer $A^*$ in the case of binary classification with the “true” loss function $\varphi^*(u) = 1_{u \leq 0}$ (in general, an NP-hard problem), last time we introduced the notion of a convex loss surrogate $\varphi$ which yields a new optimization problem known as $\varphi$-risk minimization. We proved a relationship between the excess risks of the two problems (Zhang’s lemma), which holds for the set of “soft” classifiers $B_{soft} : \{a : x \in \mathbb{R}^d \rightarrow a(x) \in \mathbb{R}\}$. This result shows that we can safely restrict ourself to considering convex loss functions. If, in addition, the set of admissible “soft” classifiers $A_{soft} \subseteq B_{soft}$ is convex, then we are left with a convex problem that takes the general form

$$\begin{array}{ll}
\text{minimize} & f(x) \\
\text{subject to} & x \in C
\end{array}$$

(9.1)

where $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is a convex function and $C \subseteq \mathbb{R}^d$ is a convex set. While the requirement of convexity for the loss function is well motivated by Zhang’s lemma, the requirement of convexity for the constraint set is motivated by computational convenience (so that the overall problem is convex, hence typically amenable to computations) and by the presence of illustrious examples, such as linear functions with convex parameter space and majority vote classifiers. At the same time, many popular soft classifiers do not form a convex set. Such is the case for neural networks, for instance, which do not form a convex set due to the non-linearities in their definition (cf. feed-forward neural networks, Section 3.5, for instance).

We now focus on convex problems of the form (9.1), and show how convexity, along with other geometrical properties such as strong convexity, smoothness, and Lipschitz continuity, can be used to both design algorithms and establish convergence guarantees. In the case of linear predictors with a bounded $\ell_2$ parameter space, we show how projected gradient descent yields a rate of convergence of the same order as the statistical rate of convergence established in Lecture 3. This fact immediately suggests a principled approach to obtain computational savings, as originally proposed in [1]. Henceforth, denote by $x^*$ any minima of (9.1).

9.2 Oracle Model

In the oracle model of computation, we assume that we can make queries to an oracle and receive some information about the function $f$ as output. Of particular interest to us are first order oracles, which take a point $x \in C$ as input and outputs a subgradient of $f$ at $x$. We are interested in understanding the oracle complexity of convex optimization, i.e., the number of necessary and sufficient queries to the oracle to find an $\epsilon$-approximate minima.

9.3 Projected Subgradient Descent

Let us consider problem (9.1). If $f$ is differentiable and $C = \mathbb{R}^d$, then the oracle at $x$ will return the gradient $\nabla f(x)$ (a local quantity, defined in terms of derivatives) which, by the requirement of convexity of $f$, is a
subgradient of $f$ and $x$ and, as such, it provides a hyperplane that uniformly bounds from below the function $f$:

$$f(y) \geq f(x) + \nabla f(x)^	op (y - x) \quad \text{for any } y \in C.$$ 

This fact suggests an algorithm to find a global minima of $f$: if we are at $x$, we can move, by a certain amount, in the direction opposite to the direction of the gradient. The basic gradient descent update reads

$$x_{s+1} = x_s - \eta \nabla f(x_s),$$

for some initial point $x_1 \in \mathbb{R}^d$ and step sizes $\eta_1, \eta_2, \ldots > 0$ (a.k.a. learning rates when an optimization routine is applied to learning problems). The question is then how to tune $\eta_s$. We expect a trade-off: we want the value of $\eta$ to be large, so to converge quickly to the solution; on the other hand, we do not want $\eta$ to be too large, otherwise we might miss the minimum and keep “oscillating” from one location to the other.

It turns out that if the convex function $f$ has any combination of the geometrical properties mentioned above (strong convexity, smoothness, Lipschitz continuity) and if we know the corresponding parameters associated to these properties, then we can provide explicit tuning for the choice of $\eta_s$ that guarantees convergence, and we can establish upper (and lower) bounds on the convergence rates. Along with depending on geometrical parameters, the learning rate will either (i) depend on the fixed amount of time we want to run the algorithm for (to be chosen a priori), or (ii) will be a decreasing function of time, typically decreasing as $1/\sqrt{t}$ or $1/t$. These are the two main mechanisms that allow to get provable convergence to a minimum, avoiding the oscillation behavior described above when $\eta$ is “too large”.

As in general we are interested in cases where the constraint set is a strict subset of $\mathbb{R}^d$ (recall the example of linear classifiers with convex parameters, or the majority vote) and the function $f$ may not be differentiable (recall the hinge loss), we look at projected subgradient descent methods instead. It is a simple generalization of the procedure described above. At each iteration, we query the oracle at $x$ and receive a subgradient $g$ of $f$ evaluated at $x$. Then, we take a step in the direction opposite to $g$, by a certain amount (controlled by the step function $\eta$). Finally, we project the new point we obtain back onto the constraint set $C$. We therefore begin by defining the projection operator and describe one of its main properties that we will need below.

Henceforth, we assume the constraint set $C$ to be compact and convex.

**Definition 9.1 (Projection operator)** For all $y \in \mathbb{R}^d$, the projection operator on $C$, $\Pi_C$, is defined by

$$\Pi_C(y) = \arg\min_{x \in C} \|x - y\|_2.$$

**Proposition 9.2 (Non-expansivity)** Let $x \in C$ and $y \in \mathbb{R}^d$. Then,

$$(\Pi_C(y) - x)^	op (\Pi_C(y) - y) \leq 0,$$

which implies $\|\Pi_C(y) - x\|_2^2 + \|y - \Pi_C(y)\|_2^2 \leq \|y - x\|_2^2$ and, in particular,

$$\|\Pi_C(y) - x\|_2 \leq \|y - x\|_2.$$

**Proof:** This is a direct consequence of Proposition 8.8 since $\Pi_C(y)$ is a minimizer of $h_y(z) = \|y - z\|_2$, and $\nabla h_y(z) = (z - y)/\|z - y\|_2$. ■
Figure 9.1: Representation of Proposition 9.2. From [2].

Algorithm 1: Projected Subgradient Descent

| Input: $x_1, \{\eta_s\}_{s \geq 1}$, stopping time $t$; for $s = 1, \ldots, t$ do
| $\tilde{x}_{s+1} = x_s - \eta_s g_s$, where $g_s \in \partial f(x_s)$,
| $x_{s+1} = \Pi_C(\tilde{x}_{s+1})$. |

Figure 9.2: Representation of projected gradient descent. From [2].

We state and prove the convergence of this algorithm under different assumptions.

9.3.1 Lipschitz functions

We first address the case of $\gamma$-Lipschitz objective functions. We provide results that hold both in the case of a fixed step size (when the step size depends on the total number of iterations $t$ to be executed, chosen
and fixed a priori, and the convergence guarantees only hold if the algorithm is run for that precise amount of time steps) and in the case of an adaptive step size (when the step size decreases with time \( t \), and the runtime of the algorithm does not need to be chosen a priori).

**Theorem 9.3 (Projected subgradient descent—Lipschitz)** Let \( f \) be convex. For every \( x \in \mathcal{C} \), let \( g \in \partial f(x) \) satisfy \( \|g\|_2 \leq \gamma \). Assume \( \|x_1 - x^*\|_2 \leq b \). Then, projected subgradient descent with \( \eta_s = \frac{b}{\gamma \sqrt{t}} \) satisfies

\[
\begin{align*}
f\left( \frac{1}{t} \sum_{s=1}^{t} x_s \right) - f(x^*) &\leq \frac{\eta}{\sqrt{t}} \text{ and } f(x^o) - f(x^*) \leq \frac{\gamma b}{\sqrt{t}} \tag{9.2}
\end{align*}
\]

where \( x^o \in \text{argmin}_{x \in \{x_1, \ldots, x_t\}} f(x) \). Moreover, if \( \eta_s = \frac{b}{\gamma \sqrt{s}} \), then

\[
\begin{align*}
f\left( \left( \sum_{s=\lceil t/2 \rceil + 1}^{t} \eta_s \right)^{-1} \sum_{s=\lceil t/2 \rceil + 1}^{t} \eta_s x_s \right) - f(x^*) &\leq c \frac{\eta b}{\sqrt{t}} \text{ and } f(x^o) - f(x^*) \leq c \frac{\gamma b}{\sqrt{t}} \tag{9.3}
\end{align*}
\]

with \( c = 2(1 + \log 2) \).

**Proof:** By convexity we have

\[
f\left( \frac{1}{t} \sum_{s=1}^{t} x_s \right) - f(x^*) \leq \frac{1}{t} \sum_{s=1}^{t} f(x_s) - f(x^*)
\]

and

\[
f(x_s) - f(x^*) \leq g_s^T (x_s - x^*).
\]

Recall that \( 2a^T b = \|a\|_2^2 + \|b\|_2^2 - \|a - b\|_2^2 \) and note that we can write \( g_s = \frac{1}{\eta} (x_s - \tilde{x}_{s+1}) \). We find, for any \( s \in [t] \),

\[
g_s^T (x_s - x^*) = \frac{1}{\eta} (x_s - \tilde{x}_{s+1})^T (x_s - x^*)
\]

\[
= \frac{1}{2\eta} \left( \|x_s - x^*\|_2^2 + \|x_s - \tilde{x}_{s+1}\|_2^2 - \|\tilde{x}_{s+1} - x^*\|_2^2 \right)
\]

\[
= \frac{1}{2\eta} \left( \|x_s - x^*\|_2^2 - \|\tilde{x}_{s+1} - x^*\|_2^2 \right) + \frac{\eta}{2} \|g_s\|_2^2
\]

\[
\leq \frac{1}{2\eta} \left( \|x_s - x^*\|_2^2 - \|x_{s+1} - x^*\|_2^2 \right) + \frac{\eta}{2} \|g_s\|_2^2,
\]

where for the last inequality we used that \( \|\tilde{x}_{s+1} - x^*\|_2 \geq \|x_{s+1} - x^*\|_2 \) by Proposition 9.2. Therefore, summing from \( s = 1 \) to \( t \), we get

\[
f\left( \frac{1}{t} \sum_{s=1}^{t} x_s \right) - f(x^*) \leq \frac{1}{2\eta t} \left( \|x_1 - x^*\|_2^2 - \|x_{t+1} - x^*\|_2^2 \right) + \frac{\eta^2}{2} \leq \frac{b^2}{2\eta t} + \frac{\eta^2}{2}.
\]

Minimizing the right-hand side of the above inequality one finds \( \eta = \frac{b}{\gamma \sqrt{t}} \) which yields (9.2). Clearly, \( f(x^o) \leq \frac{1}{t} \sum_{s=1}^{t} f(x_s) \), proving the second result of (9.2).

We now consider the case of an adaptive step size \( \eta_s \). The above derivation yields

\[
f(x_s) - f(x^*) \leq g_s^T (x_s - x^*) \leq \frac{1}{2\eta_s} \left( \|x_s - x^*\|_2^2 - \|\tilde{x}_{s+1} - x^*\|_2^2 \right) + \frac{\eta_s}{2} \|g_s\|_2^2.
\]
If we want to apply the same argument as above and get a telescoping sum with cancellations, we need to take a weighted sum weighted by \( \eta_s \). Namely, replacing \( t^{-1} \sum_{s=1}^t \) with \( (\sum_{s=1}^t \eta_s)^{-1} \sum_{s=1}^t \eta_s \) we get

\[
\left( \sum_{s=1}^t \eta_s \right)^{-1} \sum_{s=1}^t \eta_s (f(x_s) - f(x^*)) \leq \left( \sum_{s=1}^t \eta_s \right)^{-1} \left( b^2 \left( \sqrt{\frac{2}{\beta}} \right)^2 \right),
\]

which reduces to the previous result when \( \eta_s \equiv 1 \). We want the right-hand-side to go to zero as \( t \) increases, so we need both \( \sum \eta_s \to \infty \) and \( \frac{\eta_s}{t} \to 0 \). This is the case if we take \( \eta_s = \frac{1}{\sqrt{s}} \), as \( \sum_{s=1}^t \eta_s \geq c_1 k \sqrt{t} \) (e.g., \( c_1 = 1 \)) and \( \sum_{s=1}^t \eta_s^2 \leq c_2 k^2 \log t \) (e.g., \( c_2 = 1 + 1/\log 2 \) if \( t \geq 2 \), using that \( \sum_{s=1}^t 1/s \leq 1 + \int_{t/2}^t \frac{1}{s} ds = 1 + \log t \leq c_2 \log t \)). We can then choose \( k = \frac{\sqrt{2}}{\beta} \) such that

\[
\left( \sum_{s=1}^t \eta_s \right)^{-1} \sum_{s=1}^t \eta_s (f(x_s) - f(x^*)) \leq \frac{b^2}{2c_1 k \sqrt{t}} + \frac{c_2 k^2 \log t}{2c_1 \sqrt{t}} \leq \left( \frac{1 + c_2}{2c_1} \right) \gamma b \sqrt{\frac{\log t}{t}}.
\]

To get rid of the \( \log \) term, we note that if we only sum from \( \lceil t/2 \rceil + 1 \) to \( t \), for \( t \geq 3 \), we have \( \sum_{s=\lceil t/2 \rceil + 1}^t \eta_s \geq c_1' k \sqrt{t} \) (e.g., \( c_1' = 1/4 \)), as \( \sum_{s=\lceil t/2 \rceil + 1}^t 1/\sqrt{s} \geq \frac{1}{\sqrt{2} c_1'} \left( 1 - 1/t \right) \geq \frac{c_1'}{\sqrt{2}} \) and \( \sum_{s=\lceil t/2 \rceil}^t \eta_s^2 \leq c_2' k^2 \) (e.g., \( c_2' = \log 2 \) using that \( \sum_{s=\lceil t/2 \rceil}^t \eta_s^2 \leq \frac{2}{\sqrt{3}} \int_{t/2}^t \frac{1}{s} ds = \log 2 \)). Therefore, we finally have that

\[
\min_{1 \leq s \leq t} f(x_s) - f(x^*) \leq \min_{[\lceil t/2 \rceil + 1 \leq s \leq t]} f(x_s) - f(x^*) \leq \left( \sum_{s=\lceil t/2 \rceil + 1}^t \eta_s \right)^{-1} \sum_{s=\lceil t/2 \rceil + 1}^t \eta_s (f(x_s) - f(x^*)) \leq \frac{c b}{\sqrt{t}}
\]

with \( c = \left( \frac{1 + c_2'}{2c_1'} \right) \), where we used that the minimum element of a set is always less or equal to a convex combination of the elements.

The results in Theorem 9.3 demonstrate that the projected subgradient method for convex and \( \gamma \)-Lipschitz functions exhibits a convergence rate of \( O(\sqrt{\frac{\log t}{t}}) \). We can also phrase this in terms of oracle complexity: as at each iteration we make a constant (in fact, one) number of calls to the oracle, then to achieve \( \epsilon \)-convergence, we require \( \frac{b\sqrt{t}}{\epsilon} \leq \epsilon \), or \( t \geq \frac{\epsilon^2 \gamma^2}{\epsilon^2} \).

### 9.3.2 Smooth Functions

In the case of Lipschitz functions, subgradient descent converges slowly, with the rate \( 1/\sqrt{t} \). We will now see that if the function is smooth, this rate can be improved to \( 1/t \). Recall that a function \( f : \mathbb{R}^d \to \mathbb{R} \) is \( \beta \)-smooth if at any point \( x \in \mathcal{C} \) there exists a quadratic function that uniformly upper-bounds the function:

\[
f(y) \leq f(x) + \nabla f(x) (y - x) + \frac{\beta}{2} \|y - x\|^2 \quad \text{for any } y \in \mathbb{R}^d.
\]

(9.4)

This property immediately suggests an iterative algorithm to minimize \( f \). If the algorithm is currently at \( x_s \), we can compute the next point \( x_{s+1} \) by minimizing the right-hand side of the bound in (9.4) with respect to \( y \in \mathcal{C} \), with \( x = x_s \). Proceeding in this way, we find that the next point is given by

\[
x_{s+1} = \Pi_C \left( x_s - \frac{1}{\beta} \nabla f(x_s) \right),
\]

(9.5)

which corresponds precisely to a step of projected gradient descent with the choice \( \eta = 1/\beta \). This follows as

\[
\arg\min_{y \in \mathcal{C}} \left\{ f(x) + \nabla f(x) (y - x) + \frac{\beta}{2} \|y - x\|^2 \right\} = \arg\min_{y \in \mathcal{C}} \left\{ \left\| \left( x - \frac{1}{\beta} \nabla f(x) \right) - y \right\|^2 \right\} = \Pi_C \left( x - \frac{1}{\beta} \nabla f(x) \right).
\]
Therefore, we see that projected gradient descent is defined as the algorithm that at each time step moves to the point in $C$ that maximizes the guaranteed local decrease given by the quadratic function that, by smoothness, uniformly upper-bounds the function $f$ and supports it at the current location. See the illustration in Figure 9.3.2.

If $C = \mathbb{R}^d$ and we are at $x_s$, then the next move is $x_{s+1} = x_s - \frac{1}{\beta} \nabla f(x_s)$ and the guaranteed progress achieved by one step of gradient descent is given by (plugging $x_s$ and $x_{s+1}$ into (9.4))

$$f(x_{s+1}) \leq f(x_s) - \frac{1}{2\beta} \|\nabla f(x_s)\|^2$$

(9.6)

If $C \subseteq \mathbb{R}^d$ is a generic compact convex set $C$ and we are at $x_s \in C$, then the next move is $x_{s+1} = \Pi_C(\tilde{x}_{s+1})$ with $\tilde{x}_{s+1} = x_s - \frac{1}{\beta} \nabla f(x_s)$ and the guaranteed progress achieved by one step of gradient descent can be bounded as follows,

$$f(x_{s+1}) \leq f(x_s) + \nabla f(x_s)^\top (\Pi_C(\tilde{x}_{s+1}) - x_s) + \frac{\beta}{2} \|\Pi_C(\tilde{x}_{s+1}) - x_s\|^2$$

$$\leq f(x_s) + \beta (x_s - \Pi_C(\tilde{x}_{s+1}))^\top (\Pi_C(\tilde{x}_{s+1}) - x_s) + \frac{\beta}{2} \|\Pi_C(\tilde{x}_{s+1}) - x_s\|^2$$

$$\leq f(x_s) - \frac{\beta}{2} \|\Pi_C(\tilde{x}_{s+1}) - x_s\|^2,$$

(9.7)

where we used Proposition 9.2, as the inequality $\nabla f(x_s)^\top (\Pi_C(\tilde{x}_{s+1}) - x_s) \leq \beta (x_s - \Pi_C(\tilde{x}_{s+1}))^\top (\Pi_C(\tilde{x}_{s+1}) - x_s)$ can be rearranged into $(\nabla f(x_s) - \beta (x_s - \Pi_C(\tilde{x}_{s+1})))^\top (\Pi_C(\tilde{x}_{s+1}) - x_s) \leq 0$, and this is equivalent to (multiplying by $1/\beta$ and using that $\tilde{x}_{s+1} = x_s - \nabla f(x_s)/\beta$) $(\Pi_C(\tilde{x}_{s+1}) - \tilde{x}_{s+1})^\top (\Pi_C(\tilde{x}_{s+1}) - x_s) \leq 0$.

The following proposition is proved using the bound on the guaranteed progress (9.7), along with Proposition 9.2 on the projection operator.

**Theorem 9.4 (Projected gradient descent—Smooth)** Let $f$ be convex and $\beta$-smooth on $C$. Then, projected gradient descent with $\eta = \frac{1}{\beta}$ satisfies

$$f(x_t) - f(x^*) \leq \frac{3\beta \|x_1 - x^*\|^2 + f(x_1) - f(x^*)}{t}$$

(9.8)
**Proof:** By smoothness and convexity, we have

\[
    f(x_{s+1}) - f(x^*) = f(x_{s+1}) - f(x_s) + f(x_s) - f(x^*) \\
    \leq \nabla f(x_s)^T (x_{s+1} - x_s) + \frac{\beta}{2} \|x_{s+1} - x_s\|^2 + \nabla f(x_s)^T (x_s - x^*) \\
    = \nabla f(x_s)^T (x_{s+1} - x^*) + \frac{\beta}{2} \|x_{s+1} - x_s\|^2.
\]  

(9.8)

By Proposition 9.2 we have

\[
    \nabla f(x_s)^T (x_{s+1} - x^*) \leq \beta(x_s - x_{s+1})^T (x_{s+1} - x^*).
\]

(9.9)

In fact, this inequality can be rearranged into \((\nabla f(x_s) - \beta(x_s - x_{s+1}))^T (x_{s+1} - x^*) \leq 0\), which is equivalent to (multiplying by \(1/\beta\) and using that \(x_{s+1} = \Pi_C(\tilde{x}_s)\) with \(\tilde{x}_s = x_s - \nabla f(x_s)/\beta\) \((\Pi_C(\tilde{x}_s) - \tilde{x}_s)^T (\Pi_C(\tilde{x}_s) - x^*) \leq 0\).

Plugging (9.9) into (9.8) and using the Cauchy-Schwarz's inequality we obtain

\[
    f(x_{s+1}) - f(x^*) \leq \beta(x_s - x_{s+1})^T (x_{s+1} - x^*) + \frac{\beta}{2} \|x_{s+1} - x_s\|^2 \\
    = \beta(x_s - x_{s+1})^T (x_s - x^*) - \frac{\beta}{2} \|x_{s+1} - x_s\|^2 \\
    \leq \beta \|x_s - x_{s+1}\|_2 \|x_s - x^*\|_2.
\]

(9.10)

At the same time, (9.7) reads \(f(x_{s+1}) - f(x_s) \leq -\frac{\delta_s}{2} \|x_{s+1} - x_s\|^2\), which can be written as \(\|x_{s+1} - x_s\|^2 \leq f(x_{s+1}) - f(x_s)\). Combining these two inequalities, defining \(\delta_s := x_s - x^*\), we obtain

\[
    \delta_{s+1} \leq \delta_s - \frac{\delta_s^2}{2\beta \|x_s - x^*\|^2}.
\]

(9.11)

Note that \(\|x_s - x^*\|_2\) is a non-increasing function of \(s\). In fact, by (9.10) we have

\[
    (x_s - x_{s+1})^T (x_s - x^*) \geq \frac{1}{2} \|x_{s+1} - x_s\|^2,
\]

and thus

\[
    \|x_{s+1} - x^*\|^2 = \|x_{s+1} - x_s + x_s - x^*\|^2 \\
    = \|x_{s+1} - x_s\|^2 + \|x_s - x^*\|^2 + 2(x_{s+1} - x_s)^T (x_s - x^*) \\
    \leq \|x_s - x^*\|^2.
\]

Therefore, equation (9.11) yields

\[
    \delta_{s+1} \leq \delta_s - \frac{\delta_s^2}{2\beta \|x_s - x^*\|^2},
\]

and, by induction, we obtain

\[
    \delta_s \leq \frac{3\beta \|x_1 - x^*\|^2 + f(1) - f(x^*)}{s}.
\]

While gradient descent is a natural algorithm in the case of smooth functions, as we described above, the proof of Theorem 9.4 is not immediate. In general, both designing first-order methods that can naturally reflect the problem structure and establishing a neat analysis of their performance is still an active topic of research in convex optimization. We will now show that if a function is both smooth and strongly convex, then a simple analysis can be established in the unconstrained case, leading to an exponential rate of convergence (for historical reasons also called a linear rate in the optimization literature).
9.3.3 Smooth and Strongly Convex Functions

We now consider the case of a function \( f \) that is both smooth and strongly convex, and take \( C = \mathbb{R}^d \). As the function is smooth, the argument presented in Section 9.3.2 still applies and should take gradient descent with the choice \( \eta = 1/\beta \) if we want to maximize the guaranteed progress at each step. In this case, (9.5) reads

\[
x_{s+1} = x_s - \frac{1}{\beta} \nabla f(x_s).
\]

Recall that a function \( f: \mathbb{R}^d \rightarrow \mathbb{R} \) is \( \alpha \)-strongly convex if at any point \( x \in C \) there exists a quadratic function that uniformly lower-bounds the function:

\[
f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{\alpha}{2} \|y - x\|^2_2 \quad \text{for any } y \in \mathbb{R}^d.
\] (9.12)

This property immediately suggests a way to infer a lower bound on the minimum of the objective function. If the algorithm is currently at \( x_s \), we can minimize the right-hand side of (9.12) and we find

\[
f(y) \geq f(x_s) - \frac{1}{2\alpha} \|\nabla f(x_s)\|^2_2 \quad \text{for any } y \in \mathbb{R}^d,
\]

which implies, in particular,

\[
f(x^*) \geq f(x_s) - \frac{1}{2\alpha} \|\nabla f(x_s)\|^2_2
\]

See Figure 9.3.2. The proof of the following result comes by using both the upper bound on \( f(x_{s+1}) \) given by smoothness in (9.6) and the lower bound on \( f(x^*) \) given by strong convexity in (9.3.3).

**Theorem 9.5 (Projected subgradient descent—Smooth and strongly convex)** Let \( f \) be \( \alpha \)-strongly convex and \( \beta \)-smooth on \( C = \mathbb{R}^d \). Then, gradient descent with \( \eta = 1/\beta \) satisfies

\[
|f(x_t) - f(x^*)| \leq \left(1 - \frac{\alpha}{\beta}\right)^{t-1} (f(x_1) - f(x^*))
\]

**Proof:** By rearranging (9.3.3) we find

\[
-\frac{1}{2} \|\nabla f(x_s)\|^2_2 \leq -\alpha(f(x_s) - f(x^*))
\]

which, once plugged into (9.6) yields

\[
f(x_{s+1}) \leq f(x_s) - \frac{1}{2\beta} \|\nabla f(x_s)\|^2_2 \leq f(x_s) - \frac{\alpha}{\beta} (f(x_s) - f(x^*))
\]

which yields \( f(x_{s+1}) - f(x^*) \leq (1 - \frac{\alpha}{\beta})(f(x_s) - f(x^*)) \).

The quantity \( \beta \alpha \geq 1 \) that controls the exponential convergence rate is called the **condition number** of the function \( f \). In the current setting, exponential convergence can also be proved for projected gradient descent. We refer the reader to Theorem 3.10 in [2].
9.4 Oracle Complexity, Lower Bounds, and Accelerated Methods

Upon proper tuning, projected subgradient descent methods achieve the following rates (see Theorem 3.9 in [2] for the strongly convex and Lipschitz case):

<table>
<thead>
<tr>
<th></th>
<th>( L )-Lipschitz</th>
<th>( \beta )-smooth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex</td>
<td>( O(\gamma b/\sqrt{t}) )</td>
<td>( O((\beta b^2 + c)/t) )</td>
</tr>
<tr>
<td>( \alpha )-strongly convex</td>
<td>( O(\gamma^2/(\alpha t)) )</td>
<td>( O(e^{-\alpha d^2/\beta}) )</td>
</tr>
</tbody>
</table>

where \( \|x_1 - x^*\|_2 \leq b \) and \( f(x_1) - f(x^*) \leq c \).

**Remark 9.6 (On the dimension-free nature of gradient descent)** These rates are sometimes called dimension-free, since, as presented, they do not depend explicitly on the ambient dimension \( d \). However, the dependence on the dimension \( d \) enters implicitly in the constants, either the ones defining the structural properties of the problem (\( \alpha, \beta, \gamma \)) or the ones defining the suboptimality of the initial conditions (\( b, c \)). The terminology dimension-free is typically used to differentiate the rates achieved by subgradient descent methods from the rates achieved by ellipsoid methods (see, Chapter 2 in [2], for instance).

We can also phrase these results in terms of oracle complexity, namely, the number of queries to the first order oracle that are sufficient to find a \( \varepsilon \)-approximate minimum:

<table>
<thead>
<tr>
<th></th>
<th>( L )-Lipschitz</th>
<th>( \beta )-smooth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex</td>
<td>( O(\gamma^2 b^2/\varepsilon^2) )</td>
<td>( O((\beta b^2 + c)/\varepsilon) )</td>
</tr>
<tr>
<td>( \alpha )-strongly convex</td>
<td>( O(\gamma^2/(\alpha \varepsilon)) )</td>
<td>( O((\beta/\alpha) \log (c/\varepsilon)) )</td>
</tr>
</tbody>
</table>

The first order oracle model is a convenient model of computation as it also allows us to compute lower bounds on the amount of calls to the oracle that are needed to achieve a certain accuracy \( \varepsilon \). The following lower bounds can be proved (see Theorem 3.13, Theorem 3.14, and Theorem 3.15 in [2]):

<table>
<thead>
<tr>
<th></th>
<th>( L )-Lipschitz</th>
<th>( \beta )-smooth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convex</td>
<td>( \Omega(\gamma a/(1 + \sqrt{t})) )</td>
<td>( \Omega(d^2 \beta/(t + 1)^2) )</td>
</tr>
<tr>
<td>( \alpha )-strongly convex</td>
<td>( \Omega(\gamma^2/(\alpha t)) )</td>
<td>( \Omega(ad^2 e^{-t \sqrt{\alpha/\beta}}) )</td>
</tr>
</tbody>
</table>

where \( a := \max_{x \in \mathcal{C}} \|x\|_2 \) is the radius of the smallest Euclidean ball that contains \( \mathcal{C} \) and \( d := \max_{x,y \in \mathcal{C}} \|x - y\|_2 \) is the diameter of \( \mathcal{C} \).

From the lower bound table, we see that among the results we proved, only the rate for convex and Lipschitz function is optimal. To achieve the other lower bounds, we need new algorithms, so-called accelerated. While different accelerated algorithms that meet the above-mentioned guarantees have been developed over the years, understanding the fundamental mechanisms on which acceleration relies is still a topic of research.

9.5 Back to Learning: Linear Predictors with \( \ell_2 \) Constraints

Let us now apply the previously-developed theory to the case of linear predictors over convex constraints. Let \( (X_1, Y_1), \ldots, (X_n, Y_n) \in \mathcal{X} \times \{-1, 1\} \) be the training data, with \( \mathcal{X} \subseteq \mathbb{R}^d \). Let \( \varphi : \mathbb{R} \rightarrow \mathbb{R}_+ \) be a given convex loss function. Let

\[
\mathcal{A}_2 = \{ x \in \mathbb{R}^d \to a(x) = w^T x : w \in \mathcal{W}_2 \subseteq \mathbb{R}^d \},
\]


where $W_2$ is a convex set contained in a ball of radius $c_2^W$, namely, $c_2^W := \max_{w \in W_2} \|w\|_2$.

The risk minimization problem reads

$$
\begin{align*}
\text{minimize} & \quad r(w) = E \varphi(w^\top X Y) \\
\text{subject to} & \quad w \in W_2
\end{align*}
$$

Let $w^*_2$ be a minimizer of this problem. The empirical risk minimization problem reads

$$
\begin{align*}
\text{minimize} & \quad R(w) = \frac{1}{n} \sum_{i=1}^n \varphi(w^\top X_i Y_i) \\
\text{subject to} & \quad w \in W_2
\end{align*}
$$

Let $W'_2$ be a minimizer of this problem. Note that we use the upper-case letter as the optimizer is a function of the training data, which are random variables.

Let us recall the error decomposition derived in Section 1.2.2, for any $W \in W$:

$$
E \sup_{w \in W_2} \{r(w) - R(w)\} \leq 2E \text{Rad}(L \circ \{Z_1, \ldots, Z_n}\} \leq 2\gamma_x E \text{Rad}(A_2 \circ \{X_1, \ldots, X_n}\} \leq 2\gamma_x c_2^W E \frac{\max_i \|X_i\|_2}{\sqrt{n}} 
$$

so that

$$
E \text{Statistics}_2 \leq \frac{4c_2^X c_2^W \gamma_x}{\sqrt{n}}
$$

Note that the empirical risk $R$ is $(c_2^X \gamma_x)$-Lipschitz, as

$$
|R(w) - R(u)| \leq \frac{1}{n} \sum_{i=1}^n |\varphi(w^\top X_i Y_i) - \varphi(u^\top X_i Y_i)| \leq \frac{\gamma_x}{n} \sum_{i=1}^n |Y_i(w - u)^\top X_i| \leq c_2^X \gamma_x \|w - u\|_2,
$$

where we used that $|Y_i| = 1$ and we used the Cauchy-Schwarz's inequality. Therefore, by applying projected subgradient descent with $\eta_s \equiv \eta = \frac{2c_2^W}{c_2^X \gamma_x \sqrt{t}}$ to problem (9.13) we have, by Theorem 9.3,

$$
\text{Optimization}_2 = R(W_t) - R(W'_2) \leq \frac{2c_2^X c_2^W \gamma_x}{\sqrt{t}}
$$

where $W_t := \frac{1}{t} \sum_{s=1}^t W_s$, where we used that $\|W_1 - W'_2\|_2 \leq \|W_1\|_2 + \|W'_2\|_2 \leq 2c_2^W$. 
We see that, as far as bounding the expected value of the estimation error \( r(\mathbf{w}_t) - r(w^*) \), we only need to run projected subgradient descent for an amount of time steps \( t \) that is of the same order as the amount of data points \( n \) if we want to match the statistical guarantees that we have established previously in this course. In other words, the analysis that we have developed so far shows that running the algorithm for \( t \sim n \) steps suffices to match the upper bounds on the optimization error and on the statistical error. This argument, originally put forward in [1], gives a principled approach to obtain computational savings in the training phase, as we can reliably stop the optimization algorithm after \( t \sim n \) time steps. At the same time, this analysis is just based on upper bounds, so one should be careful about drawing conclusions from it!

References
