Fundamentals of Convex Optimization
Lecture 1, Date: 17.10.2014
Instructor: Nisheeth Vishnoi
Scribe: Jakub Tarnawski

Basics, Gradient Descent and Its Variants

1 Overview

Convex optimization is about minimizing a convex function over a convex set. For a convex set K, and a convex function f whose domain contains K, the goal is to solve the following problem:

$$\inf_{x \in K} f(x).$$

Convex optimization is a classical area with a long and rich history and diverse applications. Traditionally, a large fraction of algorithms in TCS have been obtained by formulating the underlying discrete problem (for example: SHORTEST PATH; MAXIMUM FLOW; MATCHING, GRAPH PARTITIONING) as a linear (or a semi-definite) program and, consequently, deploying standard methods such as the ellipsoid method or the interior point methods from convex programming to solve these programs. However, with the growth in the sizes of the real-world instances, there has been a pressing need to design faster and faster algorithms, ideally in almost the same time as it takes to read the data. In the last decade or so TCS researchers are coming remarkably close to what would have been considered a pipe-dream a few years ago. And tools and techniques from convex optimization are playing a bigger and bigger role in this goal. Since general methods of solving convex programs have runtimes which are far from linear, often this goal requires adapting known methods from convex optimization and, more often than not, coming up with novel problem-specific solutions. Thus, techniques from convex optimization have become an indispensable tool in the toolkit of any algorithm designer and in these three lectures, we cover a large ground in this regard. The methods we present include gradient descent and its many variants, multiplicative weight update methods and their application to approximately solving linear programs and semi-definite programs quickly, Newton's method and its application to path-following interior point methods. Through these methods, we will see a very intimate connection between convex optimization and *online convex*

optimization. With a good understanding of the material covered in these three lectures, a student should be well-equipped to understand many recent breakthrough works in TCS and, hopefully, push the state-of-the-art.

2 Basic Notions

2.1 Notation

We are concerned with functions $f : \mathbb{R}^n \mapsto \mathbb{R}$. By $x, y, \ldots$ we typically mean vectors in $\mathbb{R}^n$ and we use $x_1, x_2, \ldots$ to denote its coordinates. When the function is smooth enough, we can talk about its gradients and hessians. The gradient is denoted by

$$\nabla f(x) \stackrel{\text{def}}{=} \left(\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_n}(x) \right).$$

The hessian of f at a point x is a matrix denoted by $\nabla^2 f(x)$ whose (i,j)-th entry is $\frac{\partial^2 f}{\partial x_i \partial x_j}(x)$. We say that a symmetric matrix $M \succeq lI$, if all its eigenvalues are at least l. When $l \geq 0$, the matrix is said to be positive semi-definite (psd). Similarly, we denote by $M \preceq LI$ the fact that all eigenvalues of M are at-most L. $\langle x, y \rangle$ denotes the inner product between x and y. $\|x\|$ denotes the ℓ_2 norm unless stated otherwise. An important but easy to prove result is the Cauchy-Schwarz inequality which states that

$$\langle x, y \rangle \leq \|x\| \cdot \|y\|.$$

An important tool from calculus is Taylor expansion of a function f at y around a point x

$$f(y) = f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2}(y-x)^\top \nabla^2 f(x)(y-x) + \cdots$$

when the function is infinitely differentiable. The k-th order Taylor series approximation is obtained by truncating the above expression at k. However, if the function has $k+1$ derivatives, then one can use the Mean Value Theorem to truncate the above expression at k by a suitable modification and know the exact error. For instance, when $k = 1$

$$f(y) - (f(x) + \langle \nabla f(x), y - x \rangle) = \frac{1}{2}(y-x)^\top \nabla^2 f(\zeta)(y-x)$$

for some ζ in the line segment joining x and y.

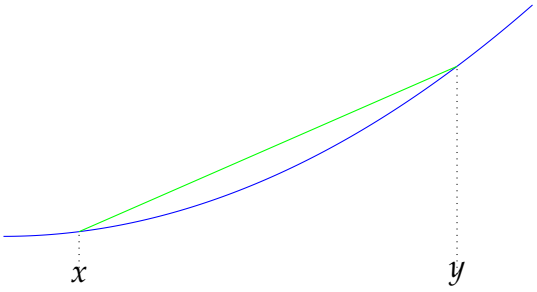

Figure 1: A convex function.

2.2 Convexity

We start by defining the basic notions of a convex set and a convex function.

Definition 1. *A set $K \subseteq \mathbb{R}^n$ is* convex *if, for every two points in K, the line segment connecting them is contained in K, i.e., for any $x, y \in K$ and $\lambda \in [0,1]$ we have*

$$\lambda x + (1-\lambda)y \in K. \tag{1}$$

Definition 2. *A function $f : K \to \mathbb{R}$ is* convex *if it satisfies*

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) \tag{2}$$

for any $x, y \in K$ and $\lambda \in [0,1]$. If inequality (2) is strict for all $x \neq y$ and $\lambda \in (0,1)$, then we say that f is strictly convex.

Geometrically, this means that for any $x, y \in K$, the segment connecting points $(x, f(x))$ and $(y, f(y))$ lies above the graph of f (in the space $\mathbb{R}^n \times \mathbb{R}$): see Fig. 1. While this is the most general definition of convexity, but not always the most useful. When f is smooth enough, we can give other equivalent definitions:

Proposition 3. *If $f : K \to \mathbb{R}$ is differentiable, then it is convex iff*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle \tag{3}$$

for any $x, y \in K$.

The geometric interpretation of this condition is that for any $x \in K$, the tangent space of f at x should lie below the graph of f: see Fig. 2. In other words, the right-hand side of Eq. (3), which is the first-order Taylor approximation of f at x, is an *under-estimation* for the value of f at every other point $y \in K$.

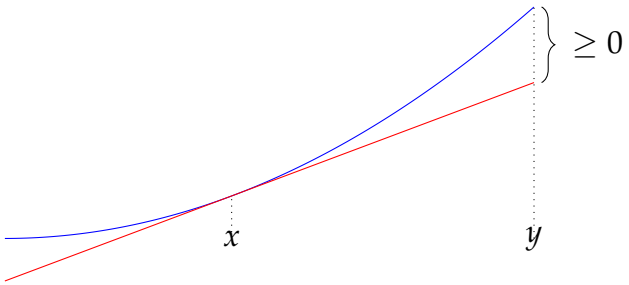

Figure 2: The first-order condition for convexity.

Proof. We prove the one-dimensional case first.

Suppose f is convex, and fix any $x, y \in K$. Then for every $\lambda \in (0, 1]$ we have

$$(1-\lambda)f(x) + \lambda f(y) \geq f(\lambda y + (1-\lambda)x) = f(x + \lambda(y-x)).$$

Subtracting $(1-\lambda)f(x)$ and dividing by λ yields

$$f(y) \geq f(x) + \frac{f(x+\lambda(y-x)) - f(x)}{\lambda}$$

and now it suffices to take the limit $\lambda \to 0$. (When $\lambda = 0$, there is nothing to prove.)

Conversely, suppose f satisfies Eq. (3) and fix $x, y \in K$ and $\lambda \in [0, 1]$. Let $z = \lambda x + (1-\lambda)y$. The first-order approximation of f at z underestimates both $f(x)$ and $f(y)$; the difference between the linear approximation of f at z using the values $f(x)$ and $f(y)$ and the actual value $f(z)$ is a weighted average of these underestimations and thus also nonnegative. Formally, we have

$$f(x) \geq f(z) + f'(z)(x-z), \tag{4}$$
$$f(y) \geq f(z) + f'(z)(y-z), \tag{5}$$

and multiplying (4) by λ and (5) by $1-\lambda$ yields

$$\lambda f(x) + (1-\lambda)f(y) \geq f(z).$$

To extend the proof to many dimensions, just note that after fixing points $x, y \in K$ it is enough to restrict our attention to the line segment connecting them. □

If the function has second derivatives, we can provide another characterization of convexity below. We leave the proof as an exercise.

Proposition 4. *Suppose K is convex and open. If $f : K \to \mathbb{R}$ is twice differentiable, then it is convex iff*

$$\nabla^2 f(x) \succeq 0$$

for any $x \in K$. If the inequality is strict for all $x \in K$, the function is called strongly convex.

In the rest of this lecture we will most often use the definition of Proposition 3.

3 Gradient Descent

We now introduce perhaps the simplest but extremely powerful *gradient descent* method to solve a convex program. Actually, it is not a single method, but a general framework with many possible realizations. We describe some concrete variants and analyze their performance. The performance guarantees that we are going to obtain will depend on assumptions that we make about f.

We describe the core ideas of the gradient descent methods in the *unconstrained setting*, i.e., $K = \mathbb{R}^n$. In Section 7.7 we discuss the constrained setting. Also, let us assume for simplicity that f has a unique minimum $x^\star$ (this follows if f is strictly convex) and that it is differentiable.

What does the gradient have to do with optimizing a convex function? We discuss this informally below. Suppose that we are at a point x, along with a value $f(x)$ that we want to decrease as much as possible. We want to accomplish this by moving to a point y at some pre-specified small distance from x. If we consider points y in a near neighbourhood of x, we should expect that the first-order Taylor approximation

$$f(y) \approx f(x) + \langle \nabla f(x), y - x \rangle$$

will be very tight: essentially an equality. Thus, if we set $\Delta x \stackrel{\text{def}}{=} y - x$, then we are tasked with minimizing the scalar product

$$\langle \nabla f(x), y - x \rangle = \langle \nabla f(x), \Delta x \rangle$$

where x is fixed, and $\|\Delta x\|$ is also given. The optimal solution of this problem has the form[1]

$$\Delta x = -\eta \cdot \nabla f(x)$$

and it yields

$$f(y) \approx f(x) - \eta \left\| \nabla f(x) \right\|^2 .$$

[1]To see this, consider the problem of minimizing $\langle v, w \rangle$ over all v with $\|v\| = 1$ for a fixed w. One cannot do better than $-\|w\|$, because $|\langle v, w \rangle| \leq \|v\| \cdot \|w\| = \|w\|$ by Cauchy-Schwartz.

Thus we see that the norm of the gradient controls the rate at which we can decrease f. More vividly, the gradient of f at x is the direction in which f grows the fastest, so if we are looking to minimize f, it makes the most sense to go in the opposite direction.

Any gradient descent method will work by constructing a sequence of points $x_1, x_2, \ldots, x_T$ with the objective of getting very close to the optimum $x^\star$ after a small number of iterations. Usually, it will try to ensure that $f(x_0) \geq \cdots \geq f(x_T)$ (although the first method that we will show does not guarantee this). We will not be able to find the exact optimum; we can only hope to get ε-close to it, for a given accuracy requirement ε.

The first nice property of convex functions that we can discover in this context, and a reason why gradient descent does not work for a general f, is the following: if we keep decreasing the value of f, we will not get "stuck" in a local minimum which is not a global minimum. This is guaranteed by the following fact, which again points out the role of the gradient in optimizing f.

Proposition 5. *For a differentiable convex function $f : \mathbb{R}^n \to \mathbb{R}$ and a point $x \in K$, the following conditions are equivalent:*

(a) x is a global minimum of f,

(b) x is a local minimum of f,

(c) $\nabla f(x) = 0$.

Proof. The direction (a)–(b) is trivial, and (b)–(c) holds for all f. For (c)–(a), just note that for any $y \in K$,

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle = f(x).$$

□

Remark 6. *The above proposition generalizes to the case $K \subseteq \mathbb{R}^n$ as follows: for a closed convex set K and a convex function $f : K \to \mathbb{R}$, a point $x \in K$ minimizes f iff*

$$\langle \nabla f(x), y - x \rangle \geq 0$$

for all $y \in K$. In other words, when there is no direction in K where the gradient decreases.

Let us proceed to the description of a general variant of gradient descent.

The algorithm will construct a sequence of points $x_1, x_2, \ldots, x_T$, for a value T which will depend on the assumptions we make about the structure of f and will be specified later. At the t-th iteration, knowing x_t, the algorithm takes x_{t+1} to be

$$x_{t+1} \stackrel{\text{def}}{=} x_t - \eta_t \nabla f(x_t)$$

where η_t is the *step size* (which will also depend on f and will be determined as per the setting.). For a given ε, our objective is to get ε-close to the optimum $f(x^\star)$ in as less iterations as possible. We want to understand what values of η_t will enable us to do achieve this goal.

4 Convex Functions, Oracles and their Properties

Till now we have ignored a very important detail: how is f given to the algorithm. In particular, to use the gradient algorithmically, one must be able to compute it. Here we will assume that the algorithm has access to an oracle which returns values of $f(x)$ and $\nabla f(x)$ for a query point $x \in K$. Such methods, which work in this oracle setting are referred to as *first-order methods*. Later, we will consider the setting where we would need access to a *second-order oracle* for f : given x, y output $(\nabla^2 f(x))^{-1} y$. Such methods are referred to as *second order methods*.

We also assume that we have a bound on the distance between the starting point and the optimum:

$$\|x_1 - x^\star\| \leq D.$$

If K were a bounded set, then $D = \operatorname{diam}(K)$ would be a valid choice; in the unconstrained setting, we must assume something about how far from the optimum we are starting.

Unfortunately, in general, it is still difficult to optimize f if we have no control at all over the magnitude of its gradient. We have to impose additional conditions on f.

We propose three such conditions. In each of these settings, we will obtain a guarantee on the convergence rate of our method, as well as a way to set the step lengths η_t.

4.1 Bounded gradient (Lipschitz f)

The simplest condition that we can propose is that the gradient be bounded from above: there should exist a $G > 0$ such that for all $x \in K$,

$$\|\nabla f(x)\| \leq G.$$

Why is this useful? Intuitively, a large gradient means that the function decreases very fast in the direction in which we are moving, which should be desirable. However, as we will see, this would make us unable to control the step size well enough.

Remark 7. *This condition is equivalent to saying that f is G-Lipschitz, that is,*

$$\|f(y) - f(x)\| \leq G \cdot \|y - x\|$$

for all $x, y \in K$. We will use this term in the sequel.

In this case we can get the following bound, which we prove in Section 5.1:

Theorem 8. *There is a gradient-descent method which, given an ε, a starting point x_1 satisfying $\|x_1 - x^\star\| \leq D$, and a function f which is G-Lipschitz, produces a sequence of points $x_1, \ldots, x_T$ such that*

$$f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - f(x^\star) \leq \varepsilon$$

where

$$T = \left(\frac{DG}{\varepsilon}\right)^2.$$

We mostly focus on the dependence of T on ε, and in this case it is

$$T = O\left(\varepsilon^{-2}\right), \quad \varepsilon = O\left(\frac{1}{\sqrt{T}}\right).$$

Note that we did not get any guarantee for the points $x_1, \ldots, x_T$ – just for their running averages.

Remark 9. *We are using the Euclidean norm here and assuming that $\|\nabla f\|_2 = O(1)$. But sometimes we might find ourselves in a setting where $\|\nabla f\|_\infty = O(1)$, and a naive bound would give us just $\|\nabla f\|_2 = O(\sqrt{n})$. We will see in later lectures how to deal with such situations (see also Section 7.6).*

4.2 Lipschitz gradient (smooth f)

We could also demand that the gradient be Lipschitz continuous.

Definition 10. *We say that a function f is L-smooth (with a constant $L > 0$) if for all $x, y \in K$,*

$$\|\nabla f(x) - \nabla f(y)\| \leq L \cdot \|x - y\|.$$

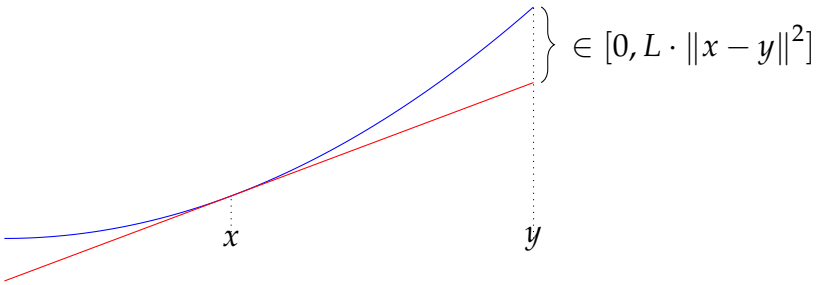

Figure 3: The Bregman divergence.

Remark 11. *L is the Lipschitz constant of ∇f. We can equivalently say that $\nabla^2 f(x) \preceq LI$ for all $x \in K$.*

To understand the usefulness of this condition, note that it implies the following crucial property:

Lemma 12. *If f is L-smooth, then for any $x, y \in K$ we have*

$$0 \leq f(y) - f(x) - \langle \nabla f(y), x - y \rangle \leq L \cdot \|x - y\|^2 .$$

This means that the distance of $f(y)$ from its first-order Taylor approximation at x is between 0 and $L \cdot \|x - y\|^2$, which is to be thought of as small; see Fig. 3. This distance is called the *Bregman divergence* with respect to the ℓ_2-norm.

Proof. We begin from the definition of convexity: $f(x) \geq f(y) + \langle \nabla f(y), x - y \rangle$, and use Cauchy-Schwarz along the way:

$$\begin{aligned} f(y) - f(x) &\leq \langle \nabla f(y), y - x \rangle \\ &= \langle \nabla f(x), y - x \rangle + \langle \nabla f(y) - \nabla f(x), y - x \rangle \\ &\leq \langle \nabla f(x), y - x \rangle + \|\nabla f(y) - \nabla f(x)\| \cdot \|y - x\| \\ &\leq \langle \nabla f(x), y - x \rangle + L \cdot \|x - y\|^2 . \end{aligned}$$

On the other hand we have (again from convexity)

$$f(y) - f(x) \geq \langle \nabla f(x), y - x \rangle .$$

□

This will enable us to obtain a better dependence of T on ε. The following bound is proved in Section 5.2:

Theorem 13. *There is a gradient-descent method which, given an ε, a starting point x_1 satisfying $\|x_1 - x^\star\| \leq D$, and an L-smooth function f, produces a sequence of points $x_1, \ldots, x_T$ such that*

$$f(x_T) - f(x^\star) \leq \varepsilon$$

and

$$T = O\left(\frac{LD^2}{\varepsilon}\right).$$

Again suppressing the constants, we see the improved dependence

$$T = O\left(\varepsilon^{-1}\right), \quad \varepsilon = O\left(\frac{1}{T}\right).$$

Moreover, this time we can show that we are really getting closer and closer to the optimum (not just on average).

While we will not cover it in the lectures, it is possible to attain a quadratic improvement using the important *accelerated gradient method* of Nesterov:

Theorem 14. *There is an algorithm which, given an ε, a starting point x_1 satisfying $\|x_1 - x^\star\| \leq D$, and an L-smooth function f, produces a sequence of points $x_1, \ldots, x_T$ such that*

$$f(x_T) - f(x^\star) \leq \varepsilon$$

and

$$T = O\left(\frac{\sqrt{L}D}{\sqrt{\varepsilon}}\right).$$

4.3 Strong convexity

Another natural restriction would be to *strongly convex* functions, i.e., those that have all eigenvalues of the hessian bounded below by a constant $\ell > 0$. In other words:

Definition 15 (strong convexity)**.** *We say that f is ℓ-*strongly convex *(with $\ell > 0$) if for all $x \in K$ we have*

$$\nabla^2 f(x) \succeq \ell I.$$

Remark 16. *This is equivalent to saying that*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\ell}{2} \|x - y\|^2 \tag{6}$$

for all $x, y \in K$. In other words, the corresponding Bregman divergence is lower bounded by $\frac{\ell}{2} \|x - y\|^2$.

If we also assume that f is G-Lipschitz, then we are able to get the following bound, whose proof appears in Section 5.3.

Theorem 17. *There is an algorithm which, given an ε, a starting point x_1, and a function f which is both G-Lipschitz and ℓ-strongly convex, produces a sequence of points $x_1, \ldots, x_T$ such that*

$$f(x_T) - f(x^\star) \leq \varepsilon$$

and

$$T = O\left(\frac{G^2}{\ell\varepsilon}\right).$$

5 Proofs of Convergence Rates

Now we prove the first two theorems stated in the previous section.

5.1 Lipschitz functions – proof of Theorem 8

Recall that we are guaranteed that f is G-Lipschitz. We will fix our step size η to be constant (independent of t).

We will be interested in understanding the quantity $f(x_t) - f(x^\star)$ – in particular, how fast it decreases with t. We begin by applying convexity and get

$$\begin{aligned} f(x_t) - f(x^\star) &\leq \langle \nabla f(x_t), x_t - x^\star \rangle \\ &= \frac{1}{\eta} \langle x_t - x_{t+1}, x_t - x^\star \rangle \\ &= \frac{1}{2\eta} \left(\|x_t - x_{t+1}\|^2 + \|x_t - x^\star\|^2 - \|x_{t+1} - x^\star\|^2 \right), \end{aligned} \tag{7}$$

where we used the well-known equality $2\langle a, b\rangle = \|a\|^2 + \|b\|^2 - \|a-b\|^2$. We can now take advantage of our assumption about f and write

$$\|x_t - x_{t+1}\|^2 = \eta^2 \|\nabla f(x)\|^2 \leq \eta^2 G^2. \tag{8}$$

From (7) and (8) we get

$$f(x_t) - f(x^\star) \leq \frac{1}{2\eta} \left(\eta^2 G^2 + \|x_t - x^\star\|^2 - \|x_{t+1} - x^\star\|^2 \right)$$

for every $t = 1, \ldots, T$. Notice that if we add all these inequalities, we will get a telescoping sum:

$$\sum_{t=1}^{T} (f(x_t) - f(x^\star)) \leq \frac{1}{2\eta} \left(T\eta^2 G^2 + \|x_1 - x^\star\|^2 - \|x_{T+1} - x^\star\|^2 \right).$$

We may bound the last term by simply zero. For the middle one, recall that we have introduced D which bounds the distance from the starting point to the optimum. Thus, using $\|x_1 - x^\star\|^2 \leq D^2$, we obtain

$$\sum_{t=1}^{T} (f(x_t) - f(x^\star)) \leq \frac{1}{2\eta} \left(T\eta^2 G^2 + D^2 \right) = \frac{1}{2} \left(T\eta G^2 + \frac{D^2}{\eta} \right).$$

The minimizing choice of η here is the one which satisfies $T\eta G^2 = \frac{D^2}{\eta}$, so that

$$\eta = \frac{D}{G\sqrt{T}}.$$

We have bounded the quantity $\frac{1}{T}\sum_{t=1}^{T} f(x_t) - f(x^\star)$. But from convexity of f we have[2] that $f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) \leq \frac{1}{T}\sum_{t=1}^{T} f(x_t)$. Thus we get

$$f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - f(x^\star) \leq \frac{1}{2T} \left(T\eta G^2 + \frac{D^2}{\eta} \right) = \frac{DG}{\sqrt{T}}.$$

In order to get $\frac{DG}{\sqrt{T}} \leq \varepsilon$ we need to set $T \geq \left(\frac{DG}{\varepsilon}\right)^2$.

5.2 Smooth functions – proof of Theorem 13

Recall that we are assuming that f is L-smooth, i.e., that ∇f is L-Lipschitz. This implies Lemma 12 (a bound on the Bregman divergence).

Once again, we will keep the step length η constant and take

$$x_{t+1} - x_t = -\eta \cdot \nabla f(x_t).$$

Let us see how the value of f changes as we iterate. Use Lemma 12 to get

$$\begin{aligned} f(x_{t+1}) - f(x_t) &\leq \langle \nabla f(x_t), x_{t+1} - x_t \rangle + L \cdot \|x_{t+1} - x_t\|^2 \\ &= -\eta \|\nabla f(x_t)\|^2 + L\eta^2 \|\nabla f(x_t)\|^2. \end{aligned}$$

[2] Extend Eq. (2) to T points.

The minimizing value of η is $\frac{1}{2L}$. By substituting it, we get

$$f(x_{t+1}) - f(x_t) \leq -\frac{1}{4L} \|\nabla f(x_t)\|^2 . \tag{9}$$

Intuitively, this means that, at every step, either the gradient is large and we are making good progress, or it is small, which means that we are already close to the optimum. Indeed, if we are at a distance θ from the optimum value:

$$f(x_t) - f(x^\star) \geq \theta,$$

then by convexity and Cauchy-Schwarz,

$$\theta \leq f(x_t) - f(x^\star) \leq \langle \nabla f(x_t), x_t - x^\star \rangle \leq \|\nabla f(x_t)\| \cdot \|x_t - x^\star\|$$

and if we bound $\|x_t - x^\star\|$ by D as in the previous analysis, then we get

$$\|\nabla f(x_t)\| \geq \frac{\theta}{D}$$

and, by (9),

$$f(x_{t+1}) - f(x_t) \leq -\frac{\theta^2}{4LD^2}.$$

Until our distance from the optimum goes down below $\frac{\theta}{2}$, we will make a progress of $\Omega\left(\frac{\theta^2}{LD^2}\right)$ at every step, so we will take at most $O\left(\frac{LD^2}{\theta}\right)$ steps before this happens. Then this process of halving is repeated, and so on, until we reach the required distance ε. Bringing the distance down from $\frac{\theta}{2^i}$ to $\frac{\theta}{2^{i+1}}$ requires $O\left(\frac{LD^2 2^i}{\theta}\right)$ steps, so we get

$$\sum_{i=0}^{\log \frac{\theta}{\varepsilon}} O\left(\frac{LD^2 2^i}{\theta}\right) = O\left(\frac{LD^2 2^{\log \frac{\theta}{\varepsilon}}}{\theta}\right) = O\left(\frac{LD^2}{\varepsilon}\right)$$

steps before we are ε-close to the optimum.

5.3 Strongly convex functions – proof of Theorem 17

Recall that f is ℓ-strongly convex and G-Lipschitz. As a start, we will try to mimic the proof from Section 5.1. However, we will not fix the step size η_t to be constant, and we use the first-order condition for convexity in the strong form (Eq. (6)). We easily get

$$f(x_t) - f(x^\star) \leq \frac{1}{2\eta_t} \left(\eta_t^2 G^2 + \|x_t - x^\star\|^2 - \|x_{t+1} - x^\star\|^2\right) - \frac{\ell}{2} \|x_t - x^\star\|^2 \tag{10}$$

for every $t = 1, \ldots, T$. Now, to obtain a better bound than previously, we must make good use of the new term (the last one). Intuitively, if $\|x_t - x^\star\|$ is large, then it is very helpful, but as t grows and $\|x_t - x^\star\|$ decreases, its importance is diminished. We will try to prevent this from happening by multiplying our inequality by t. And we will still aim to obtain a telescoping sum. So let us sum all the t-multiples of Eq. (10):

$$\sum_{t=1}^{T} t\left(f(x_t) - f(x^\star)\right) \leq \frac{G^2}{2}\sum_{t=1}^{T} t\eta_t + \sum_{t=2}^{T} \|x_t - x^\star\|^2 \cdot \left(\frac{t}{2\eta_t} - \frac{\ell t}{2} - \frac{t-1}{2\eta_{t-1}}\right) + \|x_1 - x^\star\|^2 \cdot \left(\frac{1}{2\eta_1} - \frac{\ell}{2}\right) - \|x_{T+1} - x^\star\|^2 \cdot \frac{T}{2\eta_T}.$$

As before, we bound the last term by just zero. Now, to make the sum telescoping, we would like to get $\frac{t}{2\eta_t} - \frac{\ell t}{2} - \frac{t-1}{2\eta_{t-1}} = 0$ for every $t = 2, \ldots, T$. As for the term $\frac{1}{2\eta_1} - \frac{\ell}{2}$, we would also prefer to remove it, so as not to have any dependence on $\|x_1 - x^\star\|$. Solving this system of equations (beginning with $\frac{1}{2\eta_1} - \frac{\ell}{2} = 0$) yields the following setting of η_ts:

$$\eta_t = \frac{2}{\ell(t+1)}.$$

We are thus left with:

$$\sum_{t=1}^{T} t\left(f(x_t) - f(x^\star)\right) \leq \frac{G^2}{2}\sum_{t=1}^{T} t\eta_t = \sum_{t=1}^{T} \frac{G^2 t}{\ell(t+1)} \leq \sum_{t=1}^{T} \frac{G^2}{\ell} = \frac{G^2 T}{\ell}.$$

The rest is straightforward: we normalize by $(1 + \cdots + T) = \frac{T(T+1)}{2}$ and use convexity of f to get

$$f\left(\frac{2}{T(T+1)}\sum_{t=1}^{T} t \cdot x_t\right) - f(x^\star) \leq \frac{2G^2}{\ell(T+1)},$$

and bringing this down to ε requires setting

$$T = O\left(\frac{G^2}{\ell\varepsilon}\right).$$

6 Strong Convexity - Solving PSD linear systems

Suppose we have a linear system with a positive definite constraint matrix, i.e. we want to find a vector x satisfying $Ax = b$, where $A \succ 0$. Let us formulate it as a convex problem. Define

$$f(x) = \frac{1}{2}x^\top Ax - x^\top b.$$

Then f is strongly convex:

$$\nabla^2 f(x) = A \succ 0$$

and one can verify that

$$\nabla f(x) = Ax - b$$

so that $x^\star$ minimizes f iff it is a solution to the linear system $Ax = b$. So solving the system amounts to minimizing f, and we will do this using gradient descent. We will construct a sequence of points $x_1, \ldots, x_T$ using the formula

$$x_{t+1} = x_t - \eta_t \cdot \nabla f(x_t)$$

as previously. However, this time we will not use the same step length η_t for every iteration, but instead (**exercise**) analytically find a closed-form expression for the best η_t, i.e.

$$\eta_t = \text{argmin}_\eta \, f\left(x_t - \eta \cdot \nabla f(x_t)\right).$$

So at each step we will decrease f as much as possible by going in the direction opposite to the gradient. This is called the *steepest descent method*.

If we do this, then one can show (**exercise**) that the convergence rate of this method (to a predefined accuracy ε: $f(x_T) - f(x^\star) \leq \varepsilon$) is given by

$$T = O\left(\kappa(A) \cdot \log \frac{1}{\varepsilon}\right),$$

where $\kappa(A) = \frac{L}{\ell}$, the ratio between the largest and the smallest eigenvalue of A, is the *condition number* of the matrix A.[3]

[3]For more information on this and the related *Conjugate Gradient method*, see Chapter 15 of the monograph [Vis13].

7 Discussion

7.1 Gradient vs. Sub-gradient

Throughout this lecture we assumed the function f to be differentiable or twice differentiable as and when we needed. However, in many applications (practical and theoretical), f is not differentiable everywhere. Could any of the methods presented in this section work in this non-differentiable setting? Start by noting that if f is convex then it is continuous. Hence, we need to only consider the issue of non-differentiable f. Non-differentiability poses the problem that there may be some points at which the gradient is not unique. In general, at a non-differentiable point x, the set of gradients forms a set which which we denote by $\partial f(x)$ and an element of it is referred to as a *sub-gradient*. First we note that the convexity of f implies that for any x, y

$$f(y) \geq f(x) + \langle g, y - x \rangle$$

for all $g \in \partial f(x)$. Thus, we could modify the gradient descent in a natural manner:

$$x_{t+1} = x_t - \eta_t g$$

for any $g \in \partial f(x_t)$ given to us by the first order oracle for f. Further, it can be easily checked that the guarantee claimed in Theorem 8 and its corollaries carry over with the following parameter:

$$G \stackrel{\text{def}}{=} \sup_x \sup_{g \in \partial f(x)} \|g\|.$$

7.2 Dependence on ε

We have seen in Theorems 8, 13, 14 and 17 that our methods converge to the ε-approximate solution in a number of iterations which is $O\left(\text{poly}\left(\frac{1}{\varepsilon}\right)\right)$, with the exponent equal to 2, 1 or $\frac{1}{2}$. However, a polynomial dependence on ε does not look very good if we hope to use these methods in computer science. For such applications, the right answer should be $O\left(\text{poly}\left(\log \frac{1}{\varepsilon}\right)\right)$. And, if the function is both strongly convex and smooth, it is possible to obtain a convergence rate similar to the one in Section 6: see e.g. Section 3.4.2 of [Bub14].

7.3 Dependence on n

Note that n does not appear anywhere in the oracle complexity[4] of the presented algorithms. This is an important feature of first-order methods. (Of course, the

[4]The number of queries for $f(x)$, $\nabla f(x)$ etc.

time complexity of a single iteration will still depend on n.)

7.4 Coordinate Descent

Instead of moving in the exact direction in which f decreases the fastest, we may limit ourselves only to moving along the standard basis vectors, i.e. to changing only one coordinate of x at a time.

For example, rather than computing the whole gradient

$$\nabla f(x_t) = \frac{\partial f}{\partial x_1}(x_t) \cdot e_1 + \cdots + \frac{\partial f}{\partial x_n}(x_t) \cdot e_n$$

(where $e_1, \ldots, e_n$ are standard basis vectors), we can only pick one *random* coordinate i and update

$$x_{t+1} = x_t - \eta_t \cdot \frac{\partial f}{\partial x_i}(x_t) \cdot e_i.$$

We can analyze this *random coordinate descent* by examining the expected decrease in the function value. One can prove the following theorem. We omit its simple proof.

Theorem 18. *There is an algorithm which, given an ε, a starting point x_1 satisfying $\|x_1 - x^\star\| \leq D$, and a function f which is G-Lipschitz, produces a sequence of (random) points $x_1, \ldots, x_T$ such that*

$$\mathbb{E}\left[f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right)\right] - f(x^\star) \leq \varepsilon$$

and

$$T = O\left(\left(\frac{DG}{\varepsilon}\right)^2 \cdot n\right).$$

As we would expect, changing only one coordinate at a time comes at a cost of an n-fold increase in the number of iterations (compared to our first method of Theorem 8).

7.5 Online Convex Optimization

Consider the following game against an adversary. Given are: a convex set of strategies K, and a family of convex functions $\mathcal{F}$ (for example, all convex functions which are G-Lipschitz). The game proceeds in rounds. In the t-th round, the player picks a point $x_t \in K$, and then the adversary picks a function $f_t \in \mathcal{F}$. The value $f_t(x_t)$ is considered to be the player's loss at time t. The objective of the game is to minimize the *regret* of the player up to a time T:

Definition 19. *For a sequence of points $x_1, \ldots, x_T$ and functions $f_1, \ldots, f_T$, the* regret *up to time T is defined as*

$$\text{Regret}_T = \sum_{t=1}^{T} f_t(x_t) - \min_{x \in K} \sum_{t=1}^{T} f_t(x).$$

I.e., we compare the player's loss to the minimum loss which could be achieved if one knew all the functions f_t beforehand, but were only allowed to pick a single argument x for all of them.[5] The player is trying to minimize the regret, and the adversary is trying to maximize it. This model has many practical applications, such as stock pricing and portfolio selection, see [Haz14].

One can describe a strategy for the player which uses gradient descent:

$$x_{t+1} = x_t - \eta_t \cdot \nabla f_t(x_t)$$

for a suitable choice of η_t. Even though the functions f_t change from iteration to iteration, one can still obtain a similar guarantee to the one from our first analysis. Namely, we are able to make the *average regret* go to 0 at a rate inversely proportional to the square root of T:

$$\frac{\text{Regret}_T}{T} = O\left(\frac{1}{\sqrt{T}}\right).$$

Indeed, note that in the proof of Section 5.1 we never used the fact that the function f remains the same between iterations. We actually proved the following:

Corollary 20. *There is an algorithm which, given a starting point x_1, parameters ε and D, and a family $f_1, \ldots, f_T$ of convex functions which are G-Lipschitz, produces a sequence of points $x_1, \ldots, x_T$ such that*

$$\frac{1}{T} \sum_{t=1}^{T} (f_t(x_t) - f_t(x^\star)) \leq \frac{DG}{\sqrt{T}}$$

for any $x^\star$ satisfying $\|x_1 - x^\star\| \leq D$, and T being

$$T = \left(\frac{DG}{\varepsilon}\right)^2.$$

Moreover, it produces each point x_{t+1} knowing only the functions $f_1, \ldots, f_t$.

[5] For some choices of $x_1, \ldots, x_t, f_1, \ldots, f_t$, regret can be negative.

However, we will see in the next lecture that multiplicative weight update methods are often able to perform much better in the online learning setting.

There are also other optimization scenarios where access to information is even more limited (and such models are thus closer to real life applications). For example, we might be only given oracle access to the function (i.e., we do not see any "formula" for f). In particular, we cannot compute the gradient directly. (This is called the *bandit setting*.) Even then, methods based on gradient descent are very competitive.

7.6 Non-Euclidean Norms

Our algorithm for the L-smooth case can be easily adapted to work with functions which are smooth with respect to any pair of dual norms.[6] Now, moving in the direction of the gradient makes progress which we can measure using the Euclidean norm, because then

$$\langle \nabla f(x_t), x_{t+1} - x_t \rangle = \langle \nabla f(x_t), -\eta \nabla f(x_t) \rangle = -\eta \left\| \nabla f(x_t) \right\|_2^2 .$$

If we want to utilize a different pair of dual norms, we must adapt our direction of descent to the geometry that they induce. To this end, we move instead in the direction

$$x_{t+1} = x_t - \eta \left(\nabla f(x_t) \right)^{\#} ,$$

where for any $x \in \mathbb{R}^n$ we define $x^{\#}$ to be the minimizer of

$$\frac{1}{2} \left\| x^{\#} \right\|^2 - \left\langle x^{\#}, x \right\rangle . \tag{11}$$

(Note that for $\|\cdot\| = \|\cdot\|_2$ we have $x^{\#} = x$.) Lemma 12 still holds in the new setting, and one can show that a step size of $\eta = \frac{1}{L}$ allows us to get

$$f(x_{t+1}) - f(x_t) \leq -\frac{1}{2L} \left\| \nabla f(x_t) \right\|_\star^2 .$$

[6] Let $\|\cdot\|$ be any norm. We define the dual norm $\|\cdot\|_\star$ as follows:

$$\|y\|_\star = \sup_{x \in \mathbb{R}^n : \|x\| = 1} |\langle x, y \rangle| .$$

Then we say that a function f is L-smooth with respect to $\|\cdot\|$ if for any $x, y \in K$,

$$\left\| \nabla f(x) - \nabla f(y) \right\|_\star \leq L \cdot \|x - y\| .$$

Then the analysis proceeds as in Section 5.2, giving us the following generalization of Theorem 13:

Theorem 21. *There is an algorithm which, given an ε, a starting point x_1 satisfying $\|x_1 - x^\star\| \leq D$, and a function f which is L-smooth with respect to the norm $\|\cdot\|$, produces a sequence of points $x_1, \ldots, x_T$ such that*

$$f(x_T) - f(x^\star) \leq \varepsilon$$

and

$$T = O\left(\frac{LD^2}{\varepsilon}\right).$$

To produce each point x_t, it makes one call to a routine which computes $x^\#$ for a given x, i.e. minimizes an expression of the form (11)*.*

7.7 Constrained setting – projection

If K is not the whole space $\mathbb{R}^n$, then our choice of the next iterate x_{t+1} in the gradient descent method might fall outside of the convex body K. In this case we need to project it back onto K: to find the point in K with the minimum distance to x, and take it to be our new iterate instead:

$$x_{t+1} = \text{proj}_K\left(x_t - \eta_t \cdot \nabla f(x_t)\right).$$

The convergence rates remain the same (for example, the first proof just carries over to this new setting, since the value of f at the new point will be at least as good as the bound that we had derived for its value at the original point). However, depending on K, the projection may or may not be difficult (or computationally expensive) to perform.

More precisely, as long as the algorithm has access to an oracle which, given a query point x, returns the projection $\text{proj}_K(x)$ of x onto K, then for the G-Lipschitz case we have the following analogue of Theorem 8:

Theorem 22. *There is an algorithm which, given an ε, a function $f : K \to \mathbb{R}$ which is G-Lipschitz and has a minimum $x^\star$, and a starting point x_1 satisfying $\|x_1 - x^\star\| \leq D$, produces a sequence of points $x_1, \ldots, x_T \in K$ such that*

$$f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right) - f(x^\star) \leq \varepsilon$$

and

$$T = \left(\frac{DG}{\varepsilon}\right)^2.$$

To compute each point x_t, it uses one gradient query and one projection query.

And for the L-smooth case we get the following analogue of Theorem 13.

Theorem 23. *There is an algorithm which, given an ε, a function $f : K \to \mathbb{R}$ which is L-smooth and has a minimum $x^\star$, and a starting point x_1 satisfying $\|x_1 - x^\star\| \leq D$, produces a sequence of points $x_1, \ldots, x_T \in K$ such that*

$$f(x_T) - f(x^\star) \leq \varepsilon$$

and

$$T = O\left(\frac{LD^2 + f(x_1) - f(x^\star)}{\varepsilon}\right).$$

To compute each point x_t, it uses one gradient query and one projection query.

7.8 Constrained setting – the Frank-Wolfe algorithm

In cases where computing projections is prohibitively difficult (for example, harder than the original problem), one can use another variant of gradient descent: the *Frank-Wolfe algorithm*[7]. At the t-th step, it first considers the first-order Taylor approximation of f around x_t, and tries to minimize this function over K, obtaining a minimizer y_t. Then it takes the new point x_{t+1} to be a weighted average of x_t and y_t, which is guaranteed to also be in K. More precisely, it first minimizes the function

$$f(x_t) + \langle \nabla f(x), y_t - x_t \rangle$$

over $y_t \in K$. But this function is linear in y_t, and the task amounts to finding the minimizer of $\langle y_t, \nabla f(x_t) \rangle$ over $y_t \in K$. The new point is selected as

$$x_{t+1} = (1 - \gamma_t)x_t + \gamma_t y_t$$

with $\gamma_t \in [0, 1]$ being a parameter[8]. Because this is a convex combination, we get that $x_{t+1} \in K$ (as long as the starting point x_1 was in K).

This way, instead of having to project onto K, we need to be able to optimize linear functions over K, which may very well be easier. For example, if K is a polytope, then we are left with a linear programming subproblem.

Using this method one can obtain the same bound on the number of iterations as in Theorem 13: (see also Theorem 3.4 in [Bub14]):

[7] Also called the conditional gradient descent method.

[8] A good choice is $\gamma_t = \frac{2}{t+1}$.

Theorem 24. *There is an algorithm which, given parameters ε and D, where $D = \operatorname{diam}(K)$, a function $f : K \to \mathbb{R}$ which is L-smooth (with respect to any norm $\|\cdot\|$) and has a minimum $x^\star$, and any starting point $x_1 \in K$, produces a sequence of points $x_1, \ldots, x_T \in K$ such that*

$$f(x_T) - f(x^\star) \leq \varepsilon$$

and

$$T = O\left(\frac{LD^2}{\varepsilon}\right).$$

To compute each point x_t, it uses one gradient query and one call to a routine which optimizes a linear function over K.

This algorithm has another important feature: if K is a polytope and x_1 is a vertex of K, then any iterate x_t can be written as a convex combination of t vertices of K. In constrast, Caratheodory's theorem guarantees that any $x \in K$ can be written as a convex combination of at most $n + 1$ vertices of K. Because T is independent of n (and will often be much smaller), we can say that x_T is sparse in the polytope-vertex representation.

References

[Bub14] Sebastien Bubeck. Theory of convex optimization for machine learning. Internet draft, 2014. URL: `http://arxiv.org/abs/1405.4980`.

[Haz14] Elad Hazan. Introduction to online convex optimization. Internet draft, 2014. URL: `http://ocobook.cs.princeton.edu/`.

[Vis13] Nisheeth K. Vishnoi. Lx = b. *Foundations and Trends in Theoretical Computer Science*, 8(1-2):1–141, 2013. URL: `http://research.microsoft.com/en-us/um/people/nvishno/site/Lxb-Web.pdf`.