Algorithmic Foundations of Learning **Lecture 9**

Oracle Model. Gradient Descent Methods

Lecturer: Patrick Rebeschini *Version: December 8, 2021*

9.1 Introduction

Given the computational burden associated with computing the empirical risk minimizer $A^\star$ in the case of binary classification with the "true" loss function $\varphi^\star(u) = \mathbf{1}_{u\leq 0}$ (in general, a NP-hard problem), last time we introduced the notion of a convex loss surrogate φ which yields a new optimization problem known as φ-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 $\mathcal{B}_{\text{soft}} : \{a : x \in \mathbb{R}^d \to a(x) \in \mathbb{R}\}$. This result shows that we can safely restrict ourself to considering convex loss functions. If, in addition, the set of admissible "soft" classifiers $\mathcal{A}_{\text{soft}} \subseteq \mathcal{B}_{\text{soft}}$ is convex, then we are left with a convex problem that takes the general form

$$\begin{aligned} &\underset{x}{\text{minimize}} && f(x) \\ &\text{subject to} && x \in \mathcal{C} \end{aligned} \tag{9.1}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is a convex function and $\mathcal{C} \subseteq \mathbb{R}^d$ 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 once additional assumptions hold) 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 local-to-global 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 ℓ_2 parameter space, we show how the projected gradient method 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, let us denote by $x^\star$ 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 $x \in \mathcal{C}$ 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 ε-approximate minima.

9.3 Projected Subgradient Method

Let us consider problem (9.1). If f is differentiable and $\mathcal{C} = \mathbb{R}^d$, 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:

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

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

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

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 η_s. We expect a trade-off: we want the value of η to be large, so to converge quickly to the solution; on the other hand, we do not want η 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 local-to-global 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 η_s that guarantees convergence, and we can establish upper (and lower) bounds on the convergence rates. Along with depending on those 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) 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 η 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 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 η). Finally, we project the new point onto the constraint set $\mathcal{C}$. 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 $\mathcal{C}$ to be compact and convex.

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

$$\Pi_{\mathcal{C}}(y) = \operatorname*{argmin}_{x \in \mathcal{C}} \|x - y\|_2.$$

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

$$\left(\Pi_{\mathcal{C}}(y) - x\right)^\top \left(\Pi_{\mathcal{C}}(y) - y\right) \leq 0,$$

which implies $\|\Pi_{\mathcal{C}}(y) - x\|_2^2 + \|y - \Pi_{\mathcal{C}}(y)\|_2^2 \leq \|y - x\|_2^2$ and, in particular,

$$\boxed{\|\Pi_{\mathcal{C}}(y) - x\|_2 \leq \|y - x\|_2}$$

Proof: This is a direct consequence of Proposition 8.10 since $\Pi_{\mathcal{C}}(y)$ is a minimizer of the function $z \to f_y(z) = \|y - z\|_2$, and $\nabla f_y(z) = (z - y)/\|z - y\|_2$. ■

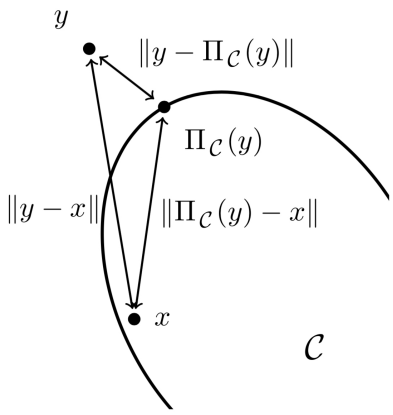

Figure 9.1: Representation of Proposition 9.2. From [2].

Algorithm 1: Projected Subgradient Method

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 $g_s \in \partial f(x_s)$,
  $x_{s+1} = \Pi_{\mathcal{C}}(\tilde{x}_{s+1})$.
end

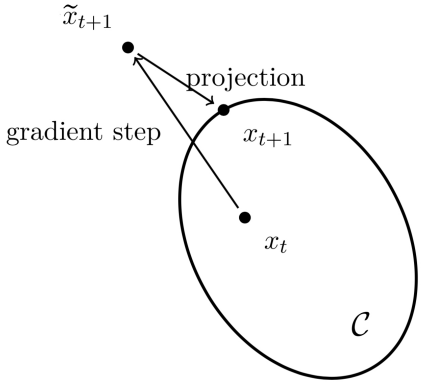

Figure 9.2: Representation of projected gradient descent. From [2].

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

9.3.1 Lipschitz functions

We first address the case of γ-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 method—Lipschitz) *Let f be convex. For every $x \in \mathcal{C}$, let $g \in \partial f(x)$ satisfy $\|g\|_2 \leq \gamma$. Assume $\|x_1 - x^\star\|_2 \leq b$. Then, the projected subgradient method with $\eta_s \equiv \eta = \frac{b}{\gamma\sqrt{t}}$ satisfies*

$$\boxed{f\left(\frac{1}{t}\sum_{s=1}^{t} x_s\right) - f(x^\star) \leq \frac{\gamma b}{\sqrt{t}} \quad \text{and} \quad f(x^o) - f(x^\star) \leq \frac{\gamma b}{\sqrt{t}}} \tag{9.2}$$

where $x^o \in \operatorname{argmin}_{x \in \{x_1, \ldots, x_t\}} f(x)$.
Now let $\sup_{x,y \in \mathcal{C}} \|x - y\|_2 \leq \tilde{b}$ (i.e., $\tilde{b}$ is an upper bound on the diameter of $\mathcal{C}$). If $\eta_s = \frac{\tilde{b}}{\gamma\sqrt{s}}$, then

$$\boxed{f\left(\left(\sum_{s=\lceil t/2\rceil+1}^{t} \eta_s\right)^{-1} \sum_{s=\lceil t/2\rceil+1}^{t} \eta_s x_s\right) - f(x^\star) \leq c\frac{\gamma\tilde{b}}{\sqrt{t}} \quad \text{and} \quad f(x^o) - f(x^\star) \leq c\frac{\gamma\tilde{b}}{\sqrt{t}}} \tag{9.3}$$

where $c = 2(1 + \log 2)$.

Proof: By convexity we have

$$f\left(\frac{1}{t}\sum_{s=1}^{t} x_s\right) - f(x^\star) \leq \frac{1}{t}\sum_{s=1}^{t} f(x_s) - f(x^\star)$$

and

$$f(x_s) - f(x^\star) \leq g_s^\top (x_s - x^\star).$$

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

$$\begin{aligned}
g_s^\top (x_s - x^\star) &= \frac{1}{\eta}(x_s - \tilde{x}_{s+1})^\top (x_s - x^\star) \\
&= \frac{1}{2\eta}\left(\|x_s - x^\star\|_2^2 + \|x_s - \tilde{x}_{s+1}\|_2^2 - \|\tilde{x}_{s+1} - x^\star\|_2^2\right) \\
&= \frac{1}{2\eta}\left(\|x_s - x^\star\|_2^2 - \|\tilde{x}_{s+1} - x^\star\|_2^2\right) + \frac{\eta}{2}\|g_s\|_2^2 \\
&\leq \frac{1}{2\eta}\left(\|x_s - x^\star\|_2^2 - \|x_{s+1} - x^\star\|_2^2\right) + \frac{\eta}{2}\|g_s\|_2^2,
\end{aligned}$$

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

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

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 η_s. The above derivation yields

$$f(x_s) - f(x^\star) \leq g_s^\top (x_s - x^\star) \leq \frac{1}{2\eta_s}\left(\|x_s - x^\star\|_2^2 - \|\tilde{x}_{s+1} - x^\star\|_2^2\right) + \frac{\eta_s}{2}\|g_s\|_2^2.$$

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 η_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

$$\left(\sum_{s=1}^t \eta_s\right)^{-1} \sum_{s=1}^t \eta_s \left(f(x_s) - f(x^\star)\right) \leq \left(\sum_{s=1}^t \eta_s\right)^{-1} \left(\frac{b^2}{2} + \left(\sum_{s=1}^t \eta_s^2\right)\frac{\gamma^2}{2}\right),$$

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

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

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 $\sum_{s=\lceil t/2\rceil+1}^t 1/\sqrt{s} \geq \frac{t-1}{2\sqrt{t}} = \frac{\sqrt{t}}{2}(1 - 1/t) \geq \frac{\sqrt{t}}{4}$) and $\sum_{s=\lceil t/2\rceil+1}^t \eta_s^2 \leq c_2' k^2$ (e.g., $c_2' = \log 2$ using that $\sum_{s=\lceil t/2\rceil+1}^t \frac{1}{s} \leq \int_{t/2}^t \frac{1}{s} ds = \log 2$). Therefore, we finally have that

$$\min_{1\leq s\leq t} f(x_s) - f(x^\star) \leq \min_{\lceil t/2\rceil+1 \leq s \leq t} f(x_s) - f(x^\star) \leq \left(\sum_{s=\lceil t/2\rceil+1}^t \eta_s\right)^{-1} \sum_{s=\lceil t/2\rceil+1}^t \eta_s \left(f(x_s) - f(x^\star)\right) \leq c\frac{\gamma\tilde{b}}{\sqrt{t}},$$

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 with a constant step size the projected subgradient method for convex and γ-Lipschitz functions exhibits a convergence rate of $O(\frac{b\gamma}{\sqrt{t}})$. 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 ε-convergence, we require $\frac{b\gamma}{\sqrt{t}} \leq \varepsilon$, or $t \geq \frac{b^2\gamma^2}{\varepsilon^2}$ iterations.

Note that the subgradient method is *not* a so-called *descent method*: the function value can (and often does) increase in one time step! This is why we can only bound the quantity $f(x^o) - f(x^\star)$, where $x^o \in \operatorname{argmin}_{x\in\{x_1,\ldots,x_t\}} f(x)$, instead of bounding the quantity $f(x_t) - f(x^\star)$ that refers to the last iterate of the method.

9.3.2 Smooth Functions

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

$$f(y) \leq f(x) + \nabla f(x)^\top (y - x) + \frac{\beta}{2}\|y - x\|_2^2 \qquad \text{for any } y \in \mathbb{R}^d. \tag{9.4}$$

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

$$x_{s+1} = \Pi_{\mathcal{C}}\left(x_s - \frac{1}{\beta}\nabla f(x_s)\right), \tag{9.5}$$

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

$$\operatorname*{argmin}_{y\in\mathcal{C}} \left\{ f(x) + \nabla f(x)^\top (y-x) + \frac{\beta}{2}\|y-x\|_2^2 \right\} = \operatorname*{argmin}_{y\in\mathcal{C}} \left\{ \left\| \left(x - \frac{1}{\beta}\nabla f(x) \right) - y \right\|_2^2 \right\} \equiv \Pi_{\mathcal{C}} \left(x - \frac{1}{\beta}\nabla f(x) \right).$$

Therefore, we see that the projected gradient method is *defined* as the algorithm that at each time step moves to the point in $\mathcal{C}$ 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 the function f is smooth, then the projected gradient method is a descent method, as we can guarantee that at each time step of the algorithm the function value does not increase. The guaranteed progress achieved by one step of gradient descent is bounded differently in the unconstrained and constrained case.

- If $\mathcal{C} = \mathbb{R}^d$ 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))

$$\boxed{f(x_{s+1}) \le f(x_s) - \frac{1}{2\beta}\|\nabla f(x_s)\|_2^2} \tag{9.6}$$

- If $\mathcal{C} \subseteq \mathbb{R}^d$ is a generic compact convex set $\mathcal{C}$ and we are at $x_s \in \mathcal{C}$, then the next move is $x_{s+1} = \Pi_{\mathcal{C}}(\tilde{x}_{s+1})$ 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,

$$\begin{aligned} f(x_{s+1}) &\le f(x_s) + \nabla f(x_s)^\top (\Pi_{\mathcal{C}}(\tilde{x}_{s+1}) - x_s) + \frac{\beta}{2}\|\Pi_{\mathcal{C}}(\tilde{x}_{s+1}) - x_s\|_2^2 \\ &\le f(x_s) + \beta(x_s - \Pi_{\mathcal{C}}(\tilde{x}_{s+1}))^\top (\Pi_{\mathcal{C}}(\tilde{x}_{s+1}) - x_s) + \frac{\beta}{2}\|\Pi_{\mathcal{C}}(\tilde{x}_{s+1}) - x_s\|_2^2, \end{aligned}$$

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

$$\boxed{f(x_{s+1}) \le f(x_s) - \frac{\beta}{2}\|\Pi_{\mathcal{C}}(\tilde{x}_{s+1}) - x_s\|_2^2}, \tag{9.7}$$

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 β-smooth on $\mathcal{C}$. Then, projected gradient descent with $\eta = \frac{1}{\beta}$ satisfies*

$$\boxed{f(x_t) - f(x^\star) \le \frac{3\beta\|x_1 - x^\star\|_2^2 + f(x_1) - f(x^\star)}{t}}$$

Proof: By smoothness and convexity, we have

$$\begin{aligned} f(x_{s+1}) - f(x^\star) &= f(x_{s+1}) - f(x_s) + f(x_s) - f(x^\star) \\ &\le \nabla f(x_s)^\top (x_{s+1} - x_s) + \frac{\beta}{2}\|x_{s+1} - x_s\|_2^2 + \nabla f(x_s)^\top (x_s - x^\star) \\ &= \nabla f(x_s)^\top (x_{s+1} - x^\star) + \frac{\beta}{2}\|x_{s+1} - x_s\|_2^2. \end{aligned} \tag{9.8}$$

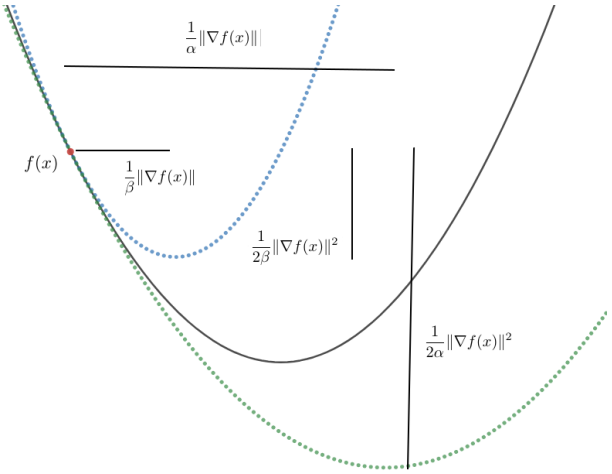

Figure 9.3: Visualization of the smoothness bound (blue) and the strong convexity bound (green) of a function f (black).

By Proposition 9.2 we have

$$\nabla f(x_s)^\top (x_{s+1} - x^\star) \leq \beta (x_s - x_{s+1})^\top (x_{s+1} - x^\star). \tag{9.9}$$

In fact, this inequality can be rearranged into $(\nabla f(x_s) - \beta(x_s - x_{s+1}))^\top (x_{s+1} - x^\star) \leq 0$, which is equivalent to (multiplying by $1/\beta$ and using that $x_{s+1} = \Pi_{\mathcal{C}}(\tilde{x}_s)$ with $\tilde{x}_s = x_s - \nabla f(x_s)/\beta$) $(\Pi_{\mathcal{C}}(\tilde{x}_s) - \tilde{x}_s)^\top (\Pi_{\mathcal{C}}(\tilde{x}_s) - x^\star) \leq 0$. Plugging (9.9) into (9.8) and using the Cauchy-Schwarz's inequality we obtain

$$\begin{aligned} f(x_{s+1}) - f(x^\star) &\leq \beta (x_s - x_{s+1})^\top (x_{s+1} - x^\star) + \frac{\beta}{2}\|x_{s+1} - x_s\|_2^2 \\ &= \beta (x_s - x_{s+1})^\top (x_s - x^\star) - \frac{\beta}{2}\|x_{s+1} - x_s\|_2^2 \\ &\leq \beta \|x_s - x_{s+1}\|_2 \|x_s - x^\star\|_2. \end{aligned} \tag{9.10}$$

which can be rewritten as

$$\|x_s - x_{s+1}\|_2 \geq \frac{f(x_{s+1}) - f(x^\star)}{\beta \|x_s - x^\star\|_2}.$$

At the same time, (9.7) reads $f(x_{s+1}) - f(x_s) \leq -\frac{\beta}{2}\|x_{s+1} - x_s\|_2^2$. Combining these two inequalities, defining $\delta_s := f(x_s) - f(x^\star)$, we obtain

$$\delta_{s+1} - \delta_s = f(x_{s+1}) - f(x_s) \leq -\frac{\beta}{2}\|x_{s+1} - x_s\|_2^2 \leq -\frac{\beta}{2}\left(\frac{\delta_{s+1}}{\beta \|x_s - x^\star\|_2}\right)^2$$

which reads

$$\delta_{s+1} \leq \delta_s - \frac{\delta_{s+1}^2}{2\beta \|x_s - x^\star\|_2^2}. \tag{9.11}$$

Note that $\|x_s - x^\star\|_2^2$ is a non-increasing function of s. In fact, by (9.10) we have

$$\beta (x_s - x_{s+1})^\top (x_s - x^\star) - \frac{\beta}{2}\|x_{s+1} - x_s\|_2^2 \geq 0,$$

that is,

$$(x_s - x_{s+1})^\top (x_s - x^\star) \geq \frac{1}{2}\|x_{s+1} - x_s\|_2^2,$$

and thus

$$\begin{aligned}\|x_{s+1} - x^\star\|_2^2 &= \|x_{s+1} - x_s + x_s - x^\star\|_2^2 \\ &= \|x_{s+1} - x_s\|_2^2 + \|x_s - x^\star\|_2^2 + 2(x_{s+1} - x_s)^\top (x_s - x^\star) \\ &\leq \|x_s - x^\star\|_2^2.\end{aligned}$$

Thus, the inequality (9.11) yields

$$\delta_{s+1} \leq \delta_s - \frac{\delta_{s+1}^2}{2\beta\|x_1 - x^\star\|_2^2},$$

and, by induction, we obtain

$$\delta_s \leq \frac{3\beta\|x_1 - x^\star\|_2^2 + f(x_1) - f(x^\star)}{s}.$$

■

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 (also called a *linear* rate in the optimization literature, as the rate is linear in log terms).

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 $\mathcal{C} = \mathbb{R}^d$. Also in this case the gradient method is a descent method. 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 $f : \mathbb{R}^d \to \mathbb{R}$ is α-strongly convex if at any point $x \in \mathcal{C}$ there exists a quadratic function that uniformly lower-bounds the function:

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{\alpha}{2}\|y - x\|_2^2 \qquad \text{for any } y \in \mathbb{R}^d. \tag{9.12}$$

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

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

which implies, in particular,

$$\boxed{f(x^\star) \geq f(x_s) - \frac{1}{2\alpha}\|\nabla f(x_s)\|_2^2} \tag{9.13}$$

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 $f(x^\star)$ given by strong convexity in (9.13).

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

$$\boxed{f(x_t) - f(x^\star) \leq \left(1 - \frac{\alpha}{\beta}\right)^{t-1} (f(x_1) - f(x^\star))}$$

Proof: By rearranging (9.13) we find

$$-\frac{1}{2}\|\nabla f(x_s)\|_2^2 \leq -\alpha(f(x_s) - f(x^\star))$$

which, once plugged into (9.6) yields

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

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

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):

	L-Lipschitz	β-smooth
Convex	$O(\gamma b/\sqrt{t})$	$O((\beta b^2 + c)/t)$
α-strongly convex	$O(\gamma^2/(\alpha t))$	$O(e^{-t\alpha/\beta} c)$

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

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 (α, β, γ) 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 ε-approximate minimum:

	L-Lipschitz	β-smooth
Convex	$O(\gamma^2 b^2/\varepsilon^2)$	$O((\beta b^2 + c)/\varepsilon)$
α-strongly convex	$O(\gamma^2/(\alpha\varepsilon))$	$O((\beta/\alpha)\log(c/\varepsilon))$

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 ε. The following lower bounds can be proved (see Theorem 3.13, Theorem 3.14, and Theorem 3.15 in [2]):

	L-Lipschitz	β-smooth
Convex	$\Omega(\gamma a/(1+\sqrt{t}))$	$\Omega(\tilde{b}^2\beta/(t+1)^2)$
α-strongly convex	$\Omega(\gamma^2/(\alpha t))$	$\Omega(\alpha\tilde{b}^2 e^{-t\sqrt{\alpha/\beta}})$

where $a := \max_{x\in\mathcal{C}} \|x\|_2$ is the radius of the smallest Euclidean ball that contains $\mathcal{C}$ and $\tilde{b} := \max_{x,y\in\mathcal{C}} \|x-y\|_2$ 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 ℓ_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 $\varphi : \mathbb{R} \to \mathbb{R}_+$ be a given convex loss function. Let

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

where $\mathcal{W}_2$ is a convex set contained in a ball of radius $c_2^{\mathcal{W}}$, namely, $c_2^{\mathcal{W}} := \max_{w\in\mathcal{W}_2} \|w\|_2$.
The risk minimization problem reads (to be precise, we should refer to this problem as φ-risk minimization)

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & r(w) = \mathbf{E}\varphi(w^\top XY) \\ \text{subject to} \quad & w \in \mathcal{W}_2 \end{aligned}$$

Let $w_2^\star$ be a minimizer of this problem. The empirical risk minimization problem reads

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & R(w) = \frac{1}{n}\sum_{i=1}^n \varphi(w^\top X_i Y_i) \\ \text{subject to} \quad & w \in \mathcal{W}_2 \end{aligned} \tag{9.14}$$

Let $W_2^\star$ 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 $W \in \mathcal{W}$:

$$r(W) - r(w_2^\star) \le \underbrace{R(W) - R(W_2^\star)}_{\texttt{Optimization}_2} + \underbrace{\sup_{w\in\mathcal{W}_2}\{r(w) - R(w)\} + \sup_{w\in\mathcal{W}_2}\{R(w) - r(w)\}}_{\texttt{Statistics}_2}$$

Let us assume that φ is γ_φ-Lipschitz and that $\mathcal{X}$ is contained in a ball of radius $c_2^{\mathcal{X}}$, namely, $c_2^{\mathcal{X}} := \max_{x\in\mathcal{X}} \|x\|_2$.

By Proposition 2.11, Proposition 3.1, and Proposition 3.2, we have, respectively, (note that for this result we do *not* need φ to be convex)

$$\begin{aligned} \mathbf{E}\sup_{w\in\mathcal{W}_2}\{r(w) - R(w)\} &\le 2\,\mathbf{E}\,\mathtt{Rad}(\mathcal{L}\circ\{Z_1,\ldots,Z_n\}) \le 2\gamma_\varphi \mathbf{E}\,\mathtt{Rad}(\mathcal{A}_2\circ\{X_1,\ldots,X_n\}) \le 2\gamma_\varphi c_2^{\mathcal{W}}\mathbf{E}\,\frac{\max_i\|X_i\|_2}{\sqrt{n}} \\ &\le \frac{2c_2^{\mathcal{X}}c_2^{\mathcal{W}}\gamma_\varphi}{\sqrt{n}}, \end{aligned}$$

so that

$$\boxed{\mathbf{E}\,\texttt{Statistics}_2 \leq \frac{4c_2^{\mathcal{X}} c_2^{\mathcal{W}} \gamma_\varphi}{\sqrt{n}}}$$

Note that the empirical risk R is $(c_2^{\mathcal{X}}\gamma_\varphi)$-Lipschitz, as

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

where we used that $|Y_i| = 1$ and we used the Cauchy-Schwarz's inequality. Therefore, by applying projected subgradient descent with $\eta_s \equiv \eta = \frac{2c_2^{\mathcal{W}}}{c_2^{\mathcal{X}}\gamma_\varphi\sqrt{t}}$ to problem (9.14) we have, by Theorem 9.3,

$$\boxed{\texttt{Optimization}_2 = R(\overline{W}_t) - R(W_2^\star) \leq \frac{2c_2^{\mathcal{X}} c_2^{\mathcal{W}} \gamma_\varphi}{\sqrt{t}}}$$

where $\overline{W}_t := \frac{1}{t}\sum_{s=1}^{t} W_s$, where we used that $\|W_1 - W_2^\star\|_2 \leq \|W_1\|_2 + \|W_2^\star\|_2 \leq 2c_2^{\mathcal{W}}$.

We see that, as far as bounding the expected value of the estimation error $r(\overline{W}_t) - r(w^\star)$, 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

[1] Olivier Bousquet and Léon Bottou. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems*, pages 161–168, 2008.

[2] Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.