250049-1 VO Stochastic Algorithms

Yurii Malitskyi

Contents

Notation

$[d]$	$\{1, \ldots, d\}$
x^k	k-th iterate of an algorithm
x_i	the i-th coordinate of the vector x.
$\mathrm{dom}(f)$	$\{x : f(x) < +\infty\}$
$\nabla_i f(x)$	the i-th coordinate of $\nabla f(x)$
$[x, y]$	an interval if $x, y \in \mathbb{R}$
$[x, y]$	a segment $\{\lambda x + (1-\lambda)y : \lambda \in [0, 1]\}$ if x, y are vectors.
$B(x, r)$	a closed ball with the center x and radius r: $\{x : \|x\|_2 \leqslant r\}$
$U(x, r)$	an open ball with the center x and radius r: $\{x : \|x\|_2 < r\}$
e_i	the i-th vector from the standard orthogonal basis

Lecture 1: Smooth minimization

What is this course about? We will cover stochastic algorithms from optimization, machine learning, and theoretical computer science (as time permits). We will learn how to analyze such algorithms, how to apply such algorithms in practice, and why randomness can be so helpful. Our focus will change from time to time, depending on the context. There will be a few basic algorithms that you will need to know by heart how to analyze, and there will be many more for which you will need a big-picture view of why they are important.

I will warn you in advance: there will be a lot of different algorithms. Analyzing them is not difficult in most cases. However, the sheer number of techniques may seem daunting.

R These lecture notes may contain typos. If you notice any (especially in math), please send them to me by email.

1.1 Optimization algorithms

There will be many algorithms in this course, and for each of them we should remember in which setting the algorithm works. This usually depends on assumptions.

For now, our central problem is the unconstrained minimization problem $\min_{x\in\mathbb{R}^d} f(x)$. Two main assumptions about f evolve along two orthogonal directions: convexity and smoothness. Thus, for each algorithm we study, we should keep this table in mind and understand in which case our algorithm is (theoretically) applicable.

Smoothness/Convexity	Nonconvex	Convex	Strongly convex
L-smooth	·	·	·
Nonsmooth	·	·	·

Recall the following definition.

Definition 1.1 A differentiable function $f:\mathbb{R}^d\to\mathbb{R}$ is called *L-smooth* if its gradient ∇f is L-smooth, that is

$$\|\nabla f(x)-\nabla f(y)\|_2 \leqslant L\|y-x\|_2, \quad \forall x,y.$$

Guarantees. Usually, when we discuss an algorithm, we are interested in some guarantees: how many iterations the algorithm must run to achieve the desired accuracy. This accuracy can be measured in several ways:

$$\|x^k-x^*\|\leqslant\varepsilon, \qquad f(x^k)-f_*\leqslant\varepsilon, \qquad \|\nabla f(x^k)\|\leqslant\varepsilon. \tag{1.1}$$

Not all of them are always applicable. For example, there is little hope of using $\|x^k - x^*\| \leqslant \varepsilon$ for the nonconvex case (in general), since x^* is not unique and we cannot guarantee that (x^k) converges to a particular x^*. The same can be said for $f(x^k) - f_* \leqslant \varepsilon$ in the nonconvex case. In the convex case, there may be multiple solutions x^*, so again having $\|x^k - x^*\| \leqslant \varepsilon$ is generally hopeless. On the other hand, in the convex case f_* is always the same, so it makes sense to consider $f(x^k) - f_* \leqslant \varepsilon$.

Also, it is good to keep in mind that in most cases the "strength" of the optimality measure in (1.1) decreases from left to right. That is, if possible, $\|x^k - x^*\| \leqslant \varepsilon$ is preferable to $\|\nabla f(x^k)\| \leqslant \varepsilon$.

We may rightly think that the nonconvex case is the most difficult. This is indeed true in terms of problem solving: in general, the best we can hope for is to find a local minimum. In terms of algorithm analysis, however, it is often the opposite. The nonconvex world is too general, it doesn't have the rich tools of the convex world. This limits the number of tricks one can try when analyzing algorithms in the nonconvex case.

It is a good habit to have a simple convex function at hand to test all the studied theory on it. The best example of such a function is a convex quadratic function.

Exercise 1.1 Consider $f(x) = \frac{1}{2}\langle Ax, x\rangle - \langle b, x\rangle$ with a symmetric positive semi-definite matrix A. Compute ∇f and $\nabla^2 f$. Prove that f is convex and L-smooth. What will change if $A \succ 0$? ∎

1.2 Gradient Descent

The most basic (and probably the most important) optimization method for continuous optimization is gradient descent:

$$x^{k+1} = x^k - \alpha \nabla f(x^k),$$

where $\alpha > 0$ is a stepsize (also known as a "learning rate" in the machine-learning literature). As I assume some familiarity with this method, I will only provide dry facts.

Here are some classical rates of GD with the stepsize $\alpha = \frac{1}{L}$:

	Convexity		
Smoothness	Nonconvex	Convex	μ-strongly convex
L-smooth	$O\left(\frac{1}{\varepsilon^2}\right)$ for $\|\nabla f(x^k)\| \leqslant \varepsilon$	$O\left(\frac{L}{\varepsilon}\right)$ for $f(x^k) - f_* \leqslant \varepsilon$	$O\left(\frac{L}{\mu}\log\frac{1}{\varepsilon}\right)$ for $f(x^k) - f_* \leqslant \varepsilon$

Note that this is not a complete picture: one could derive the convergence rate for $\|\nabla f(x^k)\| \leqslant \varepsilon$ in the convex and μ-strongly convex case and additionally $\|x^k - x^*\| \leqslant \varepsilon$ in the strongly convex case.

Lemma 1.1 Descent lemma. If f is L-smooth, then for all x, y it holds

$$f(y) \leqslant f(x) + \langle \nabla f(x), y - x\rangle + \frac{L}{2}\|y - x\|^2. \tag{1.2}$$

Proof. The proof is straightforward and follows from the Taylor expansion. ∎

Lemma 1.2 One step of GD. Suppose that f is L-smooth. Then one step of GD with the stepsize α satisfies

$$f(x^{k+1}) \leqslant f(x^k) - \frac{\alpha(2-\alpha L)}{2}\|\nabla f(x^k)\|^2. \tag{1.3}$$

Proof. By inequality (1.2) in descent lemma,

$$\begin{aligned} f(x^{k+1}) &\leqslant f(x^k) + \langle \nabla f(x^k), x^{k+1} - x^k\rangle + \frac{L}{2}\|x^{k+1} - x^k\|^2 \\ &= f(x^k) - \alpha\|\nabla f(x^k)\|^2 + \frac{\alpha^2 L}{2}\|\nabla f(x^k)\|^2 \\ &= f(x^k) - \frac{\alpha(2-\alpha L)}{2}\|\nabla f(x^k)\|^2. \end{aligned}$$

■

From inequality (1.3), we immediately see that in order to have a descent, that is $f(x^{k+1}) \leqslant f(x^k)$, the stepsize must satisfy $\alpha \in \left(0, \frac{2}{L}\right)$. Moreover, from our arguments above it follows that the optimal choice for α is $\alpha = \frac{1}{L}$[1]. With this choice, we obtain

$$f(x^{k+1}) \leqslant f(x^k) - \frac{1}{2L}\|\nabla f(x^k)\|^2. \tag{1.4}$$

Theorem 1.1 Gradient descent for convex functions. Suppose that f is convex and L-smooth. Then with $\alpha = \frac{1}{L}$, we have that

$$f(x^k) - f_* \leqslant \frac{L\|x^0 - x^*\|^2}{2k}. \tag{1.5}$$

Thus, convergence rate of GD for $f(x^k) - f_*$ is of order $O\left(\frac{1}{k}\right)$. This is known as a *sublinear* rate.

Given accuracy ε, how many iterations of GD do we need to ensure that we reached it? As before, it means that we have to find the smallest integer k such that

$$\frac{L\|x^0 - x^*\|^2}{2k} \leqslant \varepsilon \quad \Longleftrightarrow \quad k \geqslant \frac{L\|x^0 - x^*\|^2}{2\varepsilon}.$$

Hence, the convergence rate of GD in the convex case is $O\left(\frac{1}{\varepsilon}\right)$.

In general, we cannot compare rates of algorithms that are expressed in different quantitites. However, in the convex case, one can additionally prove that

$$\|\nabla f(x^k)\| = O\left(\frac{1}{k}\right),$$

which is strictly better than the one in the nonconvex case. Thus, convexity not only makes local minima global, but also guarantees faster convergence of algorithms.

Exercise 1.2 Use Lemma 1.2 to derive $O\left(\frac{1}{\varepsilon^2}\right)$ rate for $\min_{0\leqslant i\leqslant k}\|\nabla f(x^i)\| \leqslant \varepsilon$. For this we need to assume that f is lower bounded: $f(x) \geqslant f_{\text{low}}$ for all x. ■

[1] This doesn't mean that $\alpha = \frac{1}{L}$ is always the best stepsize, it follows only from the analysis we provided. But our analysis may be too loose.

Accelerated gradient method. In 1983 Y. Nesterov proposed a modification[2] of GD:

$$\begin{aligned} y^k &= x^k + \beta_k(x^k - x^{k-1}) \\ x^{k+1} &= y^k - \alpha \nabla f(y^k), \end{aligned} \tag{1.6}$$

where $k \geqslant 0$ and $x^{-1} = x^0$, with unexpected property. This scheme achieves the rate $O\left(\frac{1}{\sqrt{\varepsilon}}\right)$ for $f(x^k) - f_* \leqslant \varepsilon$, which is much better than GD has.

Theorem 1.2 Nesterov's accelerated method. Suppose that $f : \mathbb{R}^d \to \mathbb{R}$ is convex and L-smooth. Then iterates of (1.6) with $\alpha = \frac{1}{L}$ and $\beta_k = \frac{k}{k+3}$ satisfy

$$f(x^k) - f_* \leqslant \frac{2L\|x^0 - x^*\|^2}{k^2}.$$

As we discussed in the lecture, $O\left(\frac{1}{k^2}\right)$ is the best possible rate for the class of convex and L-smooth functions for the first-order algorithms.

Comments

To read more about smooth convex minimization I recommend [1, Chapter 2].

References

[1] Yurii Nesterov. *Lectures on convex optimization*. Volume 137. Springer, 2018.

[2] Original Nesterov's method was slightly different, but this is not important for our discussion.

Lecture 2: Subgradients

2.1 Subdifferential

Not all convex functions are differentiable. It useful to allow a convex function to take the value $+\infty$. For instance, if f is only defined over a convex closed set C, we may extend it to take values $+\infty$ outside of C. (Check that this extension preserves convexity!) On one hand, this helps us to not bother where f is defined. On the other, we will have problem with its differentiability.

Definition 2.1 — Subdifferential. A *subdifferential* of $f : \mathbb{R}^d \to (-\infty, +\infty]$ at x is called a set

$$\partial f(x) = \{g : f(y) - f(x) \geqslant \langle g, y - x\rangle\}.$$

The elements of $\partial f(x)$ are called *subgradients* of f at x.

We will only consider *proper* convex functions $f : \mathbb{R}^d \to (-\infty, +\infty]$, i.e., those for which $\operatorname{dom}(f) = \{x : f(x) < +\infty\} \neq \varnothing$. From the definition it follows immediately, that if f is proper and convex and $f(x) = +\infty$ for some x, then $\partial f(x) = \varnothing$.

Proposition 2.1 If $f : \mathbb{R}^d \to (-\infty, +\infty]$ is convex and $x \in \operatorname{int}\operatorname{dom}(f)$, then $\partial f(x) \neq \varnothing$.

If you know what a *directional derivative* $f'(x; v)$ is:

$$f'(x; v) = \lim_{t \to 0} \frac{f(x + tv) - f(x)}{t},$$

then the following proposition may be useful.

Proposition 2.2 Let $f : \mathbb{R}^d \to (-\infty, +\infty]$ be convex and $x \in \operatorname{dom} f$. Then

$$\partial f(x) = \{g : f'(x; v) \geqslant \langle g, v\rangle \ \forall v \in \mathbb{R}^d\}.$$

Proposition 2.3 Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex. Then for any $x \in \mathbb{R}^d$ the set $\partial f(x)$ is nonempty, convex, and compact.

Proposition 2.4 If f is convex and differentiable at x, then $\partial f(x) = \{\nabla f(x)\}$.

Proposition 2.5 If f is L-Lipschitz, then $\|g\| \leqslant L$ for all $g \in \partial f(x)$ and all x.

Exercise 2.1 Try to prove Proposition 2.5. ▪

▪ **Example 2.1** Let $f(x) = |x|$ in $\mathbb{R}$. Obviously, this function is not differentiable at $x = 0$, but is so at all other points. It is easy to see that

$$\partial f(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x < 0 \\ [-1, 1] & \text{if } x = 0 \end{cases}$$

▪

▪ **Example 2.2** Let $f(x) = \|x\|_2$. Again, this function is not differentiable only at $x = 0$:

$$\partial f(x) = \begin{cases} \frac{x}{\|x\|_2} & \text{if } x \neq 0 \\ B(0,1) & \text{if } x = 0 \end{cases}$$

▪

▪ **Example 2.3** Let C be a set in $\mathbb{R}^d$ and f be the *indicator* function $\delta_C(x)$, which is defined by

$$\delta_C(x) = 0 \ \text{ if } x \in C, \quad \delta_C(x) = +\infty \ \text{ if } x \notin C.$$

Then

$$\partial \delta(x) = \begin{cases} \{u \colon \langle u, y - x\rangle \leqslant 0 \ \forall y \in C\} & \text{if } x \in C \\ \varnothing \quad \text{otherwise} \end{cases}$$

▪

The set $N_C(x) := \{u \colon \langle u, y - x\rangle \leqslant 0 \ \forall y \in C\}$ from the last example is called the *normal cone* of C at x and plays an important role in the convex analysis.

▪ **Example 2.4** Let $f(x) = \max\{f_1(x), f_2(x)\}$, where f_i is a differentiable function, $i = 1, 2$. Then

$$\partial f(x) = \begin{cases} \nabla f_1(x) & \text{if } f_1(x) > f_2(x) \\ \nabla f_2(x) & \text{if } f_2(x) > f_1(x) \\ [\nabla f_1(x), \nabla f_2(x)] & \text{if } f_1(x) = f_2(x). \end{cases}$$

▪

The last example is not a coincidence. We have even more general result.

Proposition 2.6 Let $f(x) = \max_{i=1,\dots,m} f_i(x)$ with f_i being convex. Then

$$\partial f(x) = \operatorname{conv}\{\partial f_i(x) \colon f_i(x) = f(x)\}.$$

Proposition 2.7 Fermat's rule. Let $f : \mathbb{R}^d \to (-\infty, +\infty]$ be proper and convex. Then x is a minimum of f if and only if $0 \in \partial f(x)$.

Proof. Check that it follows from the definition of the subdifferential in one line. ■

Most often, the basic calculus that involves subdifferentials is the same as with differentials. However, the important exception is the sum-rule:

$$\partial(f + g) \subset \partial f + \partial g$$

and this inclusion can be strict. However, if f is differentiable and $x \in \operatorname{int}\operatorname{dom}(g)$, then

$$\partial(f + g)(x) = \nabla f(x) + \partial g(x)$$

2.2 Subgradient method

Knowing now what a subgradient is, the *subgradient method*[1] can be seen as the most natural extension of the gradient descent to the nonsmooth setting:

$$\boxed{\begin{aligned} g^k &\in \partial f(x^k) \\ x^{k+1} &= x^k - \alpha_k g^k \end{aligned}} \tag{2.1}$$

Unlike with GD, we cannot guarantee that in every iteration $f(x^{k+1}) \leqslant f(x^k)$, in other words, the subgradient method is not a descent method. Its analysis, however, is much simpler than the one of GD, and will be exceptionally important for us in the sequel.

Lemma 2.1 Let $f : \mathbb{R}^d \to (-\infty, +\infty]$ be convex and $\operatorname{argmin} f \neq \varnothing$. Then

$$2\sum_{k=1}^{K} \alpha_k (f(x^k) - f_*) \leqslant \|x^1 - x^*\|^2 + \sum_{k=1}^{K} \alpha_k^2 \|g^k\|^2. \tag{2.2}$$

Proof. Let x^* be any solution of $\min_x f(x)$. We start from expanding the norm

$$\begin{aligned} \|x^{k+1} - x^*\|^2 &= \|x^k - \alpha_k g^k - x^*\|^2 \\ &= \|x^k - x^*\|^2 - 2\alpha_k \langle g^k, x^k - x^* \rangle + \alpha_k^2 \|g^k\|^2 \\ &\leqslant \|x^k - x^*\|^2 - 2\alpha_k (f(x^k) - f(x^*)) + \alpha_k^2 \|g^k\|^2, \end{aligned}$$

where in the last equation we used the definition of the subgradient. From this it follows

$$2\alpha_k (f(x^k) - f_*) \leqslant \|x^k - x^*\|^2 - \|x^{k+1} - x^*\|^2 + \alpha_k^2 \|g^k\|^2.$$

Summing such inequalities for $k = 1, \ldots, K$, we get

$$2\sum_{k=1}^{K} \alpha_k (f(x^k) - f_*) \leqslant \|x^1 - x^*\|^2 - \|x^{k+1} - x^*\|^2 + \sum_{k=1}^{K} \alpha_k^2 \|g^k\|^2$$

and the desired inequality follows immediately. ∎

Corollary 2.1 Let $\|x^1 - x^*\| \leqslant R$, $A_K = \sum_{k=1}^K \alpha_k$ and $\bar{x}^K = \frac{1}{A_k} \sum_{k=1}^K \alpha_k x^k$ and suppose that $\|g\| \leqslant G$ for all $g \in \partial f(x)$ and for all x. Then

$$f(\bar{x}^K) - f_* \leqslant \frac{R^2 + G^2 \sum_{k=1}^K \alpha_k^2}{2\sum_{k=1}^K \alpha_k} \tag{2.3}$$

Proof. By Jensen's inequality

$$A_K f(\bar{x}^k) \leqslant \sum_{k=1}^{K} \alpha_k f(x^k).$$

Hence,

$$2A_K (f(\bar{x}^K) - f_*) \leqslant 2\sum_{k=1}^{K} \alpha_k (f(x^k) - f_*) \leqslant \|x^1 - x^*\|^2 + \sum_{k=1}^{K} \alpha_k^2 \|g^k\|^2 \leqslant R^2 + G^2 \sum_{k=1}^{K} \alpha_k^2$$

from which the desired inequality follows. ∎

There are a few basic strategies for selecting α_k.

[1] Without mentioning it every time, we assume that $\partial f(x^k) \neq \varnothing$ for all k.

A fixed stepsize. Let us use a fixed stepsize $\alpha_k = \alpha$ for all $k = 1, \dots, K$. We would like to optimize the RHS in (2.3) to find an optimal α. In other words, we want to solve

$$\min_{\alpha>0} \frac{R^2}{2\alpha K} + \frac{G^2\alpha}{2}$$

and deduce that the "best" $\alpha = \frac{R}{G\sqrt{K}}$, which yields the bound

$$f(\bar{x}^K) - f_* \leqslant \frac{RG}{\sqrt{K}}. \tag{2.4}$$

It is easy to see that this is equivalent to $O\left(\frac{1}{\varepsilon^2}\right)$ guarantee for $f(\bar{x}^K) - f_* \leqslant \varepsilon$. This way of choosing a stepsize is called a *finite horizon*. We fix in advance the number of iterations K that the algorithm is supposed to run. If we run it for less or more, that stepsize may be not the best. And the obtained guarantee holds only for this particular $\bar{x}^K$ with a fixed K.

Dynamic stepsizes. For the bound (2.3) we want to have

$$\sum_{k=1}^{\infty} \alpha_k = +\infty, \qquad \frac{\sum_{k=1}^{\infty} \alpha_k^2}{\sum_{k=1}^{\infty} \alpha_k} < +\infty.$$

The most natural choice for α_k is then $\alpha_k = \frac{c}{\sqrt{k}}$, where $k \geqslant 1$ and $c > 0$.

Now we will only have a rough sketch, as later we will improve our analysis anyway. It should be obvious that because of

$$\sum_{k=1}^{K} \frac{1}{\sqrt{k}} \sim \sqrt{K}, \qquad \sum_{k=1}^{K} \frac{1}{k} \sim \log K,$$

we obtain the rate

$$f(\bar{x}^K) - f_* = O\left(\frac{\log K}{\sqrt{K}}\right).$$

The difference with (2.4) is that we now have an extra log-factor, but on the other hand, this rate holds for arbitrary K.

R Instead of considering $\bar{x}^K$ we could use the "best" iterate $\min_{1 \leqslant k \leqslant K} f(x^k)$.

Comments

For a more in-depth reading on subdifferentials/subgradients I recommend anything from of [1, Chapter 3], [4, Chapter VI], [2, Chapter 3], and lecture notes [3]. On the subgradient method I recommend [1, Chapter 8] or again the lecture notes [3].

References

[1] Amir Beck. *First-order methods in optimization*. SIAM, 2017.

[2] Dimitri Bertsekas. *Convex optimization algorithms*. Athena Scientific, 2015.

[3] John C Duchi. "Introductory lectures on stochastic optimization". In: *The mathematics of data* 25 (2018), pages 99–186. URL: https://web.stanford.edu/~jduchi/PCMIConvex/Duchi16.pdf.

[4] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex analysis and minimization algorithms I: Fundamentals*. Volume 305. Springer science & business media, 1996.

Lecture 3: Projected subgradient method

It is often the case that we have to solve $\min f(x)$ not over the whole space $\mathbb{R}^d$, but over some subset of $\mathbb{R}^d$. In other words, given a convex and closed set $C \subset \mathbb{R}^d$, we are interested in

$$\min_{x \in C} f(x).$$

This is called a *constrained optimization* problem. A vector x is called *feasible* if $x \in C$ and *infeasible* otherwise.

Definition 3.1 — Local/global minimum. A point $x \in C$ is called a *local minimum* of f on C, if there exist an open ball $U(x, r)$ such that

$$f(x) \leqslant f(y) \quad \forall y \in C \cap U(x, r).$$

If $U(x, r) = \mathbb{R}^d$ above, then x is called a *global minimum* of f on C.

Before proceeding with algorithms, it is important to understand what is the optimality condition for this class of problems. Consider the problem

$$\min f(x) := x^2 \quad \text{subject to} \quad x \in [2, 3].$$

It is clear that $x^* = 2$ is the minimum of f, however, $f'(2) \neq 0$.

Proposition 3.1 Let $f : \mathbb{R}^d \to \mathbb{R}$ be a proper convex function and $C \subset \mathbb{R}^d$ be a closed convex set. Then $x \in C$ is minimizer of f over C if and only if

$$0 \in \partial f(x) + N_C(x).$$

Proof. If for some x, $0 \in \partial f(x) + N_C(x)$, then it follows immediately that x is a minimizer (check this!). The opposite direction is slightly more involved, but the main idea is the following. If $x \in \operatorname{argmin}_C f$, then $x \in \operatorname{argmin}_{\mathbb{R}^d}(f + \delta_C)$. Then by Proposition 2.7 and Example 2.3,

$$0 \in \partial(f + \delta_C)(x) \subset \partial f(x) + \partial \delta_C(x) = \partial f(x) + N_C(x).$$

Hence, to finish the proof, we need to show that instead of a strict inclusion, we actually have an equality:

$$\partial(f + \delta_C)(x) = \partial f(x) + \partial \delta_C(x).$$

This is proved by the separation theorem. ∎

Corollary 3.1 Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and differentiable and $C \subset \mathbb{R}^d$ be a closed convex set. Then $x \in C$ is a minimizer of f over C if and only if

$$\langle \nabla f(x), y - x \rangle \geqslant 0 \quad \forall y \in C.$$

Proof. This follows directly from Proposition 3.1 and the definition of the normal cone. ■

3.1 Metric projection

Definition 3.2 — Metric projection. Given a closed convex set $C \subset \mathbb{R}^d$ and $u \in \mathbb{R}^d$, a *metric projection* (also orthogonal projection) of u onto C is the solution of

$$\operatorname*{argmin}_{x \in C} \|x - u\|_2. \tag{3.1}$$

We denote this solution by $P_C u$.

First, we should convince ourselves that this definition makes sense.

Theorem 3.1 Given a closed convex set $C \subset \mathbb{R}^d$, the operator P_C is always well-defined. Moreover, $P_C u = x$ if and only if

$$\langle x - u, y - x \rangle \geqslant 0, \quad \forall y \in C.$$

Proof. Note that the problem

$$\min_{x \in C} \frac{1}{2} \|x - u\|_2^2. \tag{3.2}$$

is equivalent to (3.1). Let's denote the objective of the latter problem by f. The function f is strongly convex, hence (3.2) has always a unique solution. Therefore, $P_C u$ is well-defined for every u. Now, using optimality condition for $\min_{x \in C} f(x)$ from Corollary 3.1, we get

$$\langle \nabla f(x), y - x \rangle \geqslant 0 \quad \iff \quad \langle x - u, y - x \rangle \geqslant 0, \qquad \forall y \in C$$

and the proof is complete. ■

▪ **Example 3.1 — Projection onto the ball** $B(0, r)$. We have

$$P_{B(0,r)} u = \begin{cases} \frac{u}{\|u\|} r, & \text{if } \|u\| > r \\ u, & \text{otherwise.} \end{cases}$$

▪

▪ **Example 3.2 — Projection onto the hyperplane** $H = \{x : \langle a, x \rangle = b\}$.

$$P_H u = u - \frac{b - \langle a, u \rangle}{\|a\|^2} a.$$

▪

▪ **Example 3.3 — Projection onto the subspace** $\ker A$. Let A be a $m \times n$ matrix with $\operatorname{rank}(A) = m$. We have to solve

$$\min \frac{1}{2} \|x - u\|^2 \quad \text{s.t.} \quad Ax = 0.$$

By KKT optimality condition, we have

$$x - u + A^\top \lambda = 0 \qquad \text{and} \qquad Ax = 0.$$

Solving this system, we get

$$P_C u = x^* = \left(I - A^\top (AA^\top)^{-1} A\right) u.$$

■

Lemma 3.1 A projection P_C is a *nonexpansive* operator, that is

$$\|P_C u - P_C v\|_2 \leqslant \|u - v\|_2, \quad \forall u, v.$$

Proof. Let $\bar{u} = P_C u$ and $\bar{v} = P_C v$. Then by Theorem 3.1,

$$\langle \bar{u} - u, x - \bar{u} \rangle \geqslant 0 \quad \forall x \in C \qquad \text{and} \qquad \langle \bar{v} - v, y - \bar{v} \rangle \geqslant 0 \quad \forall y \in C.$$

Setting $x = \bar{v}$ and $y = \bar{u}$ in correspondent equations and adding them, we get

$$\langle (\bar{u} - u) - (\bar{v} - v), \bar{v} - \bar{u} \rangle \geqslant 0 \quad \Longleftrightarrow \quad \|\bar{u} - \bar{v}\|_2^2 \leqslant \langle \bar{u} - \bar{v}, u - v \rangle.$$

To conclude, it only remains to apply Cauchy-Schwarz: $\langle \bar{u} - \bar{v}, u - v \rangle \leqslant \|\bar{u} - \bar{v}\|_2 \|u - v\|_2$. ■

3.2 Projected subgradient method

A subgradient method has the following interpretation: we choose a $g^k \in \partial f(x^k)$, approximate f by a quadratic model around x^k and minimize this quadratic model, that is

$$x^{k+1} = \operatorname*{argmin}_x \left\{ f(x^k) + \langle g^k, x - x^k \rangle + \frac{1}{2\alpha_k} \|x - x^k\|_2^2 \right\}.$$

Now our problem of interest is $\min_{x \in C} f(x)$ for a convex closed set C. So it is only natural to consider the following update:

$$x^{k+1} = \operatorname*{argmin}_{x \in C} \left\{ f(x^k) + \langle g^k, x - x^k \rangle + \frac{1}{2\alpha_k} \|x - x^k\|_2^2 \right\}.$$

Let's transform the above optimization problem. Notice that $f(x^k)$ is just a constant that doesn't influence on the solution. Hence, we have

$$\begin{aligned} x^{k+1} &= \operatorname*{argmin}_{x \in C} \left\{ \langle g^k, x - x^k \rangle + \frac{1}{2\alpha_k} \|x - x^k\|_2^2 \right\} \\ &= \operatorname*{argmin}_{x \in C} \frac{1}{2\alpha_k} \|x - x^k + \alpha_k g^k\|_2^2 \\ &= P_C(x^k - \alpha_k g^k). \end{aligned}$$

To summarize, the projected subgradient method is given by

$$\boxed{\begin{aligned} g^k &\in \partial f(x^k) \\ x^{k+1} &= P_C(x^k - \alpha_k g^k). \end{aligned}} \tag{3.3}$$

We will always assume that in every iteration $\partial f(x^k) \neq \varnothing$. This, for instance, can be guaranteed if $C \subset \operatorname{int} \operatorname{dom}(f)$. This time we analyze the method under the assumption that (α_k) is nonincreasing.

Theorem 3.2 Let C be a convex and compact set with $R = \operatorname{diam}(C)$, $f : \mathbb{R}^d \to (-\infty, +\infty]$ be a convex function with $\|g\|_2 \leqslant G$ for all $g \in \partial f(x)$ and for all $x \in C$. Suppose that (α_k) is a nonincreasing sequence. Then

$$\sum_{k=1}^{K}(f(x^k)-f_*) \leqslant \frac{R^2}{2\alpha_K} + \sum_{k=1}^{K}\frac{\alpha_k G^2}{2}. \tag{3.4}$$

Proof. Let $x^* \in \operatorname{argmin}_{x\in C} f(x)$. Since $P_C x^* = x^*$, by Lemma 3.1 we have that

$$\begin{aligned}
\|x^{k+1}-x^*\|_2^2 &= \|P_C(x^k-\alpha_k g^k)-P_C x^*\|_2^2 \\
&\leqslant \|x^k-x^*-\alpha_k g^k\|_2^2 \\
&= \|x^k-x^*\|^2 - 2\alpha_k\langle g^k, x^k-x^*\rangle + \alpha_k^2\|g^k\|^2 \\
&\leqslant \|x^k-x^*\|^2 - 2\alpha_k(f(x^k)-f(x^*)) + \alpha_k^2\|g^k\|^2 \qquad \text{//By convexity.}
\end{aligned}$$

This implies

$$\begin{aligned}
f(x^k)-f_* &\leqslant \frac{1}{2\alpha_k}\|x^k-x^*\|^2 - \frac{1}{2\alpha_k}\|x^{k+1}-x^*\|^2 + \frac{\alpha_k}{2}\|g^k\|^2 \\
&= \frac{1}{2\alpha_{k-1}}\|x^k-x^*\|^2 - \frac{1}{2\alpha_k}\|x^{k+1}-x^*\|^2 + \left(\frac{1}{2\alpha_k}-\frac{1}{2\alpha_{k-1}}\right)\|x^k-x^*\|^2 + \frac{\alpha_k}{2}\|g^k\|^2.
\end{aligned}$$

Summing this inequality over $k = 1, \ldots, K$, we get

$$\begin{aligned}
\sum_{k=1}^{K}(f(x^k)-f_*) &\leqslant \frac{1}{2\alpha_0}\|x^0-x^*\|^2 - \frac{1}{2\alpha_K}\|x^{k+1}-x^*\|^2 + \sum_{k=1}^{K}\left(\frac{1}{2\alpha_k}-\frac{1}{2\alpha_{k-1}}\right)\|x^k-x^*\|^2 + \sum_{k=1}^{K}\frac{\alpha_k}{2}\|g^k\|^2 \\
&\leqslant \frac{1}{2\alpha_0}R^2 + \sum_{k=1}^{K}\left(\frac{1}{2\alpha_k}-\frac{1}{2\alpha_{k-1}}\right)R^2 + \sum_{k=1}^{K}\frac{\alpha_k G^2}{2} \qquad \text{//Because } \alpha_k \leqslant \alpha_{k-1} \\
&\leqslant \frac{R^2}{2\alpha_K} + \sum_{k=1}^{K}\frac{\alpha_k G^2}{2}.
\end{aligned} \tag{3.5}$$

■

Corollary 3.2 Let in the previous statement $\alpha_k = \frac{c}{\sqrt{k}}$, with $c > 0$. Then for $\bar{x}^K = \frac{1}{K}\sum_{k=1}^{K} x^k$ it holds

$$f(\bar{x}^K)-f_* \leqslant \frac{R^2}{2c\sqrt{K}} + \frac{cG^2}{2\sqrt{K}}. \tag{3.6}$$

Proof. Notice that

$$\sum_{k=1}^{K}\frac{1}{\sqrt{k}} \leqslant \sum_{k=1}^{K}\frac{2}{\sqrt{k}+\sqrt{k-1}} = \sum_{k=1}^{K}2(\sqrt{k}-\sqrt{k-1}) \leqslant 2\sqrt{K}.$$

Then for $\alpha_k = \frac{c}{\sqrt{k}}$, with $c > 0$, we deduce from (3.4)

$$K(f(\bar{x}^K)-f_*) \leqslant \sum_{k=1}^{K}(f(x^k)-f_*) \leqslant \frac{R^2\sqrt{K}}{2c} + c\sqrt{K}G^2$$

or, after dividing over K,

$$f(\bar{x}^K)-f_* \leqslant \frac{R^2}{2c\sqrt{K}} + \frac{cG^2}{\sqrt{K}}.$$

■

Corollary 3.3 Let $\alpha_k = \frac{R}{G\sqrt{2k}}$. Then for $\bar{x}^K = \frac{1}{K}\sum_{k=1}^K x^k$ it holds

$$f(\bar{x}^K) - f_* \leqslant \sqrt{\frac{2}{K}}RG \qquad \text{for any } K \geqslant 1. \tag{3.7}$$

Proof. We only need to optimize over c in (3.6). ■

The bound (3.7) is almost the same as in (2.4), but now it holds for arbitrary K. Note that even if the optimal constant $c = \frac{R}{G}$ is not available, we would still get $O\left(\frac{1}{\sqrt{K}}\right)$ rate, as (3.6) indicates. If we want to find $x \in C$ such that $f(x) - f_* \leqslant \varepsilon$, we need to run the method for $O\left(\frac{1}{\varepsilon^2}\right)$ iterations.

For the class of convex functions with bounded subgradients acceleration is not possible, and the $O\left(\frac{1}{\sqrt{K}}\right)$ rate is the best what we can get in general.

Comments

You can read more on the projected subgradient method in [1, Chapter 8] and [2].

References

[1] Amir Beck. *First-order methods in optimization*. SIAM, 2017.

[2] John C Duchi. "Introductory lectures on stochastic optimization". In: *The mathematics of data* 25 (2018), pages 99–186. URL: https://web.stanford.edu/~jduchi/PCMIConvex/Duchi16.pdf.

Lecture 4: Stochastic gradient/subgradient methods

Consider our basic problem $\min_x f(x)$, where f is differentiable function. Stochastic gradient descent (SGD) is the method of the form

$$x^{k+1} = x^k - \alpha_k(\nabla f(x^k) + \xi^k), \tag{4.1}$$

where ξ_k is a random vector, also called a noise. The vector $g^k = \nabla f(x^k) + \xi^k$ is called a *stochastic estimator* of the gradient $\nabla f(x^k)$, or, for brevity, a *stochastic gradient*. If $\mathbf{E}[\xi_k] = 0$, then $\mathbf{E}\left[g^k\right] = \nabla f(x^k)$ and in this case we call g^k an *unbiased estimator* of $\nabla f(x^k)$.

R The name SGD is a misnomer — this method doesn't have a descent property. The name only reflects that it is a stochastic extension of the gradient descent.

Why do we need to consider such schemes or, in other words, what can be the source of this noise? The noise ξ^k can originated from. It may be an intrinsic part of the problem, say we want to minimize f, given by

$$f(x) = \mathbf{E}[F(x,\xi)] = \int F(x,s)\,\mathrm{d}P(s).$$

Computing $\nabla f(x)$ is not possible in general, but we can use its estimators $\nabla F(x,\xi)$. This is a subject of *stochastic programming*, and we won't consider it here.

However, our main motivation to inject a random noise ξ^k into a deterministic scheme is to make an algorithm more **efficient**. Does it sound strange to you?

Algorithm complexity. In general, it is not the number of iterations in which we are interested to find a solution. More often than not, it is the time/money/energy we have to spend to find it. This leads us to the notion of the *complexity* of an algorithm. There are many ways how one can measure complexity, but essentially we should measure what matters, without going to a finer scale when it is not necessary. For instance, for the class of first-order methods, it makes sense to measure the complexity of an algorithm in the number of gradients/subgradients we have to compute. Usually, this is the "expensive" operation and for simplicity we ignore the rest, such as vector addition.

For now it is not very helpful, after all for GD the number of iterations and the number of needed gradients are the same numbers, which is $O\left(\frac{1}{\varepsilon}\right)$. But let's change now our perspective a bit. Suppose we want to solve

$$\min f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x), \tag{4.2}$$

where we assume that each f_i is convex and differentiable and that evaluation of every f_i costs roughly the same. Problem (4.2) is called a *finite-sum* minimization. In this case, the cost of one gradient $\nabla f(x)$ equals to n gradients of $\nabla f_i(x)$. So if we measure the complexity of GD in terms of $\nabla f_i(x)$, we get

$$\text{GD complexity } = O\left(\frac{1}{\varepsilon}\right) \times \text{Cost}(\nabla f(x)) = O\left(\frac{n}{\varepsilon}\right) \times \text{Cost}(\nabla f_i(x)).$$

In many applications, n can be a very large number and at the same time, perhaps we don't need to solve $\min f(x)$ up to a high accuracy. Therefore, to have an efficient algorithm, we must take everything into consideration: not only ε, but also n.

SGD for the finite-sum problem. Convince yourself that the following method

$$\boxed{\begin{aligned} &\text{Sample } i_k \sim \text{Unif}\{1,2,\dots,n\} \\ &x^{k+1} = x^k - \alpha_k \nabla f_{i_k}(x^k) \end{aligned}} \tag{4.3}$$

is a particular instance of (4.1).

Exercise 4.1 Prove in (4.3) that $\nabla f_{i_k}(x^k)$ is an unbiased estimator of $\nabla f(x^k)$. ∎

Conditional expectation. When discussing an unbiased estimator above, we took it for granted that we could compute $\mathbf{E}\left[\nabla f_{i_k}(x^k)\right]$. Actually, to do that, we implicitly assumed that x^k is already given to us as a deterministic vector. However, while analyzing the algorithm, this won't be true. Only the initial point x^1 is deterministic, the rest $x^2, \dots, x^k$ are all random vectors. Because of so much randomness, in general, expressions like $\mathbf{E}\left[\nabla f_{i_k}(x^k)\right]$, where the expectation is taken with respect to all randomness encountered up to the k-th iteration, are intractable.

A proper definition[1] of the conditional expectation is a bit technical, but for us the one you understand intuitively should suffice. The easiest way to get it is from examples.

▪ **Example 4.1** Let $\Omega = \{a, b, c, d, e, d\}$, $\mathscr{F} = 2^\Omega$, and $\mathbf{P}$ be uniform. Let define random variables X, Y, Z as follows

	a	b	c	d	e	f
X	1	3	2	4	0	7
Y	2	2	3	3	1	1
Z	4	4	4	4	5	5

In other words, $X(a) = 1$, $X(b) = 3$, and so on. It should be obvious that $\mathbf{E}[X|Y=2] = 2$, $\mathbf{E}[X|Z=4] = 2.5$. In general, if we condition X wrt Y, it means that we condition it wrt $\mathscr{F}_2 = \sigma(\{a,b\},\{c,d\},\{e,f\})$, that is σ-algebra[2] is generated by the subsets of Ω, where Y is constant. Similarly, conditioning on Z means that we condition wrt $\mathscr{F}_1 = \sigma(\{a,b,c,d\},\{e,f\})$. Convince yourself that

$$\mathbf{E}[X \mid \mathscr{F}_1] = \mathbf{E}[\mathbf{E}[X \mid \mathscr{F}_2] \mid \mathscr{F}_1], \tag{4.4}$$

or, which is the same but often simpler to write, that

$$\mathbf{E}[X \mid Z] = \mathbf{E}[\mathbf{E}[X \mid Y] \mid Z].$$

▪

[1] Compared to the classroom lecture, here we will take a closer look at conditional expectation. You can skip this part if you are confident in your knowledge.

[2] also known as σ-field

For us, σ-algebras are information. Consider $\mathbf{E}[X \mid \mathscr{G}]$ with two extreme cases. If we know nothing, then $\mathscr{G} = \{\varnothing, \Omega\}$ and $\mathbf{E}[X \mid \mathscr{G}] = \mathbf{E}[X]$. If we know everything, then $\mathscr{G} = \mathscr{F}$ and $\mathbf{E}[X \mid \mathscr{G}] = X$.

Equation (4.4) is not accidental.

Proposition 4.1 Tower property. Let $(\Omega, \mathscr{F}, \mathbf{P})$ be a probability space. Suppose two σ-algebras $\mathscr{F}_1, \mathscr{F}_2$ satisfy $\mathscr{F}_1 \subset \mathscr{F}_2 \subset \mathscr{F}$. Then

$$\mathbf{E}[X \mid \mathscr{F}_1] = \mathbf{E}[\mathbf{E}[X \mid \mathscr{F}_2] \mid \mathscr{F}_1]. \tag{4.5}$$

Corollary 4.1 — Law of total expectation. Suppose that X, Y are two random variables defined on a probability space $(\Omega, \mathscr{F}, \mathbf{P})$. Then

$$\mathbf{E}[X] = \mathbf{E}[\mathbf{E}[X \mid Y]]. \tag{4.6}$$

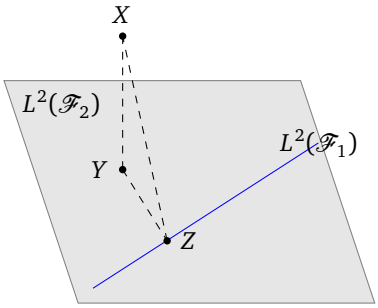
Proof. This follows from Proposition 4.1 by taking $\mathscr{F}_1 = \{\varnothing, \Omega\}$ and $\mathscr{F}_2 = \sigma(Y)$. ∎

There is a nice geometric interpretation of the conditional expectation $\mathbf{E}[X \mid \mathscr{G}]$. Suppose that X is square-integrable: $X \in L^2(\mathscr{F})$ and that $\mathscr{G} \subset \mathscr{F}$. Then $\mathbf{E}[X \mid \mathscr{G}]$ is the best mean-square approximation of X among $\mathscr{G}$-measurable random variables, that is

$$\mathbf{E}[X \mid \mathscr{G}] = \operatorname*{argmin}_{Y \in L^2(\mathscr{G})} \|Y - X\|^2.$$

In other words, $\mathbf{E}[X \mid \mathscr{G}]$ is the projection of X onto the subspace of $L^2(\mathscr{G})$. The picture below illustrates the identity (4.5) for a square-integrable random variable X.

$$\begin{aligned} Y &= \mathbf{E}[X \mid \mathscr{F}_2] = P_{L^2(\mathscr{F}_2)} X \\ Z &= \mathbf{E}[X \mid \mathscr{F}_1] = P_{L^2(\mathscr{F}_1)} X \\ P_{L^2(\mathscr{F}_1)} P_{L^2(\mathscr{F}_2)} X &= P_{L^2(\mathscr{F}_1)} X \end{aligned}$$



Of course, it's evident that $\mathbf{E}[\mathbf{E}[X \mid \mathscr{F}_1] \mid \mathscr{F}_2] = \mathbf{E}[X \mid \mathscr{F}_1]$ also holds if $\mathscr{F}_1 \subset \mathscr{F}_2$. However, this isn't particularly interesting, as even the figure above suggests.

Exercise 4.2 Random sums. Let $X_1, X_2, \ldots$ be iid random variables, and let M be a random variable taking the values $0, 1, 2, \ldots$ that is independent of the (X_k). We wish to study the sum $S = \sum_{i=0}^{M} X_i$. Assuming we know $\mathbf{E}X$ and $\mathbf{E}M$, compute $\mathbf{E}S$. ∎

Revisiting SGD for finite-sum. Now, let's revisit the equation (4.3) and incorporate the fact that x^k is also a random vector. As a random vector, x^k induces a σ-algebra encompassing all the randomness up to step k, including $i_1, \ldots, i_{k-1}$. Thus, now we can state

$$\mathbf{E}\left[\nabla f_{i_k}(x^k) \,\middle|\, x^k\right] = \mathbf{E}\left[\nabla f_{i_k}(x^k) \,\middle|\, \sigma(i_1, \ldots, i_{k-1})\right] = \nabla f(x^k).$$

As a shortcut, we will often denote the conditional expectation above as

$$\mathbf{E}_k\left[\nabla f_{i_k}(x^k)\right] = \nabla f(x^k).$$

4.1 Stochastic subgradient algorithm

The same discussion is applied verbatim to the subgradient algorithm. Its stochastic extension is straightforward:

$$\boxed{\begin{aligned}&\text{Choose } g^k \text{ s.t. } \mathbf{E}_k\left[g^k\right] \in \partial f(x^k)\\ &x^{k+1} = x^k - \alpha_k g^k\end{aligned}} \tag{4.7}$$

We don't specify how to choose such g^k, since it depends on the problem at hand. For instance, for a finite-sum problem

$$\min_x \frac{1}{n}\sum_{i=1}^n f_i(x),$$

with each $f_i : \mathbb{R}^d \to \mathbb{R}$ being convex, we can do the same as in (4.3). This gives us **stochastic subgradient method for finite-sum:**

$$\boxed{\begin{aligned}&\text{Sample } i_k \sim \text{Unif}\{1,2,\dots,n\}\\ &\text{Choose } g^k \in \partial f_{i_k}(x^k)\\ &x^{k+1} = x^k - \alpha_k g^k\end{aligned}} \tag{4.8}$$

Theorem 4.1 Suppose that $f : \mathbb{R}^d \to \mathbb{R}$ is convex, $\mathbf{E}_k\left[\|g^k\|^2\right] \leqslant G^2$, and $\|x^1 - x^*\| \leqslant R$. Then the algorithm (4.7) satisfies

$$\mathbf{E}\left[f(\bar{x}^k) - f_*)\right] \leqslant \frac{R^2}{2\sum_{k=1}^K \alpha_k} + \frac{G^2 \sum_{k=1}^K a_k^2}{2\sum_{k=1}^K \alpha_k},$$

where $\bar{x}^K = \frac{1}{A_K}\sum_{k=1}^K \alpha_k x_k$ and $A_K = \sum_{k=1}^K \alpha_k$.

Proof. The proof will be very much in spirit of the proof of Lemma 2.1. Let x^* be any solution of $\min_x f(x)$. As before, we expand the norm

$$\begin{aligned}\|x^{k+1} - x^*\|^2 &= \|x^k - \alpha_k g^k - x^*\|^2\\ &= \|x^k - x^*\|^2 - 2\alpha_k\langle g^k, x^k - x^*\rangle + \alpha_k^2\|g^k\|^2.\end{aligned} \tag{4.9}$$

This time we cannot use convexity inequality to bound the term $\langle g^k, x^k - x^*\rangle$. Taking expectation from both sides of above equation, we obtain

$$\mathbf{E}\left[\|x^{k+1} - x^*\|^2\right] = \mathbf{E}\left[\|x^k - x^*\|^2\right] - 2\alpha_k\mathbf{E}\left[\langle g^k, x^k - x^*\rangle\right] + \alpha_k^2\mathbf{E}\left[\|g^k\|^2\right].$$

Let $\widehat{\nabla} f(x^k) := \mathbf{E}_k\left[g^k\right] \in \partial f(x^k)$. Applying the law of total expectation, we have

$$\mathbf{E}\left[\langle g^k, x^k - x^*\rangle\right] = \mathbf{E}\left[\mathbf{E}_k\left[\langle g^k, x^k - x^*\rangle\right]\right] = \mathbf{E}\left[\langle \widehat{\nabla} f(x^k), x^k - x^*\rangle\right] \geqslant \mathbf{E}\left[f(x^k) - f(x^*)\right].$$

Now using this inequality in (4.9), we get

$$\|x^{k+1} - x^*\|^2 \leqslant \|x^k - x^*\|^2 - 2\alpha_k(f(x^k) - f_*) + \alpha_k^2\|g^k\|^2,$$

which we rewrite as

$$2\alpha_k(f(x^k) - f_*) \leqslant \|x^k - x^*\|^2 - \|x^{k+1} - x^*\|^2 + \alpha_k^2\|g^k\|^2.$$

Summing this inequality over $k = 1, \dots, K$ yields

$$\sum_{k=1}^{K} \alpha_k \mathbf{E}\left[f(x^k) - f_*\right] \leqslant \frac{1}{2}\mathbf{E}\left[\|x^1 - x^*\|^2\right] + \frac{1}{2}\sum_{k=1}^{K} \alpha_k^2 \mathbf{E}\left[\|g^k\|^2\right] \leqslant \frac{R^2}{2} + \frac{G^2 \sum_{k=1}^{K} a_k^2}{2}.$$

Jensen's inequality completes the proof. ■

Exercise 4.3 What should we say about (α_k) to make the derivation above completely rigorous? ▪

Exercise 4.4 Explain how we bounded $\mathbf{E}\left[\|g^k\|^2\right]$ in the last inequality of the proof of Theorem 4.1. ▪

With the same proof as for the subgradient method, we can also derive $O(\frac{1}{\sqrt{K}})$ convergence rate for SGD.

Corollary 4.2 — Fixed stepsize. If $\alpha_k = \frac{R}{G\sqrt{K}}$ for all $k \in 1, \dots, K$ and $\bar{x}^K = \frac{1}{K}\sum_{k=1}^{K} x^k$, then

$$\mathbf{E}\left[f(\bar{x}^K) - f_*\right] \leqslant \frac{RG}{\sqrt{K}}.$$

Lecture 5: SGD specifications

5.1 Projected SGD

We can also consider the projected stochastic subgradient method

$$\boxed{\begin{aligned} &\text{Choose } g^k \in \partial f_{i_k}(x^k) \\ &x^{k+1} = P_C(x^k - \alpha_k g^k). \end{aligned}} \tag{5.1}$$

The main result below is basically the same as in Theorem 3.2, only with the expectation.

Theorem 5.1 Let C be a convex and compact set with $R = \operatorname{diam}(C)$, f be a convex function with $\mathbf{E}\left[\|g^k\|_2^2\right] \leqslant G^2$. Suppose that (α_k) is a nonincreasing sequence and $\bar{x}^K = \frac{1}{K}\sum_{k=1}^K x^k$. Then

$$\mathbf{E}\left[f(\bar{x}^K) - f_*\right] \leqslant \frac{R^2}{2K\alpha_K} + \frac{1}{2K}\sum_{k=1}^K \alpha_k G^2 \tag{5.2}$$

Natuarally, its proof is also very similar to the one of Theorem 3.2.

Corollary 5.1 Let assumptions of Theorem 5.1 hold. Then (5.1) with stepsize $\alpha_k = \frac{R}{G\sqrt{k}}$ satisfies

$$\mathbf{E}\left[f(\bar{x}^K) - f_*\right] \leqslant \frac{3RG}{2\sqrt{K}}.$$

Proof. It is similar to the proof of (3.6) in Corollary 3.2, which this time we apply to (5.2) with a simpler (and slightly less optimal) choice of $c = 1$. ■

5.1.1 Probabilistic guarantees

Results in expectation are useful, but sometimes we would like to have some probabilistic guarantees: what is the likelihood that $\bar{x}^k$ produced by SGD will satisfy the above bound?

If we additionally assume that all stochastic subgradients are bounded (without expectation), then we get the following high-probability bound.

Theorem 5.2 In the condition of Theorem 5.1 suppose additionally that $\|g\|_2 \leqslant G$ for all

stochastic subgradients g. Then for any $s > 0$,

$$f(\bar{x}^K) - f(x^*) \leqslant \frac{R^2}{2K\alpha_K} + \sum_{k=1}^{K} \frac{\alpha_k G^2}{2} + \frac{RG}{\sqrt{K}} s$$

with probability at least $1 - e^{-\frac{1}{2}s^2}$.

In other words, on the right-hand side we have the same bound as in Theorem 5.1 plus another $O\left(\frac{1}{\sqrt{K}}\right)$ expression. The high-probability result means that we sacrifice only a little in the theoretical bound for the expectation, but instead get a very high probability that our inequality will hold.

To prove this result we need some concentration bounds tools.

Theorem 5.3 Azuma-Hoeffding Inequality. Let $X_1, X_2, \ldots, X_n$ be a sequence of random variables that satisfy

(i) $|X_k| \leqslant B$ for every k;

(ii) $\mathbf{E}[X_k \mid X_1, \ldots, X_{k-1}] = 0$ for every k.

Then

$$\mathbf{P}\left[\sum_{i=1}^{n} X_i \geqslant t\right] \leqslant \exp\left(-\frac{2t^2}{nB^2}\right).$$

Proof of Theorem 5.2. We proceed very much as in Theorem 3.2, just this time we separate noise $\xi_k = \widehat{\nabla} f(x^k) - g^k$, where $\widehat{\nabla} f(x^k) = \mathbf{E}_k\left[g^k\right] \in \partial f(x^k)$. Let $x^* \in \operatorname{argmin}_{x \in C} f(x)$. Since $P_C x^* = x^*$, by Lemma 3.1 we have that

$$\begin{aligned}
\|x^{k+1} - x^*\|_2^2 &= \|P_C(x^k - \alpha_k g^k) - P_C x^*\|_2^2 \\
&\leqslant \|x^k - x^* - \alpha_k g^k\|_2^2 \\
&= \|x^k - x^*\|^2 - 2\alpha_k \langle g^k, x^k - x^* \rangle + \alpha_k^2 \|g^k\|^2 \\
&= \|x^k - x^*\|^2 - 2\alpha_k \langle \widehat{\nabla} f(x^k), x^k - x^* \rangle + \alpha_k^2 \|g^k\|^2 + 2\alpha_k \langle \widehat{\nabla} f(x^k) - g^k, x^k - x^* \rangle \\
&\leqslant \|x^k - x^*\|^2 - 2\alpha_k (f(x^k) - f(x^*)) + \alpha_k^2 \|g^k\|^2 + 2\alpha_k \langle \xi^k, x^k - x^* \rangle \qquad \text{//By convexity.}
\end{aligned}$$

We do exactly as before, just now keeping one extra term $2\alpha_k \langle \xi^k, x^k - x^* \rangle$. Thus, instead of repeating the same calculations, we just reuse them and include the latter term with noise. This gives us

$$\sum_{k=1}^{K} (f(x^k) - f_*) \leqslant \frac{R^2}{2\alpha_K} + \sum_{k=1}^{K} \frac{\alpha_k G^2}{2} + \sum_{k=1}^{K} \langle \xi^k, x^k - x^* \rangle.$$

Check that this is exactly inequality (3.5) plus all the accumulated noise.

Now we apply Azuma-Hoeffding inequality for $X_k = \langle \xi^k, x^k - x^* \rangle$, $k = 1, \ldots, K$. Notice that for (X_k) we have

$$|X_k| \leqslant \|\widehat{\nabla} f(x^k) - g^k\|_2 \|x^k - x^*\|_2 \leqslant 2GR \qquad \text{and} \qquad \mathbf{E}[X_k \mid X_1, \ldots, X_{k-1}] = 0.$$

Hence, for all $t \geqslant 0$,

$$\mathbf{P}\left[\sum_{k=1}^{K} \langle \xi^k, x^k - x^* \rangle \geqslant t\right] \leqslant \exp\left(-\frac{t^2}{2KG^2R^2}\right).$$

Or using $t = GR\sqrt{K}s$,

$$\mathbf{P}\left[\frac{1}{K}\sum_{k=1}^{K} \langle \xi^k, x^k - x^* \rangle \geqslant \frac{sGR}{\sqrt{K}}\right] \leqslant \exp\left(-\frac{s^2}{2}\right).$$

Then we conclude that with probability $1-e^{-\frac{s^2}{2}}$, the expression $\frac{1}{K}\sum_{k=1}^{K}\langle \xi^k, x^k - x^*\rangle$ is less than $\frac{sGR}{\sqrt{K}}$ and the Jensen inequality completes the proof.

■

5.2 Complexity

Although we mostly study algorithms from the perspective of convergence rate, in practice the latter is not of utmost importance. We are more concerned with the time, or energy, or money we spend to solve the problem. Therefore, we have also to consider the iteration cost. One can measure iteration cost in different ways: arithmetic operations, matrix-vector multiplications, etc., but for the algorithms we study, the most natural thing would be the evaluation cost of a (sub)gradient. We assume that all other operations are negligible in comparison.

To fix the setting, consider the problem $f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$. We use $\widehat{\nabla} f_i$ to denote either a gradient or a subgradient of f_i (depending on assumptions). Since we want to compare GD and SGD, we will use the cost of the individual (sub)gradients $\widehat{\nabla} f_i$. We suppose that all $f_1, \ldots, f_n$ incur similar costs. Hence, evaluating one $\widehat{\nabla} f$ costs n (sub)gradients $\widehat{\nabla} f_i$ and in the convex setting we have the following complexity table.

Algorithms	**Convergence rate**	**Iteration cost**	**Complexity**
Gradient descent	$O\left(\frac{1}{\varepsilon}\right)$	$n \times C(\nabla f_i)$	$O\left(\frac{n}{\varepsilon}\right) \times C(\nabla f_i)$
Subgradient method	$O\left(\frac{1}{\varepsilon^2}\right)$	$n \times C(\widehat{\nabla} f_i)$	$O\left(\frac{n}{\varepsilon^2}\right) \times C(\widehat{\nabla} f_i)$
Stochastic (sub)gradient method	$O\left(\frac{1}{\varepsilon^2}\right)$	$C(\widehat{\nabla} f_i)$	$O\left(\frac{1}{\varepsilon^2}\right) \times C(\widehat{\nabla} f_i)$

Table 5.1: Complexity results of different algorithms to find x with $f(x) - f_* \leqslant \varepsilon$ for $f = \frac{1}{n}\sum f_i$, convex case

The key message is that the complexity of SGD doesn't suffer from the factor n. Thus, despite its "slowness" it can be much more efficient in practice than the (sub)gradient method. For machine learning application the size of dataset, n, can be prohibitively large. And at the same time it is often sufficient to find a low-accuracy solution.

5.3 Nonconvex case

Let's see how to analyze SGD beyond the convex case. This time we assume L-smoothness of f, but won't assume its convexity.

Theorem 5.4 Let f be L-smooth and $f(x) \geqslant f_{\text{low}}$, and $\mathbf{E}_\xi\left[\|\nabla f_\xi(x) - \nabla f(x)\|^2\right] \leqslant \sigma^2$. Then SGD with a stepsize $\alpha = \sqrt{\frac{f(x^1)-f_{\text{low}}}{\sigma^2 LK}}$ satisfies

$$\min_{1\leqslant k\leqslant K} \mathbf{E}\left[\|\nabla f(x^k)\|^2\right] \leqslant 3\sqrt{\frac{(f(x^1)-f_{\text{low}})L\sigma^2}{K}}.$$

Proof. Since f is L-smooth, we can use descent lemma (Lemma 1.1) to get

$$f(x^{k+1}) - f(x^k) - \langle \nabla f(x^k), x^{k+1} - x^k\rangle \leqslant \frac{L}{2}\|x^{k+1} - x^k\|^2,$$

which in turn is equivalent to

$$f(x^{k+1}) - f(x^k) + \alpha\langle \nabla f(x^k), \nabla f_{i_k}(x^k)\rangle \leqslant \frac{\alpha^2 L}{2}\|\nabla f_{i_k}(x^k)\|^2.$$

Taking expectation $\mathbf{E}_k$ and using that $\mathbf{E}_k\left[\|\nabla f_{i_k}(x^k)\|^2\right] \leqslant \sigma^2 + \|\nabla f(x^k)\|^2$, we obtain

$$\mathbf{E}_k\left[f(x^{k+1})\right] - f(x^k) + \frac{\alpha(2-\alpha L)}{2}\|\nabla f(x^k)\|^2 \leqslant \frac{\alpha^2\sigma L}{2}.$$

Now we take total expectation $\mathbf{E}$ and regroup some terms

$$\frac{\alpha(2-\alpha L)}{2}\mathbf{E}\left[\|\nabla f(x^k)\|^2\right] \leqslant \mathbf{E}\left[f(x^k) - f(x^{k+1})\right] + \frac{\alpha^2\sigma L}{2}.$$

Summing this inequality over $k = 1, \ldots, K$ yields

$$\frac{\alpha(2-\alpha L)}{2}\sum_{k=1}^{K}\mathbf{E}\left[\|\nabla f(x^k)\|^2\right] \leqslant \mathbf{E}\left[f(x^1) - f(x^{K+1})\right] + \frac{\alpha^2\sigma LK}{2} \leqslant f(x^1) - f_{\text{low}} + \frac{\alpha^2\sigma LK}{2}. \tag{5.3}$$

With $\alpha = \sqrt{\frac{f(x^1)-f_{\text{low}}}{\sigma^2 LK}}$ we may assume without loss of generality that $1 - \alpha L \geqslant 0$. Then $\alpha(2-\alpha L) \geqslant \alpha$ and we derive

$$\frac{1}{K}\sum_{k=1}^{K}\mathbf{E}\left[\|\nabla f(x^k)\|^2\right] \leqslant 3\sqrt{\frac{(f(x^1)-f_{\text{low}})L\sigma^2}{K}},$$

from which the conclusion follows. ■

The statement of the theorem may appear unclear when $\sigma = 0$ due to the specific stepsize α we've employed. However, the proof remains valid until inequality (5.3) (including), before substituting a particular value for α. Thus, beyond just examining SGD in the nonconvex scenario, we've also gained insight how to analyze GD in this context.

Exercise 5.1 Derive a proper convergence rate for GD in the nonconvex case assuming that f is L-smooth and lower bounded $f(x) \geqslant f_{\text{low}}$ for all x. ■

5.4 Strongly convex case

So far we have mentioned strong convexity a few times, but never dicussed it in details.

Definition 5.1 — Strong convexity. A function $f : \mathbb{R}^d \to (-\infty, +\infty]$ is μ-strongly convex if

$$\lambda f(x) + (1-\lambda)f(y) \geqslant f(\lambda x + (1-\lambda)y) + \frac{\mu\lambda(1-\lambda)}{2}\|x-y\|_2^2.$$

Usually, when we consider μ-strongly convex functions, we implicitly assume that $\mu > 0$. For us a much more useful will be the following characterization.

Proposition 5.1 Let f be a convex function and $\mu > 0$. The following are equivalent:

1. f is μ-strongly convex.
2. $f(y) - f(x) - \langle g_x, y - x\rangle \geqslant \frac{\mu}{2}\|y-x\|_2^2$ for all x, y and $g_x \in \partial f(x)$.
3. $\langle g_x - g_y, x - y\rangle \geqslant \mu\|x-y\|_2^2$ for all x, y and $g_x \in \partial f(x)$, $g_y \in \partial f(y)$.

Naturally, if f is μ-strongly convex, we should expect a better rate for the standard algorithms we study. We illustrate this in the case of the projected SGD.

Theorem 5.5 Let C be a closed and convex set, f be a μ-strongly convex function, and $\mathbf{E}_k\left[\|g^k\|_2^2\right] \leqslant G^2$. Then the projected SGD (5.1) with $\alpha_k = \frac{2}{\mu(k+1)}$ satisfies

$$\mathbf{E}\left[f(x^{\text{best}}) - f_*\right] \leqslant \frac{2G^2}{\mu(K+1)}.$$

where $x^{\text{best}} = \operatorname{argmin}_{1\leqslant k\leqslant K} f(x^k)$.

Proof. We start as usual

$$\begin{aligned}\|x^{k+1} - x^*\|_2^2 &= \|P_C(x^k - \alpha_k g^k) - P_C x^*\|_2^2 \\ &\leqslant \|x^k - x^* - \alpha_k g^k\|_2^2 \\ &= \|x^k - x^*\|_2^2 - 2\alpha_k\langle g^k, x^k - x^*\rangle + \alpha_k^2\|g^k\|_2^2.\end{aligned}$$

Let $\widehat{\nabla} f(x^k) = \mathbf{E}_k\left[g^k\right]$. Taking conditional expectation $\mathbf{E}_k$ from both sides, gives us

$$\begin{aligned}\mathbf{E}_k\left[\|x^{k+1} - x^*\|_2^2\right] &= \|x^k - x^*\|_2^2 - 2\alpha_k\langle\widehat{\nabla} f(x^k), x^k - x^*\rangle + \alpha_k^2\,\mathbf{E}_k\left[\|g^k\|_2^2\right] \\ &\leqslant \|x^k - x^*\|_2^2 - 2\alpha_k(f(x^k) - f_* + \frac{\mu}{2}\|x^k - x^*\|_2^2) + \alpha_k^2 G^2 \\ &= (1 - \alpha_k\mu)\|x^k - x^*\|_2^2 - 2\alpha_k(f(x^k) - f_*) + \alpha_k^2 G^2\end{aligned}$$

where in the last inequality we used strong convexity and $\mathbf{E}_k\left[\|g^k\|_2^2\right] \leqslant G^2$. Dividing by $2\alpha_k$ and rearranging the terms, we obtain

$$f(x^k) - f_* \leqslant \frac{1}{2}\left(\frac{1}{\alpha_k} - \mu\right)\|x^k - x^*\|_2^2 - \frac{1}{2\alpha_k}\,\mathbf{E}_k\left[\|x^{k+1} - x^*\|_2^2\right] + \frac{\alpha_k G^2}{2}.$$

We can substitute $\alpha_k = \frac{2}{\mu(k+1)}$ to get

$$f(x^k) - f_* \leqslant \frac{\mu(k-1)}{4}\|x^k - x^*\|_2^2 - \frac{\mu(k+1)}{4}\,\mathbf{E}_k\left[\|x^{k+1} - x^*\|_2^2\right] + \frac{G^2}{\mu(k+1)}.$$

Now we do something that seems strange at first: we multiply both sides by k

$$k(f(x^k) - f_*) \leqslant \frac{\mu(k-1)k}{4}\|x^k - x^*\|_2^2 - \frac{\mu k(k+1)}{4}\,\mathbf{E}_k\left[\|x^{k+1} - x^*\|_2^2\right] + \frac{kG^2}{\mu(k+1)}.$$

This allows us to nicely telescope the right-hand side after taking total expectation. This yields

$$\sum_{k=1}^{K} k\mathbf{E}\left[(f(x^k) - f_*)\right] \leqslant 0 - \frac{\mu K(K+1)}{4}\mathbf{E}\left[\|x^{K+1} - x^*\|_2^2\right] + \frac{G^2}{\mu}\sum_{k=1}^{K}\frac{k}{k+1} \leqslant \frac{KG^2}{\mu}.$$

Now we conclude in the usual way using so-called "best iterate":

$$\mathbf{E}\left[f(x^k) - f_*\right] \geqslant \mathbf{E}\left[\min_{1\leqslant i\leqslant K} f(x^i) - f_*\right] = \mathbf{E}\left[f(x^{\text{best}}) - f_*\right].$$

Since $\sum_{k=1}^{K} k = \frac{K(K+1)}{2}$, we conclude

$$\frac{K(K+1)}{2}\mathbf{E}\left[f(x^{\text{best}}) - f_*\right] \leqslant \frac{KG^2}{\mu}$$

and the final bound follows immediately. Clearly, instead of using "best iterate", we could use Jensen's inequality to derive the same bound for a certain weighted average sequence. ■

Comments

The analysis of the projected SGD including its probabilistic guarantees is from [2]. The analysis of the strongly convex case is from [1].

In the convex but differentiable case the assumptions

$$\mathbf{E}\left[\|\nabla f_\xi(x)\|^2\right] \leqslant G^2 \quad \text{or} \quad \mathbf{E}\left[\|\nabla f_\xi(x) - \nabla f(x)\|^2\right] \leqslant \sigma^2$$

are both restrictive, since they essentially require the sequence (x^k) to be bounded, which is a notably strong assumption. There are more refined analyses of SGD that rely on more realistic assumptions, such as

$$\mathbf{E}\left[\|\nabla f_\xi(x)\|^2\right] \leqslant A(f(x) - f_*) + B,$$

for some constants $A, B > 0$. As an exercise, check that the finite sum problem $f = \frac{1}{n}\sum f_i$ where each f_i is L-smooth, satisfies the above property.

References

[1] Amir Beck. *First-order methods in optimization*. SIAM, 2017.

[2] John C Duchi. "Introductory lectures on stochastic optimization". In: *The mathematics of data* 25 (2018), pages 99–186. URL: https://web.stanford.edu/~jduchi/PCMIConvex/Duchi16.pdf.

Lecture 6: Coordinate gradient method

We use e_j to denote the j-th basis vector $(0,\dots,1,\dots,0)\in\mathbb{R}^d$.

Recall that f is L-smooth if and only if its gradient ∇f is L-Lipschitz continuous. This time we give a more refined definition.

Definition 6.1 We call $f:\mathbb{R}^d\to\mathbb{R}$ *coordinate-wise* $(L_1,\dots,L_d)$*-smooth*, if for each $i\in[d]$,

$$|\nabla_i f(x+te_i)-\nabla_i f(x)|\leqslant L_i|t|,\qquad \forall x\in\mathbb{R}^d,\forall t\in\mathbb{R}.$$

R Whenever we discuss smoothness of f, we always consider the smallest possible constants L_i and L.

It is easy to see that coordinate-wise smoothness is equivalent to L_i-smoothness of the function $t\mapsto f(x+te_i)$ for every $i\in[d]$ and any $x\in\mathbb{R}^d$. If f is twice differentiable, then it is also equivalent to saying that $[\nabla^2 f(x)]_{ii}\leqslant L_i$, or, in terms of matrices, to

$$\operatorname{diag}(\nabla^2 f(x))\preccurlyeq\operatorname{diag}(L_1,\dots,L_d).$$

Exercise 6.1 Check the last statement. ∎

Exercise 6.2 Prove that if f is coordinate-wise $(L_1,\dots,L_d)$-smooth then it is also L-smooth for some $L>0$. Moreover, the following must hold

$$\max_i L_i\leqslant L\leqslant\sum_{i=1}^d L_i.$$

∎

Instead of GD

$$x^{k+1}=x^k-\alpha\nabla f(x^k),$$

one can update only one coordinate at a time: say we update only the j-th coordinate and keep the rest the same

$$x_j^{k+1}=x_j^k-\alpha\nabla_j f(x^k) \tag{6.1}$$

and we keep $x_i^{k+1}=x_i^k$ for $i\neq j$. Alternatively, we can write it as

$$x^{k+1}=x^k-\alpha\nabla_j f(x^k)e_j, \tag{6.2}$$

where e_j is the j-th basis vector.

This is still not a well-defined method — we need to determine some strategy how to select j in every iteration. This will be discussed a bit later. The main question is why to even consider such a method?

Potential advantages:

- cheap update. In many situations, computing

$$\nabla_j f(x) = \frac{\partial f(x)}{\partial x_j}$$

  can be d times cheaper than computing $\nabla f(x)$.
- larger stepsize: perhaps, instead of $\alpha < \frac{2}{L}$ we can use $\alpha_j < \frac{2}{L_j}$.

Potential disadvantages:

- slow convergence. Since we only update one coordinate at a time, it is obvious that this update is worse than GD's.

For a long time coordinate gradient methods haven't been popular, as their convergence rate was worse than GD's. This has changed in 2010 when Nesterov suggested a certain random strategy how to select index j.

6.1 Examples

Separable function. A function f is called *separable* if $f(x) = \sum_{i=1}^{d} f_i(x_i)$. For this type of function, basic GD isn't a good method in general. Consider $f(x_1, x_2) = x_1^2 + 10^6 x_2^2$. For this function $L = 2 \cdot 10^6$ (check this!) and hence we have to use a very small stepsize. If we start from $x^0 = (1, 1)$, we will need to perform too many iterations to be close to the solution $x^* = (0, 0)$.

In general, of course, you should never use a black-box GD for such functions. Instead you should solve each $\min f_i(x)$ individually.

Pairwise-separable functions. Let

$$f(x) = \sum_{i,j=1}^{d} f_{ij}(x_i, x_j).$$

This function is a prototypical example of the following situation. Consider a complete graph G with vertices $\{1, \dots, d\}$. Each edge (i, j) defines some function f_{ij} that depends on the vertices x_i, x_j of this edge. Our goal is to find vertices weights $(x_1, \dots, x_d)$ with the smallest total energy f. In a similar way, we can define the problem for non-complete graphs.

In machine learning, such functions are also used for label propagation tasks. It's used when we have a partially labeled dataset and want to propagate labels throughout the dataset by considering the labels of neighboring data points.

To compute gradient $\nabla f(x)$, we obviously need to compute gradients of each term in the double sum. This costs $O(d^2)$. On the other hand, $\nabla_j f(x)$ only costs $O(d)$.

Empirical risk minimization. These are the problems where typical machine learning training takes place. Consider f defined by

$$\min_{x \in \mathbb{R}^d} f(x) := \sum_{i \in [n]} \varphi_i(a_i^\top x) + g(x),$$

where $a_i \in \mathbb{R}^d$ are the rows of the $n \times d$ matrix A and φ_i, g are convex. Here a_i are our data points, φ_i are loss functions and g is a regularizer.

The dual problem[1] is

$$\min_{y\in\mathbb{R}^n} \sum_{i\in[n]} \varphi_i^*(y_i) + g^*(-A^\top y),$$

which can often be tackled by coordinate methods (since the objective is almost separable).

6.2 Analysis

Lemma 6.1 Coordinate-wise descent lemma. If f is coordinate-wise $(L_1,\dots,L_d)$-smooth, then

$$f(x+te_i) \leqslant f(x) + \langle \nabla_i f(x), t\rangle + \frac{L_i}{2}|t|^2, \qquad \forall x\in\mathbb{R}^d, \forall t\in\mathbb{R}. \tag{6.3}$$

(We write inner product $\langle \nabla_i f(x), t\rangle$ just to resemble similarity with classical descent lemma. Of course, now it is just a product of two real numbers.)

Proof. By definition, coordinate-wise smoothness is equivalent to L_i-smoothness of $t\mapsto f(x+te_i)$ for every $i\in[d]$. Let $g_i(t) = f(x+te_i)$. Then $g_i'(t) = \langle \nabla f(x+te_i), e_i\rangle = \nabla_i f(x+te_i)$. Since g_i is L_i-smooth, we can apply descent lemma to g at t and 0 to get

$$g_i(t) \leqslant g_i(0) + g_i'(0)\cdot t + \frac{L_i}{2}|t|^2$$

■

Consider coordinate descent with stepsize $\alpha = \frac{1}{L_j}$, that is

$$x^{k+1} = x^k - \frac{1}{L_j}\nabla_j f(x^k) e_j. \tag{6.4}$$

Lemma 6.2 Suppose $f:\mathbb{R}^d\to\mathbb{R}$ is differentiable and coordinate-wise $(L_1,\dots,L_d)$-smooth. Then one iteration of coordinate descent (6.4) satisfies

$$f(x^{k+1}) \leqslant f(x^k) - \frac{1}{2L_j}|\nabla_j f(x^k)|^2. \tag{6.5}$$

Proof. This follows directly from (6.3). ■

6.2.1 Choice of the index j

Cyclic choice. This is probably the simplest choice: we select $j = k \pmod d + 1$. In other words, we update each coordinate cyclically. Unfortunately, for such a choice it is not possible to get easy guarantees, although it may perform well on some particular instances.

Greedy rule. It is natural to want to decrease f as much as possible in every iteration. Considering (6.5), we may choose

$$j = \operatorname*{argmax}_{i\in[d]} \left\{ \frac{|\nabla_i f(x^k)|^2}{L_i} \right\}.$$

This is called a greedy or Gauss-Southwell's rule. In most cases this approach is not practical, since this requires us to compute a full gradient $\nabla f(x^k)$. However, for certain particular problems it can be made efficient.

[1] Skip it if you don't know what the dual problem is.

Uniform random choice. Let us choose j uniformly at random

$$\begin{aligned} j &\sim \text{Unif}\{1,\dots,d\} \\ x^{k+1} &= x^k - \frac{1}{L}\nabla_j f(x^k) e_j. \end{aligned} \tag{6.6}$$

Here we used stepsize $\alpha = \frac{1}{L}$ and not $\alpha = \frac{1}{L}$, since for the latter step we cannot derive a good rate.

If we take the conditional expectation of both sides in (6.5), we obtain

$$\begin{aligned} \mathbf{E}_j\left[f(x^{k+1})\right] &\leqslant f(x^k) - \frac{1}{2}\sum_{i\in[d]} \frac{p_i}{L}|\nabla_i f(x^k)|^2 \\ &= f(x^k) - \frac{1}{2dL}\|\nabla f(x^k)\|^2. \end{aligned}$$

Comparing this inequality with (1.4) (one step of GD), we see that they are of the same nature, but the former has a much worse constant — dL instead of L. Thus, if we continue with the analysis, the final constant will end up being d times larger than that of GD. Since, the cost of each iteration is at best d times cheaper, we conclude that this approach won't bring any benefits compared to GD.

Random importance choice. It is reasonable to sample more often those coordinates whose partial Lipschitz constants are large. This leads to the random importance sampling

$$\begin{aligned} &\text{Sample } j \text{ with } \mathbf{P}[j=i] = \frac{L_i}{L_1+\dots+L_d} \\ &x^{k+1} = x^k - \frac{1}{L_j}\nabla_j f(x^k) e_j. \end{aligned} \tag{6.7}$$

Lemma 6.3 Let f be coordinate-wise $(L_1,\dots,L_d)$-smooth and let $\bar{L} = \frac{1}{d}\sum_{i\in[d]} L_i$. Then for the scheme (6.7) it holds

$$\mathbf{E}_j\left[f(x^{k+1})\right] \leqslant f(x^k) - \frac{1}{2d\bar{L}}\|\nabla f(x^k)\|^2. \tag{6.8}$$

Proof. We just need to take expectation of both sides in (6.5) over j:

$$\begin{aligned} \mathbf{E}_j\left[f(x^{k+1})\right] &\leqslant f(x^k) - \frac{1}{2}\sum_{i\in[d]} \frac{p_i}{L_i}|\nabla_i f(x^k)|^2 \\ &= f(x^k) - \frac{1}{2\sum_{i\in[d]} L_i}\|\nabla f(x^k)\|^2 \\ &= f(x^k) - \frac{1}{2d\bar{L}}\|\nabla f(x^k)\|^2. \end{aligned}$$

■

Theorem 6.1 Let f be convex and coordinate-wise $(L_1,\dots,L_d)$-smooth. Suppose that

$$\mathscr{L}(x^0) = \left\{x \in \mathbb{R}^d : f(x) \leqslant f(x^0)\right\} \text{ is compact and } R = \operatorname{diam}(\mathscr{L}(x^0)).$$

Then
$$\mathbf{E}\left[f(x^k)-f^*\right] \leqslant \frac{2d\bar{L}R^2}{k}.$$

Proof. Since f is convex, we have that for any $x \in \mathbb{R}^d$

$$f(x^k)-f^* \leqslant \langle \nabla f(x^k), x^k - x^* \rangle \leqslant \|\nabla f(x^k)\|\|x^k - x^*\| \leqslant R\|\nabla f(x^k)\|,$$

where we used that $x^k \in \mathscr{L}(x^0)$ by (6.5). Hence,

$$\mathbf{E}_j\left[f(x^{k+1})-f^*\right] \leqslant f(x^k)-f^* - \frac{1}{2d\bar{L}R^2}(f(x^k)-f^*)^2.$$

Taking total expecation and denoting $a_k = \mathbf{E}\left[f(x^k)-f^*\right]$, $c = \frac{1}{2d\bar{L}R^2}$, and $b_k = ca_k$, we deduce

$$b_{k+1} \leqslant b_k - b_k^2.$$

This yields
$$\frac{1}{b_{k+1}} - \frac{1}{b_k} = \frac{b_k - b_{k+1}}{b_k b_{k+1}} \geqslant \frac{b_k^2}{b_k b_{k+1}} \geqslant 1.$$
Summing up first k inequalities of this type, we get

$$\frac{1}{b_k} - \frac{1}{b_0} \geqslant k \implies \frac{1}{b_k} \geqslant k \implies b_k \leqslant \frac{1}{k},$$

and the result follows. ■

Exercise 6.3 What if we didn't drop out the term $\frac{1}{b_0}$ at the end of the proof? How would the final bound look like? ▪

Exercise 6.4 Notice that in the same way one could analyze GD. How would the final bound look like in this case? ▪

Strong convexity. In the case of strong convexity, the analysis is even simpler.

Theorem 6.2 Let f be μ-strongly convex and coordinate-wise $(L_1, \dots, L_d)$-smooth, then

$$\mathbf{E}\left[f(x^k)-f^*\right] \leqslant \left(1 - \frac{\mu}{d\bar{L}}\right)^k (f(x^0)-f^*). \tag{6.9}$$

Proof. By strong convexity, we have that $\|\nabla f(x)\|^2 \geqslant 2\mu(f(x)-f^*)$ for all x. Applying it in (6.8) yields

$$\mathbf{E}_j\left[f(x^{k+1})\right] \leqslant f(x^k) - \frac{\mu}{d\bar{L}}(f(x^k)-f^*). \tag{6.10}$$

By taking the total expectation, we get

$$\mathbf{E}\left[f(x^{k+1})-f^*\right] \leqslant \left(1 - \frac{\mu}{d\bar{L}}\right)(f(x^k)-f^*),$$

and the proof follows by iterating the inequality above. ■

This rate is again strictly worse than that of gradient descent. However, if the algorithm update is much cheaper, it may be still advantageous.

Exercise 6.5 Prove that if f is μ-strongly convex and differentiable, then

$$\|\nabla f(x)\|^2 \geqslant 2\mu(f(x) - f^*).$$

■

Comments

1. We can improve the convergence rate of coordinate methods by incorporating acceleration. Roughly speaking, the acceleration idea works the same way as for the basic GD, but for the coordinate variant we have to be more careful not to use full vector-vector operations. For more details see [3, 4] and the review paper [6].
2. Instead of coordinates, we could also use block of coordinates. This is more or less should be straightforward.
3. One can also analyze the cyclic version of coordinate method, see paper [2] or the exposition in [7, Chapter 6], also a more general version in [1, Chapter 11].
4. Analysis of the greedy rule can be found in [5].

References

[1] Amir Beck. *First-order methods in optimization*. SIAM, 2017.

[2] Amir Beck and Luba Tetruashvili. "On the convergence of block coordinate descent type methods". In: *SIAM Journal on Optimization* 23.4 (2013), pages 2037–2060.

[3] Yin Tat Lee and Aaron Sidford. "Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems". In: *2013 IEEE 54th annual symposium on foundations of computer science*. IEEE. 2013, pages 147–156.

[4] Yurii Nesterov and Sebastian U Stich. "Efficiency of the accelerated coordinate descent method on structured optimization problems". In: *SIAM Journal on Optimization* 27.1 (2017), pages 110–123.

[5] Julie Nutini et al. "Coordinate descent converges faster with the Gauss-Southwell rule than random selection". In: *International Conference on Machine Learning*. PMLR. 2015, pages 1632–1641.

[6] Stephen J Wright. "Coordinate descent algorithms". In: *Mathematical programming* 151.1 (2015), pages 3–34.

[7] Stephen J Wright and Benjamin Recht. *Optimization for data analysis*. Cambridge University Press, 2022.

Lecture 7: Randomized Kaczmarz method

7.1 Kaczmarz Algorithm

We want to solve a linear system $Ax = b$ for given $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. We assume that this system is feasible (that is it has a solution). Let S denote the set of all solutions and $a_1, \dots, a_m$ denote the rows of A. Then the linear system can be written as

$$\langle a_i, x\rangle = b_i, \quad i = 1, \dots, m.$$

SGD. If the linear system $Ax = b$ has a solution, then we can equivalently solve the least squares problem

$$\min_x F(x) := \frac{1}{2m}\|Ax - b\|^2 = \frac{1}{2m}\sum_{i=1}^{m}(\langle a_i, x\rangle - b_i)^2,$$

where we put factor $\frac{1}{2m}$ for mere convenience. Let $F_i(x) = \frac{1}{2}(\langle a_i, x\rangle - b_i)^2$. The problem above is a finite sum minimization. And hence, we can apply SGD:

$$\begin{aligned} x^{k+1} &= x^k - \alpha_k \nabla F_j(x^k) \\ &= x^k - \alpha_k(\langle a_j, x^k\rangle - b_j)a_j, \end{aligned} \tag{7.1}$$

where j is sampled uniformly at random and $\alpha_k > 0$ is a stepsize.

Coordinate descent for dual. The system $Ax = b$ may have many solutions (actually infinitely many). It makes sense to seek the least-norm solution, which is given by

$$\min_x \frac{1}{2}\|x\|^2 \quad \text{s.t.} \quad Ax = b. \tag{7.2}$$

This is a constrained optimization problem. Let's instead consider its dual problem:

$$\min_{x\in\mathbb{R}^n}\max_{y\in\mathbb{R}^m} \frac{1}{2}\|x\|^2 + \langle Ax - b, y\rangle = \max_{y\in\mathbb{R}^m}\min_{x\in\mathbb{R}^n} \frac{1}{2}\|x\|^2 + \langle Ax - b, y\rangle = \max_{y\in\mathbb{R}^m} -\frac{1}{2}\|A^\top y\|^2 - \langle b, y\rangle$$

Thus, now we can solve instead

$$\min_y f(y) := \frac{1}{2}\|A^\top y\|^2 + \langle b, y\rangle$$

and from a solution y^* of this problem we will be able to recover the primal solution $x^* = -A^\top y^*$.

We have $\nabla f(y) = AA^\top y + b$ and thus $\nabla_i f(y) = \langle a_i, A^\top y\rangle + b_i$, where $a_i \in \mathbb{R}^n$ is the i-th row of A. Hence, the coordinate descent for $\min_y f(y)$ is

$$y^{k+1} = y^k - \alpha_k(\langle a_j, A^\top y^k\rangle + b_j)\cdot e_j,$$

where $\alpha_k > 0$ is some stepsize. Now we use the standard trick. We don't want to compute $A^\top y^k$ in every iteration. If we multiply both sides by $-A^\top$ and use the change of variables $x^k := -A^\top y^k$, then

$$\begin{aligned} x^{k+1} &= x^k + \alpha_k(\langle a_j, A^\top y^k\rangle + b_j)\cdot A^\top e_j \\ &= = x^k - \alpha_k(\langle a_j, x^k\rangle - b_j)a_j. \end{aligned}$$

What we see now it that this is exactly the same update as we got by SGD in (7.1). This already implies that we don't need to use a small stepsize for this method as SGD requires.

7.1.1 Kaczmarz algorithm

Stefan Kaczmarz in 1937 suggested [1] the following purely geometric method. Take any $x^0 \in \mathbb{R}^n$ and project it onto the hyperplane defined by the first row $\langle a_i, x\rangle = b_i$. In other words,

$$x^1 = \operatorname*{argmin}_{x\,:\,\langle a_1, x\rangle = b_1} \|x - x^0\|^2,$$

which can be computed explicitly as

$$x^1 = x^0 + \frac{b_1 - \langle a_1, x^0\rangle}{\|a_1\|^2} a_1. \tag{7.3}$$

Now project x^1 onto the hyperplane defined by the second row and so on. This method later was generalize to arbitrary convex closed sets and known as the *cyclic projection* method. One can prove that in the case of a linear system this method linearly converges to a solution, but it is not clear how to obtain convergence rate explicitly. Actually, Kaczmarz in his paper didn't even prove its convergence but only noticed "The convergence of this method is obvious from a geometric perspective"[1].

Looking once more at the update (7.3), we see that it is the same update as described earlier. The only difference is that the original Kaczmarz method selects j in a cyclic order. We will see that the analysis of the method becomes much easier when we introduce some randomness.

7.1.2 Randomized Kaczmarz algorithm

We consider the same update

$$\boxed{x^{k+1} = x^k - \frac{\langle a_j, x^k\rangle - b_j}{\|a_j\|^2} a_j,} \tag{7.4}$$

but this time we chose the j-th row is a certain randomized way. We consider two main strategies.

[1] "Die Konvergenz des Verfahrens ist geometrisch ohne weiteres einleuchtend"

Uniform sampling. Let us select each j uniformly with probability $\frac{1}{m}$. Choose any solution x^* and note that $x^k - x^{k+1} \perp x^{k+1} - x^*$. Taking the conditional expectation with respect to j, we obtain

$$\begin{aligned}
\mathbf{E}_k\left[\|x^{k+1}-x^*\|^2\right] &= \|x^k - x^*\|^2 - \mathbf{E}_k\left[\|x^{k+1}-x^k\|^2\right] && \text{Pythagoras' theorem}\\
&= \|x^k - x^*\|^2 - \mathbf{E}_k\left[\frac{|\langle a_j, x^k\rangle - b_j|^2}{\|a_j\|^2}\right] \\
&= \|x^k - x^*\|^2 - \sum_{i=1}^m \frac{1}{m}\frac{|\langle a_i, x^k\rangle - b_i|^2}{\|a_i\|^2} && \text{definition of expectation}\\
&\leqslant \|x^k - x^*\|^2 - \frac{1}{m\|A\|_{2,\infty}^2}\|Ax^k - b\|^2 && \|A\|_{2,\infty}^2 := \max_i \|a_i\|^2\\
&= \|x^k - x^*\|^2 - \frac{1}{m\|A\|_{2,\infty}^2}\|A(x^k - x^*)\|^2 && Ax^* = b.
\end{aligned}$$

In general, $\|A\|_{p,q}$ defines the induced operator norm as $\|A\|_{p,q} := \max_{x\neq 0} \frac{\|Ax\|_q}{\|x\|_p}$.

Exercise 7.1 Using the definition above, show that $\|A\|_{2,\infty} = \max_i \|a_i\|$. ∎

It would be nice to get a bound like $\|A(x^k - x^*)\|^2 \geqslant c\|x^k - x^*\|^2$, which is equivalent to

$$\langle A^\top A(x^k - x^*), x^k - x^*\rangle \geqslant c\|x^k - x^*\|^2. \tag{7.5}$$

The matrix $A^\top A$ is symmetric positive semidefinite with eigenvalues $0 \leqslant \lambda_1 \leqslant \ldots \leqslant \lambda_n$. If $\lambda_1 > 0$, that is $A^\top A$ was positive definite, then, we could obviously take $c = \lambda_1$. In general, this is not true. But if we choose $x^* \in S$ in a special way, in particular as

$$x^* = P_S x^k = \operatorname*{argmin}_{x\,:\,Ax=b} \|x - x^k\|^2,$$

then it must hold that $x^k - x^* \perp \ker(A)$. Hence, in fact (7.5) holds with $c = \lambda_+$, where λ_+ is the minimal nonzero eigenvalue of $A^\top A$. Note that for this particular x^*, it holds that $\|x^k - x^*\| = \operatorname{dist}(x^k, S)$ by definition.

Thus, we can continue

$$\begin{aligned}
\mathbf{E}_k\left[\|x^{k+1}-x^*\|^2\right] &\leqslant \|x^k - x^*\|^2 - \frac{\lambda_+}{m\|A\|_{\infty,2}^2}\|x^k - x^*\|^2\\
&= \left(1 - \frac{\lambda_+}{m\|A\|_{\infty,2}^2}\right)\operatorname{dist}(x^k, S)^2.
\end{aligned}$$

Note that we can lower bound the left-hand side by $\|x^{k+1} - x^*\|^2 \geqslant \operatorname{dist}(x^{k+1}, S)^2$. Therefore,

$$\mathbf{E}_k\left[\operatorname{dist}(x^{k+1}, S)^2\right] \leqslant \left(1 - \frac{\lambda_+}{m\|A\|_{\infty,2}^2}\right)\operatorname{dist}(x^k, S)^2. \tag{7.6}$$

As a sanity check, let's verify that this rate is meaningful, that is

$$0 \leqslant 1 - \frac{\lambda_+}{m\|A\|_{\infty,2}^2} < 1.$$

The right inequality is clear, since $\lambda_+ > 0$ by definition. The left one is equivalent to

$$\lambda_+(A^\top A) \leqslant m\|A\|_{\infty,2}^2.$$

Indeed, it is true by

$$\lambda_+(A^\top A) \leqslant \sum_{i=1}^{n} \lambda_i(A^\top A) = \operatorname{tr}(A^\top A) = \|A\|_F^2 = \sum_{i=1}^{m} \|a_i\|^2 \leqslant m\|A\|_{\infty,2}^2. \tag{7.7}$$

Exercise 7.2 Prove that if $x^* = P_S x^k = \operatorname{argmin}_{x:\, Ax=b} \|x - x^k\|^2$, then $x^k - x^* \perp \ker(A)$. ∎

Finally, we can formulate the following result.

Theorem 7.1 Kaczmarz algorithm (7.4) with uniform sampling $\mathbf{P}[j = i] = \frac{1}{m}$ satisfies

$$\mathbf{E}\left[\|x^k - x^*\|^2\right] = \left(1 - \frac{\lambda_+}{m\|A\|_{\infty,2}^2}\right)^k \operatorname{dist}(x^0, S)^2,$$

where λ_+ is the smallest nonzero eigenvalue of $A^\top A$.

Non-uniform sampling. In the previous analysis we use a crude inequality $\|a_i\|^2 \leqslant \|A\|_{\infty,2}^2$. This is possible to avoid if we sample rows of A based on their "importance". That was an insightful idea of Strohmer & Vershynin [2]. In particular, we shall sample j with probability

$$\mathbf{P}[j = i] = \frac{\|a_i\|^2}{\|A\|_F^2}.$$

Borrowing the analysis from the previous case, we have

$$\begin{aligned}
\mathbf{E}_k\left[\|x^{k+1} - x^*\|^2\right] &= \|x^k - x^*\|^2 - \mathbf{E}\left[\frac{|\langle a_j, x^k\rangle - b_j|^2}{\|a_j\|^2}\right] \\
&= \|x^k - x^*\|^2 - \sum_{i=1}^{m} \frac{\|a_i\|^2}{\|A\|_F^2} \frac{|\langle a_i, x^k\rangle - b_i|^2}{\|a_i\|^2} && \text{definition of expectation} \\
&= \|x^k - x^*\|^2 - \frac{1}{\|A\|_F^2} \sum_{i=1}^{m} \|A(x^k - x^*)\|^2
\end{aligned}$$

Doing the same tricks as above, that is selecting $x^* = P_S x^k$, we deduce

$$\mathbf{E}_k\left[\operatorname{dist}(x^{k+1}, S)^2\right] \leqslant \left(1 - \frac{\lambda_+(A^\top A)}{\|A\|_F^2}\right) \operatorname{dist}(x^k, S)^2.$$

Inequality (7.7) immediately tells us that (i) this rate is meaningful and (ii) it is better than that in (7.6). We can summarize it as following.

Theorem 7.2 Kaczmarz algorithm (7.4) with selecting the j-th row as $\mathbf{P}[j = i] = \frac{\|a_i\|^2}{\|A\|_F^2}$ satisfies

$$\mathbf{E}\left[\|x^k - x^*\|^2\right] = \left(1 - \frac{\lambda_+}{\|A\|_F^2}\right)^k \operatorname{dist}(x^0, S)^2,$$

where λ_+ is the smallest nonzero eigenvalue of $A^\top A$.

Question: We have a linear system $Ax = b$ and we can scale its rows as we wish. Isn't it a bit strange that this "importance sampling" rule controls the convergence rate?

References

[1] Stefan Kaczmarz. “Angenäherte auflösung von systemen linearer gleichungen”. In: *Bulletin International de l’Académie Polonaise des Sciences et des Lettres. Classe des Sciences Mathématiques et Naturelles, Séria A* 35 (1937), pages 355–357.

[2] Thomas Strohmer and Roman Vershynin. “A randomized Kaczmarz algorithm with exponential convergence”. In: *Journal of Fourier Analysis and Applications* 15.2 (2009), pages 262–278.

Lecture 8: Variance reduction

Probability reminder. Let X, Y be (one-dimensional) random variables. The *variance* of X is defined as $\mathbf{Var}[X] = \mathbf{E}\left[(X - \mathbf{E}[X])^2\right]$. The *covariance* of X and Y is defined as

$$\mathbf{cov}(X, Y) = \mathbf{E}[(X - \mathbf{E}[X]) \cdot (Y - \mathbf{E}[Y])].$$

Evidently, $\mathbf{Var}[X] = \mathbf{cov}(X, X)$.

If X is a random vector in $\mathbb{R}^d$, the usual notion in probability is the *covariance matrix*

$$\mathbf{cov}(X, X) = \mathbf{E}\left[(X - \mathbf{E}[X])(X - \mathbf{E}[X])^\top\right] \in \mathbb{R}^{d \times d}.$$

However, sometimes in the probability literature this may be also called a variance of X. Unfortunately, this contradicts a common convention in optimization to call the variance a much simpler object

$$\mathbf{Var}[X] = \mathbf{E}\left[(X - \mathbf{E}[X])^\top (X - \mathbf{E}[X])\right] = \mathbf{E}\left[\|X - \mathbf{E}[X]\|^2\right] = \mathbf{E}\left[\|X\|^2\right] - \|\mathbf{E}[X]\|^2 \in \mathbb{R}. \quad (8.1)$$

One can see that in this context, $\mathbf{Var}[X]$ is just the trace of the covariance matrix $\mathbf{cov}(X, X)$. For this lecture, formula (8.1) is basically the only new thing we need to know in terms of probability theory.

8.1 GD vs SGD

We consider f in the finite sum form $f(x) = \frac{1}{n}\sum_{i=1}^n f_i(x)$ and assume that each f_i is L-smooth. For simplicity we also assume that f is μ-strongly convex function, although the same story holds for convex functions (just with different rates).

Algorithms	Convergence rate	Iteration cost	Complexity
Gradient descent	$O\left(\frac{L}{\mu}\log\frac{1}{\varepsilon}\right)$	n	$O\left(\frac{nL}{\mu}\log\frac{1}{\varepsilon}\right)$
Stochastic gradient method	$O\left(\frac{1}{\varepsilon}\right)$	1	$O\left(\frac{1}{\varepsilon}\right)$

Table 8.1: Complexity results of GD and SGD to find x with $f(x) - f_* \leqslant \varepsilon$ for $f = \frac{1}{n}\sum f_i$, strongly convex case. We used a simplified expression for SGD

In Table 8.1 one can see the convergence rate and the complexity (in the number of gradients ∇f_i) of GD and SGD. One can wonder if there is an optimization method with a convergence rate between GD and SGD.

Another reason why we want to do this is that so far, smoothness hasn't brought us any advantages in the SGD case; it just allowed us to use more realistic assumptions (see Comments in Lecture 5).

Consider the update

$$x^{k+1} = x^k - \alpha g^k \tag{8.2}$$

Many algorithms including GD and SGD can be put in the framework of (8.2).

We know that SGD is slow because it doesn't use a full gradient information, in other words it accumulates noise

$$g^k \neq \nabla f(x^k) \quad \Longrightarrow \quad \mathbf{Var}\left[g^k\right] = \mathbf{E}\left[\|g^k - \nabla f(x^k)\|^2\right] > 0.$$

Now consider (8.2) with a new estimator

$$g^k = \nabla f_{i_k}(x^k) - \nabla f_{i_k}(w) + \nabla f(w),$$

where w is a certain deterministic vector. First notice that this is an unbiased estimator, $\mathbf{E}\left[g^k\right] = \nabla f(x^k)$. What can we choose instead of w?

- $w = x^k \implies g^k = \nabla f(x^k)$. This is a very good estimator, but expensive.
- $w = x^{k-10} \implies g^k - \nabla f(x^k) = \left(\nabla f_{i_k}(x^k) - \nabla f_{i_k}(x^{k-10})\right) + \left(\nabla f(x^{k-10}) - \nabla f(x^k)\right)$. Suppose we proved convergence of the proposed algorithm (with a new estimator). Then (x^k) converges to a solution and $x^k - x^{k-10} \to 0$. Hence, by continuity of ∇f, we conclude that $g^k - \nabla f(x^k) \to 0$, that is $\mathbf{Var}\left[g^k\right]$ decreases as $k \to \infty$. Of course, there is nothing special in x^k, we could use x^{k+1}, x^{k+2} and so one — all with the same vector $w = x^{k-10}$. But we cannot use this w for ever, so from time to time we must also update w.

The idea is as follows:

we will infrequently compute the full gradient and use it to improve our usual SGD estimator.

In optimization this idea goes back to the SVRG algorithm [1], that appeared in 2008. In this lecture, however, we will consider another algorithm with the same complexity but with a slightly simpler analysis [2].

Algorithm 1 Loopless SVRG

Parameters: stepsize $\alpha > 0$, probability $p = \frac{1}{n}$
Initialization: $x^0 = w^0 \in \mathbb{R}^d$
for $k = 0, 1, 2, \ldots$ **do**
  Sample $i_k \in \{1, \ldots, n\}$ uniformly at random
  $g^k = \nabla f_{i_k}(x^k) - \nabla f_{i_k}(w^k) + \nabla f(w^k)$
  $x^{k+1} = x^k - \alpha g^k$
  $w^{k+1} = \begin{cases} x^k & \text{with probability } p \\ w^k & \text{with probability } 1-p \end{cases}$
end for

R It may look a bit strange, if not annoying, that w^{k+1} does not use the most recent information x^{k+1}. After we have finished the analysis, try to see if it is possible to modify the analysis so that we can use a better w^{k+1}.

8.2 Analysis

Before proceeding with the analysis, let's introduce some quantities

$$D_k = 4\alpha^2 \sum_{i=1}^{n} \|\nabla f_i(w^k) - \nabla f_i(x^*)\|^2$$
$$\Phi_k = \|x^k - x^*\|^2 + D_k$$

Lemma 8.1

$$\mathbf{E}_k\left[\|x^{k+1} - x^*\|^2\right] \leqslant (1-\mu\alpha)\|x^k - x^*\|^2 - 2\alpha(f(x^k) - f_*) + \alpha^2 \mathbf{E}_k\left[\|g^k\|^2\right] \tag{8.3}$$

Proof. This is the standard inequality that we derived many times, where we additionally used strong convexity of f. ■

Exercise 8.1 Complete the proof of Lemma 8.1. ▪

Lemma 8.2

$$\mathbf{E}_k\left[\|g^k\|^2\right] \leqslant 4L(f(x^k) - f_*) + \frac{1}{2\alpha^2 n} D_k \tag{8.4}$$

Proof. We have

$$\begin{aligned}\mathbf{E}_k\left[\|g^k\|^2\right] &= \mathbf{E}_k\left[\|\nabla f_{i_k}(x^k) - \nabla f_{i_k}(x^*) + \nabla f_{i_k}(x^*) - \nabla f_{i_k}(w^k) + \nabla f(w^k)\|^2\right] \\ &\leqslant 2\,\mathbf{E}_k\left[\|\nabla f_{i_k}(x^k) - \nabla f_{i_k}(x^*)\|^2\right] + 2\,\mathbf{E}_k\left[\|\nabla f_{i_k}(x^*) - \nabla f_{i_k}(w^k) + \nabla f(w^k)\|^2\right] \end{aligned} \tag{8.5}$$

For the first term we used the standard inequality that characterizes L-smooth functions (see Lemma 8.3)

$$\begin{aligned}\mathbf{E}_k\left[\|\nabla f_{i_k}(x^k) - \nabla f_{i_k}(x^*)\|^2\right] &\leqslant 2L\,\mathbf{E}_k\left[f_{i_k}(x^k) - f_{i_k}(x^*) - \langle \nabla f_{i_k}(x^*), x^k - x^*\rangle\right] \\ &= 2L(f(x^k) - f_*).\end{aligned}$$

For the second term, we only notice that

$$\begin{aligned}\mathbf{E}_k\left[\|\nabla f_{i_k}(x^*) - \nabla f_{i_k}(w^k) + \nabla f(w^k)\|^2\right] &= \mathbf{Var}\left[\|\nabla f_{i_k}(w^k) - \nabla f_{i_k}(x^*)\|^2\right] \\ &\leqslant \mathbf{E}_k\left[\|\nabla f_{i_k}(w^k) - \nabla f_{i_k}(x^*)\|^2\right] \\ &= \frac{1}{n}\sum_{i=1}^{n} \|\nabla f_i(w^k) - \nabla f_i(x^*)\|^2 = \frac{1}{4\alpha^2 n} D_k.\end{aligned}$$

■

Lemma 8.3 If φ is convex and L-smooth, then

$$\varphi(x) - \varphi(y) - \langle \nabla\varphi(y), x - y\rangle \geqslant \frac{1}{2L}\|\nabla\varphi(x) - \nabla\varphi(y)\|^2.$$

This lemma is very important in general in optimization. It tells us that for a convex L-smooth functions our basic gradient inequality can be improved.

Proof. Let us fix x, y and consider $h(u) = \varphi(u) - \varphi(y) - \langle \nabla\varphi(y), u - y\rangle$. The function h is convex, nonnegative: $h(u) \geqslant 0$ for all u, h is also L-smooth, and $y \in \operatorname{argmin} h$. Hence, one step of the gradient descent applied to h with the step $\alpha = \frac{1}{L}$ and initial point x

$$x^+ = x - \frac{1}{L}\nabla h(x)$$

by inequality (1.4) must satisfy

$$h(x^+) \leqslant h(x) - \frac{1}{2L}\|\nabla h(x)\|^2.$$

Hence,

$$h(x) \geqslant h(x^+) + \frac{1}{2L}\|\nabla h(x)\|^2 \geqslant \frac{1}{2L}\|\nabla h(x)\|^2 = \frac{1}{2L}\|\nabla\varphi(x) - \nabla\varphi(y)\|^2,$$

which is exactly the inequality we want to prove. ■

Lemma 8.4

$$\mathbf{E}_k[D_{k+1}] \leqslant \left(1 - \frac{1}{n}\right) D_k + 8\alpha^2 L(f(x^k) - f_*) \tag{8.6}$$

Proof. We have

$$\begin{aligned}\mathbf{E}_k[D_{k+1}] &= \frac{4\alpha^2}{n}\sum_{i=1}^n \|\nabla f_i(x_k) - \nabla f_i(x^*)\|^2 + \frac{4\alpha^2(n-1)}{n}\sum_{i=1}^n \|\nabla f_i(w_k) - \nabla f_i(x^*)\|^2 \\ &\leqslant 8\alpha^2 L(f(x^k) - f_*) + \left(1 - \frac{1}{n}\right) D_k,\end{aligned}$$

where in the last inequality we again used Lemma 8.3 ■

Lemma 8.5 If $\alpha \leqslant \frac{1}{6L}$, then

$$\mathbf{E}_k[\Phi_{k+1}] \leqslant (1 - \mu\alpha)\|x^k - x^*\|^2 + \left(1 - \frac{1}{2n}\right) D_k.$$

Proof.

$$\begin{aligned}\mathbf{E}_k[\Phi_{k+1}] &\leqslant (1 - \mu\alpha)\|x^k - x^*\|^2 - 2\alpha(f(x^k) - f_*) + \alpha^2 \mathbf{E}_k\left[\|g^k\|^2\right] + \left(1 - \frac{1}{n}\right) D_k + 8\alpha^2 L(f(x^k) - f_*) \\ &\leqslant (1 - \mu\alpha)\|x^k - x^*\|^2 - (2\alpha - 4\alpha^2 L - 8\alpha^2 L)(f(x^k) - f_*) + \left(1 - \frac{1}{n} + \frac{1}{2n}\right) D_k.\end{aligned}$$

Since $6\alpha L \leqslant 1$, the desired inequality follows. ■

Theorem 8.1 Let $\alpha = \frac{1}{6L}$, then

$$\mathbf{E}[\Phi_k] \leqslant \max\left\{1 - \frac{\mu}{6L}, 1 - \frac{1}{2n}\right\}^k \Phi_0.$$

This implies $\mathbf{E}[\Phi_k] \leqslant \varepsilon$ as long as

$$k \geqslant O\left(\left(n + \frac{L}{\mu}\right)\log\frac{\Phi_0}{\varepsilon}\right).$$

Exercise 8.2 Prove Theorem 8.1. ∎

Comments

As I said at the beginning of the lecture, the same situation (albeit with different rates) occurs when f is merely convex.

References

[1] Rie Johnson and Tong Zhang. "Accelerating Stochastic Gradient Descent using Predictive Variance Reduction". In: *Advances in Neural Information Processing Systems*. Volume 26. 2013. URL: https://proceedings.neurips.cc/paper_files/paper/2013/file/ac1dd209cbcc5e5d1c6e28598e8cbbe8-Paper.pdf.

[2] Dmitry Kovalev, Samuel Horváth, and Peter Richtárik. "Don't Jump Through Hoops and Remove Those Loops: SVRG and Katyusha are Better Without the Outer Loop". In: *Proceedings of the 31st International Conference on Algorithmic Learning Theory*. Volume 117. PMLR, 2020, pages 451–467. URL: https://proceedings.mlr.press/v117/kovalev20a.html.

Lecture 9: Mirror descent

R In this lecture $\|\cdot\|_2$ denotes the Euclidean norm, $\|\cdot\|$ denotes an abstract norm in $\mathbb{R}^d$.

9.1 Preliminaries

So far, we have worked exclusively with the Euclidean norm $\|\cdot\|_2$. This choice was not only for analysis but also for assumptions, such as all subgradients being bounded $\|g\|_2 \leqslant G$, or f being μ-strongly convex with respect to $\|\cdot\|_2$. However, sometimes the problem's structure may yield better estimates in a different norm. While in theory, we can switch between norms (as all norms are equivalent in finite-dimensional spaces), this switch can introduce undesirable constants. For example, if we have the bound $\|g\|_\infty \leqslant G$, switching to the Euclidean norm results in $\|g\|_2 \leqslant \sqrt{d}G$, adding a dimension-dependent factor to the algorithm's complexity.

Today, we will learn how to use the problem's geometry directly without relying on the Euclidean structure.

We suppose that $\mathbb{R}^d$ is equipped with the primal norm $\|\cdot\|$. In turn, the primal norm defines the dual norm

$$\|y\|_* = \max_{\|x\|=1} \langle y, x\rangle.$$

One can check that the dual of the ℓ_2-norm is again the ℓ_2-norm, the dual of the ℓ_1-norm is the ℓ_∞-norm, and the dual of the ℓ_p-norm is the ℓ_q-norm for $p > 1$ and $\frac{1}{p} + \frac{1}{q} = 1$.

The definition of the dual norm implies the Hölder inequality $\langle x, y\rangle \leqslant \|x\|\|y\|_*$.

Exercise 9.1

1. Check the latter statement.
2. The Hölder inequality is obviously a generalization of the Cauchy-Schwarz inequality. However, we usually prove the latter inequality properly, while for the Hölder we have just said that it follows from the definition. Why is it so?

■

9.2 Bregman divergence

Consider $\min_{x\in C} f(x)$, where C is a closed convex set and $f : \mathbb{R}^d \to \mathbb{R}$ is a convex function. If f is differentiable, then we can apply the projected gradient method

$$x^{k+1} = \operatorname*{argmin}_{x\in C} \left\{ f(x^k) + \langle \nabla f(x^k), x - x^k\rangle + \frac{1}{2\alpha}\|x - x^k\|_2^2 \right\}. \tag{9.1}$$

If we don't want to use the Euclidean norm, perhaps the simplest solution would be to change it to the abstract norm $\|\cdot\|$ in the update above. The easiest objection to such an atrocity is that we need to keep our update simple. After all, (9.1) is another optimization problem, and we don't want to make it difficult to solve.

A better way to deal with our objective is to introduce Bregman's divergence.

Definition 9.1 — Mirror map. A mirror map (or kernel) is a convex function $h\colon \mathbb{R}^d \to (-\infty, +\infty]$ such that

1. h is differentiable on $\operatorname{dom}\partial h$;
2. $C \subset \operatorname{dom} h$;
3. h is 1-strongly convex over C wrt $\|\cdot\|$.

Unfortunately, the definition is a bit more technical than we would like to have. For us it is important to remember that h is a convex function with some nice properties.

Definition 9.2 — Bregman divergence. Given a mirror map h, the Bregman divergence (or distance) $D\colon \operatorname{dom} h \times \operatorname{dom}\partial h \to \mathbb{R}$ is defined as

$$D(x,y) = h(x) - h(y) - \langle \nabla h(y), x - y\rangle.$$

We immediately obtain the following properties of D:

- $D(x,y) \geqslant 0$.
- $D(x,y) \neq D(y,x)$ (in general).
- For all y, the function $x \mapsto D(x,y)$ is convex.
- $D(x,y) \geqslant \frac{1}{2}\|x-y\|^2$.

Because of the second property, D is not a real distance. Nevertheless, nothing prevents us from using it to measure the distance between x and y. If x and y are close, then from the Taylor series it follows that $D(x,y) \approx \frac{1}{2}\langle \nabla^2 h(y)(x-y), x-y\rangle$. In this sense, the function h imposes its geometry on how we measure the distance between points.

Once we define D, we can formulate the mirror descent algorithm

$$x^{k+1} = \operatorname*{argmin}_{x\in C}\left\{ f(x^k) + \langle \nabla f(x^k), x - x^k\rangle + \frac{1}{\alpha} D(x, x^k)\right\}.$$

Similarly, one can formulate a more general update using g^k, which can represent a gradient, subgradient, or stochastic (sub)gradient.

$$x^{k+1} = \operatorname*{argmin}_{x\in C}\left\{ f(x^k) + \langle g^k, x - x^k\rangle + \frac{1}{\alpha} D(x, x^k)\right\}. \tag{9.2}$$

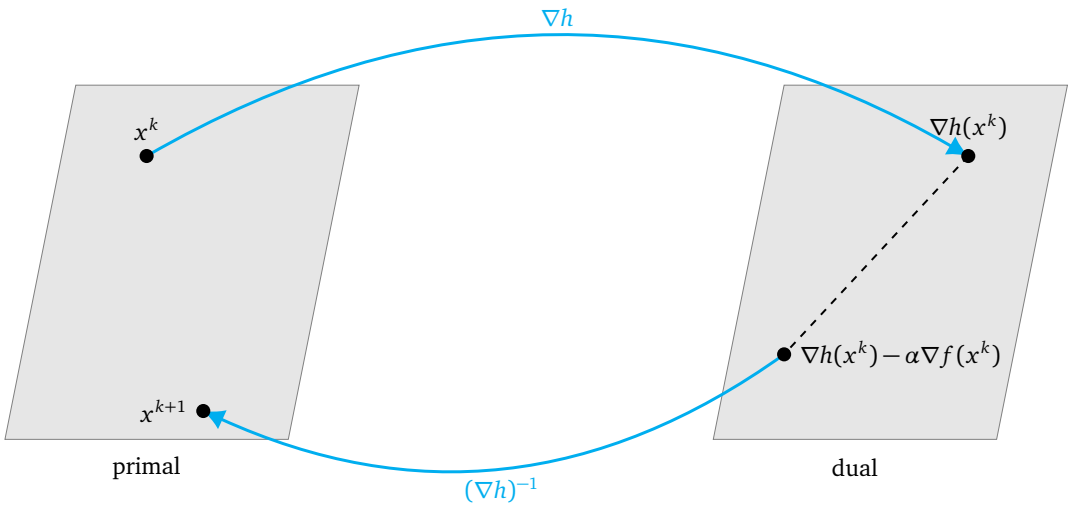
9.2.1 Examples

▪ **Example 9.1** Let $C = \mathbb{R}^d$ and $h(x) = \frac{1}{2}\|x\|_2^2$. Then it is easy to see that $D(x,y) = \frac{1}{2}\|x-y\|_2^2$, hence we recover the usual Euclidean metric. ▪

▪ **Example 9.2** Let $C = \mathbb{R}^d$ and $h(x) = \frac{1}{2(p-1)}\|x\|_p^2$, $p \in (1,2]$. Then one can check (we won't) that $\nabla(x) = \frac{1}{p-1}\|x\|_p^{2-p}(\operatorname{sign} x_1 |x_1|^{p-1}, \ldots, \operatorname{sign} x_d |x_d|^{p-1})$. We won't substitute this into D, but once we have h and ∇h, it is straightforward, just tedious. ▪

The last example, while hiding behind the complexity of the formula, illuminates another key point of mirror descent. It is useful to keep in mind that whenever we talk about gradients or subgradients, these vectors live in *dual* space as opposed to x which lives in *primal* space. On the other hand, the basic GD update uses addition between a primal vector x^k and the

dual vector $\nabla f(x^k)$. We don't have a problem with this, because in finite dimensional space all these spaces are equivalent. But if our original space were L^p, then the dual would be L^q, with $\frac{1}{p}+\frac{1}{q}=1$, and then adding x^k and $\nabla f(x^k)$ won't be possible. This is where the mirror map comes in. We have something like the following picture.



One of the reasons we assumed so many things about h is that both ∇h and $(\nabla h)^{-1}$ must be well defined to make sense of that picture. If we have an additional constraint set C, the picture above will look a bit different.

▪ **Example 9.3** Let $C=\Delta_d=\{x\in\mathbb{R}^d_+\colon \sum_{i=1}^d x_i=1\}$. The primal norm here will be $\|\cdot\|_1$. Hence, the dual is $\|\cdot\|_\infty$. For the mirror map h we choose the negative entropy

$$h(x)=\begin{cases}\sum_{i=1}^d x_i\log x_i=\langle x,\log x\rangle & \text{if } x\in\mathbb{R}^d_{++}\\ +\infty & \text{otherwise.}\end{cases}$$

Then this h induces the Bregman distance D

$$D(x,y)=\sum_{i=1}^d x_i\log\frac{x_i}{y_i}=\langle x,\log x-\log y\rangle.$$

This is the famous Kullback-Leibler divergence, prominent in information theory, probability, and statistics. Often it is denoted as $\mathrm{KL}(x\parallel y)$.

The fact that h is 1-strongly convex wrt $\|\cdot\|_1$ is called the Pinsker inequality

$$D(x,y)\geqslant\frac{1}{2}\|x-y\|_1^2.$$

It is a good exercise to think how to prove this inequality (relying on convexity tools should be sufficient). ▪

9.3 Analysis

The next identity will be fundamental for us.

Lemma 9.1 For $a,b,c\in\mathbb{R}^d$ it holds

$$D(a,c)-D(a,b)-D(b,c)=\langle\nabla h(b)-\nabla h(c),a-b\rangle. \tag{9.3}$$

Proof. We use $D(a,c) = h(a) - h(c) - \langle \nabla h(c), a - c\rangle$ and similar formulas for $D(a,b)$, $D(b,c)$ and sum them up to get (9.3) ■

Exercise 9.2 Complete the proof of Lemma 9.1. ▪

Exercise 9.3 What does identity (9.3) recover when $D(x,y) = \frac{1}{2}\|x-y\|_2^2$? ▪

Now let's analyse the algorithm (9.2). We can transform it as follows

$$\begin{aligned} x^{k+1} &= \operatorname*{argmin}_{x\in C}\left\{ f(x^k) + \langle g^k, x - x^k\rangle + \frac{1}{\alpha_k} D(x, x^k)\right\} \\ &= \operatorname*{argmin}_{x\in C}\left\{\langle \alpha_k g^k, x\rangle + h(x) - \langle \nabla h(x^k), x\rangle\right\} \\ &= \operatorname*{argmin}_{x\in C}\left\{\langle \alpha_k g^k - \nabla h(x^k), x\rangle + h(x)\right\} \end{aligned} \tag{9.4}$$

Below we analyze MD in the stochastic setting. It is clear that the same proof works in the deterministic, when we use subgradients for g^k.

Theorem 9.1 Suppose that $f : \mathbb{R}^d \to \mathbb{R}$ is convex, $\mathbf{E}_k\left[\|g^k\|^2\right] \leqslant G^2$, and $D(x^*, x^1) \leqslant R^2$. Then the algorithm (4.7) satisfies

$$\mathbf{E}\left[f(\bar{x}^k) - f_*\right] \leqslant \frac{R^2}{\sum_{k=1}^K \alpha_k} + \frac{G^2 \sum_{k=1}^K \alpha_k^2}{2\sum_{k=1}^K \alpha_k},$$

where $\bar{x}^K = \frac{1}{A_K}\sum_{k=1}^K \alpha_k x_k$ and $A_K = \sum_{k=1}^K \alpha_k$.

The proof we are going to study is the most instructive so far. It may look a bit longer that our usual proof of the subgradient method or SGD, but it more natural and in fact is easier to come up with.

Proof. We would like to use an analytical expression for x^{k+1}, but the only thing we know about it is that x^{k+1} is a solution of the optimization problem. Hence, let's write down the optimality condition that characterizes x^{k+1}. Applying Corollary 3.1, we obtain

$$\langle \nabla h(x^{k+1}) - \nabla h(x^k) + \alpha_k g^k, x - x^{k+1}\rangle \geqslant 0 \quad \forall x \in C.$$

Using identity (9.3), we obtain

$$\begin{aligned} D(x, x^{k+1}) + D(x^{k+1}, x^k) &\leqslant D(x, x^k) + \alpha_k \langle g^k, x - x^{k+1}\rangle \\ &= D(x, x^k) + \alpha_k \langle g^k, x - x^k\rangle + \alpha_k \langle g^k, x^k - x^{k+1}\rangle \\ &\leqslant D(x, x^k) + \alpha_k \langle g^k, x - x^k\rangle + \frac{\alpha_k^2}{2}\|g^k\|_*^2 + \frac{1}{2}\|x^{k+1} - x^k\|^2, \end{aligned} \tag{9.5}$$

where in the last inequality we used Hölder's inequality. By strong convexity of h, we have that $D(x^{k+1}, x^k) \geqslant \frac{1}{2}\|x^{k+1} - x^k\|^2$. Hence, after taking expectation $\mathbf{E}_k$ from both sides, we arrive at

$$\begin{aligned} \mathbf{E}_k\left[D(x, x^{k+1})\right] &\leqslant D(x, x^k) + \alpha_k \langle \tilde{\nabla} f(x^k), x - x^k\rangle + \frac{\alpha_k^2}{2}\mathbf{E}_k\left[\|g^k\|_*^2\right] \\ &\leqslant D(x, x^k) - \alpha_k(f(x^k) - f(x)) + \frac{\alpha_k^2 G^2}{2}. \end{aligned} \tag{9.6}$$

From this moment, we set $x = x^*$. Taking the total expectation and summing over $k = 1, \dots, K$ yields

$$\sum_{k=1}^{K} \alpha_k \mathbf{E}\left[f(x^k) - f_*\right] \leqslant D(x^*, x^1) + \frac{G^2}{2} \sum_{k=1}^{K} \alpha_k^2 \leqslant R^2 + \frac{G^2}{2} \sum_{k=1}^{K} a_k^2.$$

Jensen's inequality applied to the left-hand side completes the proof. ■

Corollary 9.1 — Fixed stepsize. If $\alpha_k = \frac{R}{G}\sqrt{\frac{2}{K}}$ for all $k \in 1, \dots, K$ and $\bar{x}^K = \frac{1}{K}\sum_{k=1}^{K} x^k$, then

$$\mathbf{E}\left[f(\bar{x}^K) - f_*\right] \leqslant RG\sqrt{\frac{2}{K}} = O\left(\frac{RG}{\sqrt{K}}\right).$$

Exercise 9.4 Why do we have a slightly different constant in the convergence rate than in Corollary 4.2? ▪

Obviously, we can also get a similar convergence rate by using a stepsize $\alpha_k \sim \frac{1}{\sqrt{k}}$ for $k \geqslant 1$. This will result in an additional logarithmic factor, the same as in the Euclidean setting. Finally, if C is a compact set, we can use a slightly more sophisticated analysis to get rid of this logarithmic dependence (as in Theorem 3.2).

9.4 Mirror descent on the probability simplex

Let's revisit Example 9.3. From the general update (9.4), we would like to derive an explicit form for x^k. In other words, we want to solve

$$x^{k+1} = \operatorname*{argmin}_{x \in \Delta^d} \left\{\langle \alpha g^k - \nabla h(x^k), x\rangle + h(x)\right\}.$$

Since $h(x) = \langle x, \log x\rangle$ for $x \in \Delta^d$ and $\nabla h(x) = \log x + \mathbf{1}$, we have to solve an equivalent problem

$$x^{k+1} = \operatorname*{argmin}_{x \in \Delta^d} \{\langle u, x\rangle + \langle x, \log x\rangle\}, \qquad u := \alpha g^k - \log x - \mathbf{1}.$$

This is a simple constraint optimization problem and we can find its solution analytically. We have $d+1$ constraints: d inequalities $x_i \geqslant 0$ and one equality $\sum_i x_i = \langle \mathbf{1}, x\rangle = 1$. Assume first that this problem has a solution x^* with all positive entries. Then first d constraints won't be active and we can discard them. We can form a Lagrange function

$$L(x, y) = \langle u, x\rangle + \langle x, \log x\rangle + y\left(\langle \mathbf{1}, x\rangle - 1\right)$$

and write its optimality condition:

$$\begin{aligned} \nabla_x L(x, y) = 0 \quad &\Longleftrightarrow \quad u + \log x^* + \mathbf{1} + y^* \mathbf{1} = 0 \\ \nabla_y L(x, y) = 0 \quad &\Longleftrightarrow \quad \langle \mathbf{1}, x^*\rangle = 1. \end{aligned}$$

From this we obtain that $x_i^* = e^{-u_i - 1 - y^*} = e^{-u_i} \cdot \beta$ with $\beta = e^{-1-y^*}$. From the second equation we obtain that $\beta = \sum_i e^{-u_i}$ and hence

$$x_i^* = \frac{e^{-u_i}}{\sum_{j=1}^{d} e^{-u_j}}, \quad i = 1, \dots, d.$$

Notice that x^* indeed has all positive coordinates, so our assumption in the beginning wasn't restrictive. Substituting back the value of u and using $\circ$ for the elementwise multiplication, we get $e^{-u} = x^k \circ e^{-\alpha g^k}$. Therefore, we arrive at the following update

$$\begin{aligned} y^k &= x^k \circ e^{-\alpha g^k} \\ x^{k+1} &= \frac{y^k}{\|y^k\|_1}. \end{aligned}$$

Now suppose we are given that $\|g^k\|_\infty \leqslant G$. As we have discussed before, if we switch to the Euclidean setting, we have to use $\|g^k\|_\infty \leqslant \sqrt{d}G$. To make comparison of Euclidean vs non-Euclidean fair, we also should compare constants R, which we denote, for brevity, as $R_{\ell_1} = D(x^*, x^1)^{1/2}$ and $R_{\ell_2} = \|x^* - x^1\|_2$.

For Euclidean case it's easy, as $R_{\ell_2} \leqslant \operatorname{diam} \Delta^d = 2$. For mirror descent, by selecting $x^1 = \frac{1}{d}\mathbf{1}$, we get

$$D(x, x^1) = \langle x, \log x - \log x^1 \rangle = \langle x, \log x \rangle + \log d \leqslant \log d,$$

where the last inequality is the consequence of the fact that negentropy is non-positive on Δ^d:

$$\langle x, \log x \rangle = \sum_i x_i \log x_i \leqslant 0 \qquad \text{since} \quad \log x_i \leqslant 0 \ \forall i.$$

Therefore, $R_{\ell_1} \leqslant \sqrt{\log d}$. All together, we have the following bounds for $\mathbf{E}\left[f(\bar{x}^K) - f_*\right]$:

$$\text{SGD: } O\left(\frac{G\sqrt{d}}{\sqrt{K}}\right) \qquad \text{vs.} \qquad \text{Mirror SGD: } O\left(\frac{G\sqrt{\log d}}{\sqrt{K}}\right),$$

so at least in this case we can see a potential advantage of the mirror descent framework.

Comments

Unfortunately, in the most general case, a rigorous treatment of mirror descent — proving that the iterates are well-defined, as well as their convergence — can be a bit technical. That's why we glossed over those details. Paper [2] or lecture notes [3] are a good short reference for those who want more details. For a more thorough treatment, I recommend [1].

References

[1] Amir Beck. *First-order methods in optimization*. SIAM, 2017.

[2] Amir Beck and Marc Teboulle. "Mirror descent and nonlinear projected subgradient methods for convex optimization". In: *Operations Research Letters* 31.3 (2003), pages 167–175.

[3] John C Duchi. "Introductory lectures on stochastic optimization". In: *The mathematics of data* 25 (2018), pages 99–186. URL: https://web.stanford.edu/~jduchi/PCMIConvex/Duchi16.pdf.

Lecture 10: Weighted majority algorithm

10.1 Learning from expert advice

Today, we forget about our usual optimization algorithms and switch to a different context. Alice plays T rounds of a game against the environment. In each round t, Alice must choose between a binary decision: YES or NO. There are n experts available to help her decide, each providing their own YES or NO recommendation. Alice then makes her choice and receives feedback from the environment indicating her loss, which is 0 if she guessed correctly and 1 otherwise. Alice aims to minimize her total loss over T rounds. The environment can determine the loss, as it pleases, and can be adversarial, potentially even knowing Alice's algorithm. We summarize the game below.

Rules of the game:

1. T rounds, n experts.
2. Alice receives prediction from each experts: YES or NO.
3. Alices makes its own prediction: YES or NO.
4. Alice receives a correct answer.

Given this, a reasonable strategy for Alice is to make uniformly random choices each round, making it impossible for an adversarial environment to predict her moves[1]. In this scenario, Alice would expect an average loss of $T/2$. This is the best guaranteed outcome for Alice against an adversarial environment (we won't prove it).

However, we are less interested in Alice's total number of mistakes and more interested in comparing her mistakes to those of the best expert. Thus, the only way for an adversarial environment to cause Alice a significant loss is to create a sequence of loss functions that even the best expert incurs a substantial loss.

Algorithm 2 Weighted Majority Algorithm (WMA)

Initialization: Fix $\alpha \in \left(0, \frac{1}{2}\right]$ and $w_1(i) = 1$ for every $i \in [n]$.
for $t = 1, 2 \ldots, T$ **do**
1. Make the prediction based on the weights $w_t(1), \ldots, w_t(n)$ (which one has a higher total weight of experts).
2. Penalize wrong experts: for every expert i who predicts wrongly, set
$$w_{t+1}(i) = (1-\alpha)w_t(i)$$

end for

[1] For this, we need to assume that the environment decides on the loss independently of Alice's choice.

Consider the Weighted Majority Algorithm given by Algorithm 2.

Let $m_t(i)$ be the number of expert i's mistakes after t rounds and M_t be the number of mistakes our algorithm has made.

Theorem 10.1 After T iteration, the WMA satisfies the following inequality for every i:

$$M_T \leqslant 2(1+\alpha)m_T(i) + \frac{2\log n}{\alpha}.$$

In particular, this holds for i with the smallest $m_T(i)$ (the best expert).

Proof. It is obvious that $w_{t+1}(i) = (1-\alpha)^{m_t(i)}$. Let $\Phi_t = \|w_t\|_1 = \sum_i w_t(i)$ be the potential function. In the beginning, we have $\Phi_1 = n$. Each time we make a mistake, the weighted majority of experts also made a mistake, hence at least half of the total weight decrease by a factor $1-\alpha$. Thus,

$$\Phi_{t+1} = \sum_{\text{right}} w_{t+1}(i) + \sum_{\text{wrong}} w_{t+1}(i) = \sum_{\text{right}} w_t(i) + (1-\alpha)\sum_{\text{wrong}} w_t(i) \leqslant \Phi_t\left(1-\frac{\alpha}{2}\right).$$

A simple recursion gives $\Phi_{T+1} \leqslant n(1-\alpha/2)^{M_T}$. On the other hand, $\Phi_{T+1} \geqslant w_{T+1}(i) = (1-\alpha)^{m_T(i)}$. By combining these two bounds and using

$$\log(1-\alpha/2) \leqslant -\alpha/2, \quad \forall \alpha$$
$$-\alpha-\alpha^2 \leqslant \log(1-\alpha), \quad \forall \alpha \in \left(0, \frac{1}{2}\right],$$

we obtain the desired inequality. ∎

10.2 Randomized weighted majority algorithm

In this version of the algorithm the player makes her decision randomly: she chooses an expert i with probability $p_i = \frac{w_t(i)}{\|w_t\|_1}$ at round t. The rest of the algorithm remains the same.

Theorem 10.2 After T iteration, for every i the randomized WMA algorithm satisfies

$$M_T \leqslant (1+\alpha)m_T(i) + \frac{\log n}{\alpha}.$$

In particular, this holds for i with the smallest $m_T(i)$ (the best expert).

Proof. As before, let $\Phi_t = \|w_t\|_1$. Denote

$\Delta_t = M_t - M_{t-1}$ is 1 if the algorithm made a mistake at round t and 0 otherwise

$\delta_t(i) = m_t(i) - m_{t-1}(i)$ is 1 if the ith expert made a mistake at round t and 0 otherwise

Notice that with this notation $\mathbf{E}[\Delta_t] = \sum_i p_t(i)\delta_t(i)$. We have

$$\Phi_{t+1} = \sum_i w_t(i)(1-\alpha\delta_t(i)) = \|w_t\|_1\left(1-\alpha\sum_i p_t(i)\delta_t(i)\right) = \Phi_t(1-\alpha\mathbf{E}[\Delta_t]) \leqslant \Phi_t e^{-\alpha\mathbf{E}[\Delta_t]}.$$

This implies $\Phi_{T+1} \leqslant ne^{-\alpha\mathbf{E}[M_T]}$. On the other hand, $\Phi_{T+1} \geqslant w_{T+1}(i) = (1-\alpha)^{m_T(i)}$. By combining these two bounds and using

$$\log(1-\alpha/2) \leqslant -\alpha/2, \quad \forall \alpha$$
$$-\alpha-\alpha^2 \leqslant \log(1-\alpha), \quad \forall \alpha \in \left(0, \frac{1}{2}\right],$$

we obtain

$$-m_T(i)\alpha(1+\alpha) \leqslant \log n - \alpha \mathbf{E}[M_T]$$

and the desired inequality follows immediately. ■

Let m_T^* denote $m_T(i)$ for the bext expert i, then by choosing $\alpha^* = \sqrt{\frac{\log n}{m_T^*}}$, we get

$$M_T - m_T^* \leqslant 2\sqrt{m_T^* \log n}.$$

Comments

More material on this and related topics can be found in [1].

References

[1] Elad Hazan. *Introduction to online convex optimization*. MIT Press, 2022.

Lecture 11: Multiplicative Weight Update

The concept of experts being right or wrong can be extended to arbitrary loss functions. At each time $t = 1, \ldots, T$, suppose the algorithm produces a probability vector $p_t \in \Delta^n$, from which we sample index $i_t \in [n]$. The adversary produces losses $\ell_t = (\ell_t(1), \ldots, \ell_t(n))$, where $\ell_t(i)$ represents the loss incurred by the i-th expert at time t. Our expected cost for sampling decision i_t from p_t is given by

$$\mathbf{E}[\ell_t(i_t)] = \langle \ell_t, p_t \rangle.$$

Therefore, the total expected cost over all rounds is $\sum_{t=1}^{T} \langle \ell_t, p_t \rangle$. After T rounds, once we know all the losses ℓ_t, we (or the algorithm) regret not having followed the advice of the best expert in hindsight. This regret is naturally expressed as

$$\text{Regret}_T = \sum_{t=1}^{T} \langle \ell_t, p_t \rangle - \min_{p \in \Delta^n} \sum_{t=1}^{T} \langle \ell_t, p \rangle = \sum_{t=1}^{T} \langle \ell_t, p_t \rangle - \min_{i \in [n]} \sum_{t=1}^{T} \ell_t(i). \tag{11.1}$$

(Explain the second equality above.) Notice that we consistently use the term "loss", even though we allow $\ell_t(i)$ to be negative, which actually represents gain.

Algorithm 3 Multiplicative Weights Update

Initialization: Fix $\alpha \in \left(0, \frac{1}{2}\right]$ and $w_1(i) = 1$ for every $i \in [n]$.
for $t = 1, 2 \ldots, T$ **do**
1. Sample a decision i from the probability distribution $p_t = \frac{w_t}{\|w_t\|_1}$.
2. Observe the loss $\ell_t = (\ell_t(1), \ldots, \ell_t(n))$.
3. Update weights: $w_{t+1}(i) = w_t(i)(1 - \alpha \ell_t(i))$ for all $i \in [n]$.

end for

Theorem 11.1 Suppose that $\ell_t(i) \in [-1, 1]$ for all t and i. Then the Multiplicative Weights algorithm guarantees that after T iterations, for any decision i one has

$$\sum_{t=1}^{T} \langle \ell_t, p_t \rangle - \sum_{t=1}^{T} \ell_t(i) \leqslant \alpha \sum_{t=1}^{T} |\ell_t(i)| + \frac{\log n}{\alpha}$$

We select the best expert i, use $|\ell_t(i)| \leqslant 1$ and the stepsize $\alpha = \sqrt{\frac{\log n}{T}}$ to obtain the bound

$$\text{Regret}_T \leqslant 2\sqrt{T \log n}.$$

Say, we have $T = 10000$ and $n = 100$. Then the bound above implies $\text{Regret}_T \leqslant 430$, which is not that bad, taking into account how many rounds we play.

Proof. Let $\Phi_t = \|w_t\|_1 = \sum_i w_t(i)$. We have

$$\begin{aligned}\Phi_{t+1} &= \sum_i w_{t+1}(i) = \sum_i w_t(i)(1-\alpha\ell_t(i)) = \Phi_t - \alpha\Phi_t \sum_i \ell_t(i)p_t(i)\\ &= \Phi_t(1-\alpha\langle \ell_t, p_t\rangle) \leqslant \Phi_t \exp(-\alpha\langle \ell_t, p_t\rangle).\end{aligned}$$

Thus, after T rounds, we have

$$\Phi_{T+1} \leqslant \Phi_1 \exp\left(-\alpha\sum_{t=1}^{T}\langle \ell_t, p_t\rangle\right) = n\exp\left(-\alpha\sum_{t=1}^{T}\langle \ell_t, p_t\rangle\right) \tag{11.2}$$

Next, we use

$$\begin{aligned}(1-\alpha)^x &\leqslant 1-\alpha x, \quad \forall x \in [0,1]\\ (1+\alpha)^{-x} &\leqslant 1-\alpha x, \quad \forall x \in [-1,0]\end{aligned} \tag{11.3}$$

For every i, we have

$$\Phi_{T+1} \geqslant w_{T+1}(i) = \prod_{t=1}^{T}(1-\alpha\ell_t(i)) \geqslant (1-\alpha)^{\sum_{\geqslant 0}\ell_t(i)} \cdot (1+\alpha)^{-\sum_{<0}\ell_t(i)}. \tag{11.4}$$

Taking logarithms in (11.5) and (11.4), we get

$$\log n - \alpha\sum_{t=1}^{T}\langle \ell_t, p_t\rangle \geqslant \sum_{\ell_t(i)\geqslant 0}\ell_t(i)\log(1-\alpha) - \sum_{\ell_t(i)<0}\ell_t(i)\log(1+\alpha).$$

With some little algebra,

$$\begin{aligned}\sum_{t=1}^{T}\langle \ell_t, p_t\rangle &\leqslant \frac{\log n}{\alpha} + \frac{1}{\alpha}\sum_{\geqslant 0}\ell_t(i)\log\frac{1}{1-\alpha} + \frac{1}{\alpha}\sum_{<0}\ell_t(i)\log(1+\alpha)\\ &\leqslant \frac{\log n}{\alpha} + \frac{1}{\alpha}\sum_{\geqslant 0}\ell_t(i)(\alpha+\alpha^2) + \frac{1}{\alpha}\sum_{<0}\ell_t(i)(\alpha-\alpha^2)\\ &= \frac{\log n}{\alpha} + \sum_{t=1}^{T}\ell_t(i) + \alpha\sum_{\geqslant 0}\ell_t(i) - \alpha\sum_{<0}\ell_t(i)\\ &= \frac{\log n}{\alpha} + \sum_{t=1}^{T}\ell_t(i) + \alpha\sum_{t=1}^{T}|\ell_t(i)|,\end{aligned}$$

where we used $\log(\frac{1}{1-\alpha}) \leqslant \alpha+\alpha^2$ and $\log(1+\alpha) \geqslant \alpha-\alpha^2$ for $\alpha \in (0,1/2)$. ∎

11.1 The Hedge algorithm

Instead of updating weights by multiplying with factors $(1-\alpha\ell_t(i))$, we can use exponential factor:

$$w_{t+1}(i) = w_t(i)\cdot e^{-\alpha\ell_t(i)}.$$

With this change, previous MWA becomes the Hedge algorithm. Since α is a small number (from previos analysis we know that optimal $\alpha^* = \sqrt{\frac{\log n}{T}}$), the approximation $e^{-\alpha\ell_t(i)} \approx 1-\alpha\ell_t(i)$ suggest that both algorithm should perform similarly.

Theorem 11.2 Suppose $\ell_t(i) \in [-1,1]$ and $\alpha \in (0,1]$. Then the Hedge algorithm guarantees that after T rounds, for any decision i,

$$\sum_{t=1}^{T}\langle \ell_t, p_t\rangle - \sum_{t=1}^{T}\ell_t(i) \leqslant \alpha \sum_{t=1}^{T}\langle \ell_t^2, p_t\rangle + \frac{\log n}{\alpha}.$$

Proof. For $\Phi_t = \|w_t\|_1$, we have

$$\begin{aligned}
\Phi_{t+1} &= \sum_i w_t(i) e^{-\alpha \ell_t(i)} \\
&= \Phi_t \sum_i p_t(i) e^{-\alpha \ell_t(i)} \\
&\leqslant \Phi_t \sum_i p_t(i)\left(1 - a\ell_t(i) + \alpha^2 \ell_t^2(i)\right) && e^{-x} \leqslant 1 - x + x^2 \ \text{ for } |x| \leqslant 1 \\
&= \Phi_t (1 - \alpha\langle \ell_t, p_t\rangle + \alpha^2 \langle \ell_t^2, p_t\rangle) \\
&\leqslant \Phi_t \exp\left(-\alpha\langle \ell_t, p_t\rangle + \alpha^2 \langle \ell_t^2, p_t\rangle\right) && 1 - x \leqslant e^{-x} \ \text{ for all } x.
\end{aligned}$$

Thus, after T rounds, we have

$$\Phi_{T+1} \leqslant n \exp \sum_{t=1}^{T}(-\alpha\langle \ell_t, p_t\rangle + \alpha^2 \langle \ell_t^2, p_t\rangle). \tag{11.5}$$

Now, a lower bound. For every i, we have

$$\Phi_{T+1} \geqslant w_{T+1}(i) = w_T e^{-\alpha \ell_T(i)} = \exp\left(-\alpha \sum_{t=1}^{T}\ell_t(i)\right).$$

Combining two bounds and taking the logarithm, we obtain

$$-\sum_{t=1}^{T}\ell_t(i) \leqslant \log n + \sum_{t=1}^{T}(-\alpha\langle \ell_t, p_t\rangle + \alpha^2 \langle \ell_t^2, p_t\rangle),$$

which is equivalent to the desired inequality. ∎

11.1.1 Analysis via mirror descent

Notice, that we can express the Hedge algorithm as

$$\begin{aligned}
p_t(i) &= \frac{w_t(i)}{\|w_t\|_1} \\
w_{t+1}(i) &= w_t(i) \cdot \exp(-\alpha \ell_t(i)).
\end{aligned}$$

We can rewrite it as

$$\begin{aligned}
w_{t+1}(i) &= w_t(i) \cdot \exp(-\alpha \ell_t(i)) \\
p_{t+1}(i) &= \frac{w_{t+1}(i)}{\|w_{t+1}\|_1}.
\end{aligned}$$

Since we care only about p_t (the weights w_t are only auxiliary), the above scheme generates the same p_t as

$$\begin{aligned}
w_{t+1}(i) &= p_t(i) \cdot \exp(-\alpha \ell_t(i)) \\
p_{t+1}(i) &= \frac{w_{t+1}(i)}{\|w_{t+1}\|_1}.
\end{aligned}$$

But the latter algorithm is exactly the update of mirror descent on the probability simplex! At first glance, it may seem strange since we previously minimized a single function f, which is not present here. In fact, the mirror descent framework is more general. Suppose we run mirror descent in each iteration with a different function $f_t(p) = \langle \ell_t, p \rangle$, using its gradient $\nabla f_t(p) = \ell_t$. Since the analysis of mirror descent involved only one iteration before summation, inequality (9.6) will still hold:

$$\begin{aligned} D(p, p_{t+1}) &\leqslant D(p, p_t) + \alpha \langle \nabla f_t(p_t), p - p_t \rangle + \frac{\alpha^2 \|\ell_t\|_*^2}{2} \\ &= D(p, p_t) + \alpha \langle \ell_t, p - p_t \rangle + \frac{\alpha^2 \|\ell_t\|_\infty^2}{2}. \end{aligned}$$

Compared to (9.6), here we removed the expectation, since ∇f_t is a deterministic gradient of f_t. We changed x^k to p_t to match with the previous material, used a fixed stepsize α, and a particular dual norm $\|\cdot\|_* = \|\cdot\|_\infty$.

Hence, we obtain

$$\langle \ell_t, p_t - p \rangle \leqslant \frac{1}{\alpha} (D(p, p_t) - D(p, p_{t+1})) + \frac{\alpha \|\ell_t\|_\infty^2}{2}.$$

Summing up this inequality over $t = 1, \dots, T$ and using that $D(p, p_1) \leqslant \log n$ for $p_1 = \frac{1}{n}\mathbf{1}$ (as we proved before), we deduce

$$\sum_{t=1}^T \langle \ell_t, p_t \rangle - \sum_{t=1}^T \langle \ell_t, p \rangle \leqslant \frac{\log n}{\alpha} + \frac{\alpha}{2} \sum_{t=1}^T \|\ell_t\|_\infty^2.$$

Hence, we can conclude

$$\mathrm{Regret}_T = \sum_{t=1}^T \langle \ell_t, p_t \rangle - \min_{p \in \Delta^n} \sum_{t=1}^T \langle \ell_t, p \rangle \leqslant \frac{\log n}{\alpha} + \frac{\alpha}{2} \sum_{t=1}^T \|\ell_t\|_\infty^2.$$

This is a very similar bound to the one we have in Theorem 11.2. There the right-hand side can be estimated as

$$\text{RHS in Th. (11.2)} = \alpha \sum_{t=1}^T \langle (\ell_t^2, p_t \rangle + \frac{\log n}{\alpha} \leqslant \alpha \sum_{t=1}^T \|\ell_t\|_\infty^2 \|p\|_1 + \frac{\log n}{\alpha} = \frac{\log n}{\alpha} + \sum_{t=1}^T \|\ell_t\|_\infty^2.$$

If we compare those final bounds, then MD is slightly sharper — it has an extra $\frac{1}{2}$. However, the original bound of the Hedge algorithm in Theorem 11.2 used $\langle (\ell_t^2, p_t \rangle$ and this can be much smaller than $\|\ell_t\|_\infty^2$.

Comments

MWU algorithm is extremely popular in the theoretical computer science community. A nice review on it with many applications and historical developments can be found in [1]. There is also a nice series of blogposts on it.

References

[1] Sanjeev Arora, Elad Hazan, and Satyen Kale. "The multiplicative weights update method: a meta-algorithm and applications". In: *Theory of computing* 8.1 (2012), pages 121–164.

Lecture 12: Two-person zero-sum games

12.1 Introduction

Today we continue playing games, but this time there are two players. A zero-sum means is that one player's loss is the other player's gain and vice versa.

▪ **Example 12.1 — Hider vs. Chooser.** Hider has two 1-euro coins. He either places one coin in his left hand (leaving the right hand empty) or both coins in his right hand. Chooser then picks a hand and takes all the coins in it. Without an extra bonus, Hider is at a disadvantage, so the question is: how much should Chooser pay Hider to make the game fair?

Formally, we can represent this situation with the following table from Chooser's perspective:

Chooser \ Hider	L	R
L	1	0
R	0	2

From the table, we see that no deterministic strategy will satisfy Hider. Therefore, suppose Hider plays probabilistically. Let Hider choose the first column with probability y_1 and the second column with $1-y_1$. In the worst case, Hider's average loss, if Chooser plays optimally, is $\max\{y_1, 2(1-y_1)\}$. Naturally, Hider wants to minimize this quantity:

$$\min_{y_1} \max\{y_1, 2(1-y_1)\}.$$

This can be easily solved, yielding $y_1 = \frac{2}{3}$ with the value of that quantity also being $\frac{2}{3}$.

Analogously, solving the maximization problem from Chooser's perspective, we obtain:

$$\max_{x_1} \min\{x_1, 2(1-x_1)\} = \frac{2}{3}.$$

This is called the value of the game — the value Chooser can guarantee to gain and Hider to lose (on average). Neither player has an incentive to deviate from their strategy.

To make this a fair game, Chooser must pay Hider 2/3 euros. Hider wouldn't accept anything less, and anything more would be a loss for Chooser.

▪

12.2 Zero-sum games

A two-person zero-sum game is described as follows. We have an $m \times n$ matrix A, called the *payoff* matrix. Simultaneously, Player I selects action i (the i-th row) and Player II selects action j (the j-th column). Their selections are revealed, and Player II pays Player I a_{ij}. Note that a negative gain becomes a loss and vice versa, but for convenience, we always refer to these as gains for Player I and losses for Player II.

Player I picks the i-th row, aiming to maximize his worst-case gain:

$$\max_{i\in[m]} \min_{j\in[n]} a_{ij}.$$

Similarly, Player II picks the j-th column, aiming to minimize her worst-case loss:

$$\min_{j\in[n]} \max_{i\in[m]} a_{ij}.$$

In general, these two quantities are not equal, and we always have:

$$\text{Player I's gain} = \max_{i\in[m]} \min_{j\in[n]} a_{ij} \leqslant \min_{j\in[n]} \max_{i\in[m]} a_{ij} = \text{Player II's loss}.$$

This inequality is unsatisfactory for both players. Instead, players should play probabilistically.

Definition 12.1 A strategy in which action is selected with some probability is called a *mixed strategy*.

We denote by $x \in \Delta^m$ the strategy of Player I and by $y \in \Delta^n$ the strategy of Player II. That is

$$\mathbf{P}[\text{pick } i\text{-th row}] = x_i \quad \text{and} \quad \mathbf{P}[\text{pick } j\text{-th column}] = y_j.$$

A strategy, where some action is played with probability 1 is called a *pure strategy*; in other words these are orthonormal vectors $(e_i)_{i=1}^m$ and $(e_j)_{j=1}^n$.

With such randomization, it is easy to see that the expected gain of Player I and the expected loss of Player II is

$$x^\top A y = \sum_{i\in[m]} \sum_{j\in[n]} x_i a_{ij} y_j.$$

If Player I uses strategy x, then he can guarantee (in average)

$$\min_{y\in\Delta^n} x^\top A y = \min_j (A^\top x)_j. \tag{12.1}$$

Hence, he wants to choose x to maximize his gain, that is

$$\max_{x\in\Delta^m} \min_{y\in\Delta^n} x^\top A y.$$

Similarly, Player II wants to minimize her loss, that is

$$\min_{y\in\Delta^n} \max_{x\in\Delta^m} x^\top A y.$$

The following result signifies importance of probabilistic strategies.

Theorem 12.1 von Neumann's Minimax theorem. For any $m \times n$ payoff matrix A,

$$\max_{x\in\Delta^m} \min_{y\in\Delta^n} x^\top A y = \min_{y\in\Delta^n} \max_{x\in\Delta^m} x^\top A y =: v.$$

The number v is called the value of the game.

Definition 12.2 A strategy (i^*, j^*) such that $\max_i a_{ij^*} = a_{i^*j^*} = \min_j a_{i^*j}$ is calles a *pure Nash equilibrium* (which may not exist). A probabilistic strategy (x^*, y^*) such that

$$\min_{y\in\Delta^n} \langle x^*, Ay\rangle = \langle x^*, Ay^*\rangle = \max_{x\in\Delta^m} \langle x, Ay^*\rangle$$

is called a *Nash equilibrium*. Both (i^*, j^*) and (x^*, y^*) are also called *saddle point*.

As a consequence of (x^*, y^*) being a saddle point, we immediately have

$$\min_{x\in\Delta^m} \langle x, Ay^*\rangle \leqslant \langle x^*, Ay^*\rangle \leqslant \max_{y\in\Delta^n} \langle x^*, Ay\rangle. \tag{12.2}$$

12.3 Solving minimax

Now the question is how to find a Nash equilibrium? In orther words, we would like to solve

$$\min_{y\in\Delta^n} \max_{x\in\Delta^m} \langle x, Ay\rangle.$$

We will show how MWU/Hedge algorithms can be applied to this problem. Let Player II be our player in the Hedge algorithm, where the columns she chooses are our "experts." Player II wants to understand which columns perform well and which do not when Player I plays optimally. For each strategy $y_t \in \Delta^n$ that Player II adopts, Player I's best response is $x_t = \arg\max_{x\in\Delta^m} \langle x, Ay_t\rangle = e_{i_t}$, where $i_t = \arg\max_{i\in[m]} (Ay_t)_i$. Thus, for each $j \in [n]$ drawn from y_t, Player II has to pay Player I $a_{i_t j} = (A^\top x_t)_j$. Consequently, it is natural to consider the loss vector $\ell_t = A^\top x_t$. Over time, this allows Player II to identify which columns are "good" and which are "bad."

Algorithm 4 Multiplicative Weights Update/Hedge for zero-sum games

Initialization: Fix $\alpha \in \left(0, \frac{1}{2}\right]$ and $w_1(j) = 1$ for every $j \in [n]$.
for $t = 1, 2 \ldots, T$ **do**
1. Sample a decision j from the probability distribution $y_t = \frac{w_t}{\|w_t\|_1}$.
2. Observe the loss $\ell_t = A^\top x_t$, where

$$x_t = \operatorname*{argmax}_{x\in\Delta^m} \langle x, Ay_t\rangle = \operatorname*{argmax}_{i\in[m]} (Ay_t)_i.$$

3. Update weights: $w_{t+1}(j) = w_t(j) e^{-\alpha \ell_t(j)}$ for all $j \in [n]$.

end for

Theorem 12.2 Let A be a $m \times n$ payoff matrix, (x^*, y^*) be its saddle point, and $\varepsilon > 0$. Suppose that $T = \frac{4\|A\|_\infty^2 \log n}{\varepsilon^2}$ and $\alpha = \sqrt{\frac{\log n}{T}} = \frac{\varepsilon}{2\|A\|_\infty}$. Then Algorithm 4 after T iterations approximates the value of the game as

$$\langle x^*, Ay^*\rangle \leqslant \frac{1}{T}\sum_{t=1}^{T} \langle x_t, Ay_t\rangle \leqslant \langle x^*, Ay^*\rangle + \varepsilon.$$

Moreover, the strategy $\bar{y}_T = \frac{1}{T}\sum_{t=1}^{T} y_t$ is ε-optimal strategy for Player II

$$\langle x^*, Ay^*\rangle \leqslant \langle x^*, A\bar{y}_T\rangle \leqslant \langle x^*, Ay^*\rangle + \varepsilon.$$

Proof. We want to apply the bound from Theorem 11.2, but for this we had to assume that $\ell_t(j) \in [-1,1]$. Since ℓ_t is just a row of A, we could redefine our losses in the algorithm by $\frac{\ell_t}{\|A\|_\infty}$. This is just a scalar that doesn't change the behavior of the algorithm. Hence, we have

$$\begin{aligned}\sum_{t=1}^T \frac{\ell_t(y_t)}{\|A\|_\infty} - \sum_{t=1}^T \frac{\ell_t(y^*)}{\|A\|_\infty} &\leqslant \frac{\log n}{\alpha} + \frac{\alpha}{\|A\|_\infty^2} \sum_{t=1}^T \langle \ell_t^2, y_t\rangle \\ &\leqslant \frac{\log n}{\alpha} + \frac{\alpha}{\|A\|_\infty^2} \sum_{t=1}^T \|\ell_t\|_\infty^2 \|y_t\|_1 \\ &\leqslant \frac{\log n}{\alpha} + \alpha T.\end{aligned}$$

Hence, if we use $\alpha = \sqrt{\frac{\log n}{T}}$, we obtain

$$\sum_{t=1}^T \ell_t(y_t) - \sum_{t=1}^T \ell_t(y^*) \leqslant 2\sqrt{T \log n}\|A\|_\infty.$$

Dividing both sides over T yields

$$\frac{1}{T}\sum_{t=1}^T \ell_t(y_t) - \frac{1}{T}\sum_{t=1}^T \ell_t(y^*) \leqslant 2\sqrt{\frac{\log n}{T}}\|A\|_\infty = \varepsilon.$$

Now notice that $\ell_t(y^*) = \langle x_t, Ay^*\rangle \leqslant \langle Ax^*, y^*\rangle$ by (12.2). Also,

$$\ell_t(y_t) = \langle x_t, Ay_t\rangle \geqslant \langle x^*, Ay_t\rangle \geqslant \langle x^*, Ay^*\rangle.$$

Combining them, we get

$$\langle x^*, Ay^*\rangle \leqslant \frac{1}{T}\sum_{t=1}^T \langle x^*, Ay_t\rangle \leqslant \frac{1}{T}\sum_{t=1}^T \ell_t(y_t) = \frac{1}{T}\sum_{t=1}^T \langle x_t, Ay_t\rangle \leqslant \langle x^*, Ay^*\rangle + \varepsilon,$$

which is the desired inequality. Note that for Player II, $\bar{y}_T = \frac{1}{T}\sum_{t=1}^T$ is indeed an ε-optimal strategy, since

$$\langle x^*, Ay^*\rangle \leqslant \frac{1}{T}\sum_{t=1}^T \langle x^*, Ay_t\rangle = \langle x^*, A\bar{y}_T\rangle \leqslant \langle x^*, Ay^*\rangle + \varepsilon.$$

■

12.3.1 Subgradient method perspective

Notice that our problem can be cast as just a minimization problem $\min_{y\in\Delta^n} \varphi(y)$, with

$$\varphi(y) = \max_{x\in\Delta^m} \langle x, Ay\rangle.$$

The function φ is convex as the maximum of convex functions, but obviously not differentiable. Its subdifferential can be computed in the standard way

$$\partial\varphi(y) = \left\{A^\top x : x \in \operatorname*{argmax}_{x\in\Delta^m} \langle x, Ay\rangle\right\}.$$

To summarize, we have a constrained problem with a convex function φ, whose subgradints are easy to compute. We have all the ingredients to apply the projected subgradient method[1]

$$\begin{aligned} x_k &= \operatorname*{argmax}_{x\in\Delta^m} \langle x, Ay_k\rangle \\ y_{k+1} &= P_{\Delta^n}(y_k - \alpha A^\top x_k) \qquad\qquad //\text{since } \widetilde{\nabla}\varphi(y_k) = A^\top x_k. \end{aligned}$$

[1] For simplicity, we choose any subgradient $A^\top x_k$.

For simplicity, we used above a fixed stepsize α. There is no explicit formula for the projection onto the simplex Δ^n, but the algorithm that computes it is quite efficient with complexity $O(n \log n)$.

12.3.2 Mirror descent perspective

Since our minimization problem is constrained in the unit simplex, it is natural to apply the mirror subgradient method for ℓ_1-metric (with the kernel map being a negentropy). This yields the following update

$$\begin{aligned} x_k &= \operatorname*{argmax}_{x \in \Delta^m} \langle x, A y_k \rangle \\ w_{k+1} &= y_k \circ e^{-\alpha A^\top x_k} && \text{//since } \widetilde{\nabla} \varphi(y_k) = A^\top x_k \\ y_{k+1} &= \frac{w_{k+1}}{\|w_{k+1}\|_1}. \end{aligned}$$

Comparing it with Algorithm 4, we see they are almost identical (the difference in indices k and t is only cosmetic). The more substantial difference is that mirror descent uses $w_{k+1} = y_k \circ e^{-\alpha \ell_k}$ instead of $w_{k+1} = w_k \circ e^{-\alpha \ell_k}$. However, this shouldn't make any difference for the Hedge algorithm, since y_k is just a scaling of w_k, and both updates lead to the same y_k.

Comments

Exercise 12.1 Explain equality sign in (12.1). ∎

Lecture 13: Stochastic Mirror descent for zero-sum games

R In this lecture we swap x and y, to make it consistent with existing literature. All results from the previous lecture hold subject to this change.

13.1 Zero-sum games

Previously, we saw that the Hedge algorithm (or subgradient method) has a convergence rate of $O\left(\frac{1}{\varepsilon^2}\right)$ with a cost of $O(mn)$ per iteration, resulting in a total complexity of $O\left(\frac{mn}{\varepsilon^2}\right)$. There are two ways to improve this complexity:

1. Achieve an ε^{-1} dependency instead of ε^{-2}.
2. Maintain ε^{-2} but reduce the iteration cost $O(mn)$.

Today, we will consider the latter. Before proceeding, let's swap x and y from the previous lecture, as it is typical to solve minimization problems over x and maximization problems over y. Hence, our problem now is:

$$\min_{x\in\Delta^n}\max_{y\in\Delta^m}\langle Ax,y\rangle$$

and for every saddle point (x^*,y^*), it holds that:

$$\min_{x\in\Delta^n}\langle Ax,y^*\rangle=\min_{x\in\Delta^n}\max_{y\in\Delta^m}\langle Ax,y\rangle=\max_{y\in\Delta^m}\langle Ax^*,y\rangle=\langle Ax^*,y^*\rangle. \tag{13.1}$$

Moreover, if for given x,y one has that

$$\max_{y'\in\Delta^m}\langle Ax,y'\rangle=v,$$

where v is the value of the game, then then we may conclude that x is a primal solution and similarly for y.

Definition 13.1 Given a pair (x,y), the *primal-dual gap* is defined as

$$\mathrm{PD}(x,y)=\max_{y'\in\Delta^m}\langle Ax,y'\rangle-\min_{x'\in\Delta^n}\langle Ax',y\rangle.$$

Lemma 13.1 For every feasible (x,y), it holds that $\mathrm{PD}(x,y)\geqslant 0$ and $\mathrm{PD}(x,y)=0$ if and only if (x,y) is a saddle point.

Proof. Nonnegativity follows from

$$\mathrm{PD}(x,y)=\underbrace{\max_{y'\in\Delta^m}\langle Ax,y'\rangle-\langle Ax,y\rangle}_{\geqslant 0}+\underbrace{\langle Ax,y\rangle-\min_{x'\in\Delta^n}\langle Ax',y\rangle}_{\geqslant 0}.$$

Now if $\mathrm{PD}(x,y)=0$, then from above it follows that

$$\max_{y'\in\Delta^m}\langle Ax,y'\rangle=\langle Ax,y\rangle=\min_{x'\in\Delta^n}\langle Ax',y\rangle$$

or, equivalently,

$$\langle Ax,y'\rangle\leqslant\langle Ax,y\rangle\leqslant\langle Ax',y\rangle\qquad\forall(x',y')\in\Delta^n\times\Delta^m,$$

which by definition implies that (x,y) is a saddle point. ■

Consider (13.1) again. Since we know that x^*,y^* are solutions of the mostleft and mostright problems there, we can write down optimality conditions:

$$\begin{aligned}\langle A^\top y^*,x-x^*\rangle&\geqslant 0\quad\forall x\in\Delta^n,\\ -\langle Ax^*,y-y^*\rangle&\geqslant 0\quad\forall y\in\Delta^m.\end{aligned}$$

These two inequalities are completely independent, so their system is equivalent to their sum, which we cast in a matrix form

$$\left\langle\begin{bmatrix}0&A^\top\\-A&0\end{bmatrix}\begin{pmatrix}x^*\\y^*\end{pmatrix},\begin{pmatrix}x\\y\end{pmatrix}-\begin{pmatrix}x^*\\y^*\end{pmatrix}\right\rangle\geqslant 0\qquad\forall(x,y)\in\Delta^n\times\Delta^m.$$

All these expression are a bit cumbersome to write, so after introducing some notations

$$z=\begin{pmatrix}x\\y\end{pmatrix},\quad z^*=\begin{pmatrix}x^*\\y^*\end{pmatrix},\quad F=\begin{bmatrix}0&A^\top\\-A&0\end{bmatrix},\quad C=\Delta^n\times\Delta^m.$$

we can rewrite it as

$$\langle F(z^*),z-z^*\rangle\geqslant 0\quad\forall z\in C.\tag{13.2}$$

This is called a *variational inequality*. We have already met this condition when studied optimality condition for $\min_{x\in C}f(x)$ — there our operator F was just the gradient ∇f. This time F is not a gradient of any function, still the optimality condition is very much the same.

13.2 Mirror descent for variational inequalities

Recalling the projected gradient method, it is tempting to consider the following scheme

$$z^{k+1}=P_C(z^k-\alpha_kF(z^k)).$$

or even more generally, the mirror descent method

$$z^{k+1}=\operatorname*{argmin}_{z\in C}\left\{\langle F(z^k),z-z_k\rangle+\frac{1}{\alpha_k}D(z,z^k)\right\},$$

where D is some Bregman distance. Despite F not being the gradient, we could use the same arguments as we did before in Lecture 9. Repeating the derivation from (9.5), we obtain

$$D(z,z^{k+1})+D(z^{k+1},z^k)\leqslant D(z,z^k)+\alpha_k\langle F(z^k),z-z^k\rangle+\frac{\alpha_k^2}{2}\|F(z^k)\|_*^2+\frac{1}{2}\|z^{k+1}-z^k\|^2.$$

And after using $D(z^{k+1},z^k)\geqslant\frac{1}{2}\|z^{k+1}-z^k\|^2$ (the consequence of strong convexity), we have

$$D(z,z^{k+1})\leqslant D(z,z^k)+\alpha_k\langle F(z^k),z-z^k\rangle+\frac{\alpha_k^2}{2}\|F(z^k)\|_*^2.$$

Before, $\langle F(z^k), z - z^k\rangle$ was handled by convexity. This time, notice that

$$\langle F(z^k), z^k - z\rangle = \langle Ax^k, y\rangle - \langle x, A^\top y^k\rangle$$

and

$$\max_{z\in C}\langle F(z^k), z^k - z\rangle = \max_{y\in\Delta^m}\langle Ax^k, y\rangle - \min_{x\in\Delta^n}\langle x, A^\top y^k\rangle = \mathrm{PD}(x^k, y^k),$$

which can be also served as an optimality measure, similar to $f(x) - f_*$ for minimization problems. We won't finish the derivation for the deterministic case, as we'll consider a more general stochastic case.

All of the above can also be applied with a stochastic unbiased estimator F_ξ of F. Suppose we have access to F_ξ such that $\mathbf{E}\left[F_\xi(z)\right] = F(z)$. We can similarly consider a stochastic mirror descent:

$$z^{k+1} = \operatorname*{argmin}_{z\in C}\left\{\langle F_\xi(z^k), z - z_k\rangle + \frac{1}{\alpha_k}D(z, z^k)\right\}.$$

The corresponding inequality will now be:

$$\alpha_k\langle F_\xi(z^k), z^k - z\rangle \leqslant D(z, z^k) - D(z, z^{k+1}) + \frac{\alpha_k^2}{2}\|F_\xi(z^k)\|_*^2. \tag{13.3}$$

Theorem 13.1 Let F_ξ be an unbiased stochastic estimator of F, with $\mathbf{E}\left[\|F_\xi(z)\|^2\right] \leqslant G^2$ for all z. Let $R^2 \geqslant D(z, z^1)$ for all $z \in C$ and let $\bar{z}^K = (\bar{x}^K, \bar{y}^K) = \frac{1}{A_K}\sum_{k=1}^K \alpha_k z^k$ and $A_K = \sum_{k=1}^K \alpha_k$. Then the following inequality holds

$$\mathrm{PD}(\mathbf{E}\left[\bar{x}^K\right], \mathbf{E}\left[\bar{y}^K\right]) = \max_{z\in C}\mathbf{E}\left[\langle F(\bar{z}^k), \bar{z}^k - z\rangle\right] \leqslant \frac{R^2}{A_K} + \frac{G^2\sum_{k=1}^K \alpha_k^2}{2A_K}. \tag{13.4}$$

Proof. We take a conditional expectation in (13.3) to get

$$\alpha_k\langle F(z^k), z^k - z\rangle \leqslant D(z, z^k) - \mathbf{E}_k\left[D(z, z^{k+1})\right] + \frac{\alpha_k^2}{2}\mathbf{E}_k\left[\|F(z^k)\|_*^2\right].$$

After taking the total expectation and telescoping we obtain

$$\mathbf{E}\left[\sum_{k=1}^K \alpha_k\langle F(z^k), z^k - z\rangle\right] \leqslant D(z, z^1) + \frac{1}{2}\sum_{k=1}^K \alpha_k^2\mathbf{E}_k\left[\|F(z^k)\|_*^2\right].$$

Using

$$\mathbf{E}\left[\sum_{k=1}^K \alpha_k\left(\langle Ax^k, y\rangle - \langle x, A^\top y^k\rangle\right)\right] \leqslant R^2 + \frac{1}{2}\sum_{k=1}^K \alpha_k^2 G^2.$$

By linearity of expectation, we have

$$\langle A\mathbf{E}\left[\bar{x}^K\right], y\rangle - \langle x, A^\top\mathbf{E}\left[\bar{y}^K\right]\rangle \leqslant \frac{R^2}{A_K} + \frac{1}{2A_K}\sum_{k=1}^K \alpha_k^2 G^2.$$

By maximizing the LHS over all $(x, y) \in C$, we derive the desired inequality. ∎

R The quantity for which we obtain the convergence rate, $\mathrm{PD}(\mathbf{E}\left[\bar{x}^K\right], \mathbf{E}\left[\bar{y}^K\right])$ is weaker than the more standard $\mathbf{E}\left[\mathrm{PD}(\bar{x}^K, \bar{y}^K)\right]$ (similarly, as $f(\mathbf{E}\left[\bar{x}^K\right]) - f_*$ is weaker than $\mathbf{E}\left[f(\bar{x}^k) - f_*\right]$). With a bit more work the same rate can be also shown for $\mathbf{E}\left[\mathrm{PD}(\bar{x}^K, \bar{y}^K)\right]$, although with some additional multiplicative factors.

Notice that the RHS in (13.4) is exactly the same as in Theorem 9.1. This shouldn't be surprising, since we were using essentially the same bounds. Therefore, with our standard choice of stepsizes α_k we can get the usual $O(\frac{RG}{\sqrt{K}})$ convergence rate.

Now we would like to be more specific: we have to fix setting where we can calculate R and G and devise a stochastic estimator F_ξ. In general we need the following:

- Primal x-space $(\mathbb{R}^n, \|\cdot\|)$, dual x-space $(\mathbb{R}^n, \|\cdot\|_\infty)$.
- Primal y-space $(\mathbb{R}^m, \|\cdot\|)$[1], dual y-space $(\mathbb{R}^m, \|\cdot\|_\infty)$.
- Fix mirror functions $h_1 : \mathbb{R}^n \to (-\infty, +\infty]$, $h_2 : \mathbb{R}^m \to (-\infty, +\infty]$ with the usual good properties.
- On $\mathbb{R}^n \times \mathbb{R}^m$ we define the primal norm as $\|z\| = (\|x\|^2 + \|y\|^2)^{1/2}$. It induces the dual norm $\|w\|_* = \|(u,v)\|_* = (\|u\|_*^2 + \|v\|_*^2)^{1/2}$. We set $h(z) = h_1(x) + h_2(y)$ that defines $D(z,z') = D_1(x,x') + D_2(y,y')$. If h_1 and h_2 are 1-strongly convex wrt their primal norms, then $D(z,z') \geqslant \frac{1}{2}\|z - z'\|^2$.

▪ **Example 13.1 — ℓ_2/ℓ_2 case.** We consider $\mathbb{R}^n \times \mathbb{R}^m$ with the primal norm $\|z\| = (\|x\|_2^2 + \|y\|_2^2)^{1/2}$, which is the usual ℓ_2-norm on $\mathbb{R}^n \times \mathbb{R}^m$ and the dual is the same. Let $h(z) = \frac{1}{2}\|x\|_2^2 + \frac{1}{2}\|y\|_2^2$ and $D(z,z') = \frac{1}{2}\|x - x'\|_2^2 + \frac{1}{2}\|x - x'\|_2^2$. Since, $x, x' \in \Delta^n$ and $y, y' \in \Delta^m$, we have $R^2 = 4$ or $R = 2$. ▪

▪ **Example 13.2 — ℓ_1/ℓ_1 case.** We consider $\mathbb{R}^n \times \mathbb{R}^m$ with the primal norm $\|z\| = (\|x\|_1^2 + \|y\|_1^2)^{1/2}$ and the correspondent dual norm $\|w\|_* = (\|x\|_\infty^2 + \|y\|_\infty^2)^{1/2}$. Let $h_1(x) = \langle x, \log x\rangle$ for $x \in \mathbb{R}^n_+$ and $+\infty$ otherwise. Similarly, for $h_2(y)$. We set $h(z) = h_1(x) + h_2(y)$ and define $D(z,z') = D_1(x,x') + D_2(y,y')$. As we have already computed for mirror descent before, if we use $z^1 = (\frac{1}{n}\mathbf{1}_n, \frac{1}{m}\mathbf{1}_m)$ we have $R^2 = \log n + \log m = \log mn$. ▪

Now we have to come up with a stochastic oracle F_ξ that has a relatively low cost and estimate $\mathbf{E}\left[\|F_\xi(z)\|_*^2\right]$ for each case.

13.2.1 Stochastic oracle F_ξ

Since computing $F(z)$ basically involves two matrix-vector multiplications, we have to know how to estimate such operations stochastically.

Let A_ξ denotes the stochastic oracle for A, that is $\mathbf{E}\left[A_\xi x\right] = Ax$. Since $Ax = A_{:1}x_1 + \cdots + A_{:n}x_n$, it makes sense to consider

$$A_\xi x = \frac{1}{p_j} A_{:j} x_j, \qquad \mathbf{P}[\xi = j] = p_j.$$

Obviously, this has a cost that is m times smaller than Ax.

Doing the same for $A^\top$, we end up with the following stochastic oracle for F:

$$F_\xi(x,y) = \begin{pmatrix} \frac{1}{q_i} A_{i:} y_i \\ \frac{1}{p_j} A_{:j} x_j \end{pmatrix}, \qquad \mathbf{P}[\xi = (i,j)] = p_j q_i,$$

where we sample i and j independently. Now, based on the dual norm $\|\cdot\|_*$, we need to find distribution p and q, so that inequality $\mathbf{E}\left[\|F_\xi(z)\|_*^2\right] \leqslant G^2$ holds for all $z \in C$.

ℓ_1-geometry. In this case, we are interested in bounding $\mathbf{E}\left[\|F_\xi(z)\|_*^2\right] = \mathbf{E}\left[\|A_{\xi_2} x\|_\infty^2 + \|A_{\xi_1}^\top y\|_\infty^2\right]$.

Let us choose $p_j = x_j$ and $q_i = y_i$. In other words,

$$A_{\xi_2} x = A_{:j}, \qquad \mathbf{P}[\xi_2 = j] = x_j$$

[1] There shouldn't be a confusion with x-space, since they have different dimensions.

and similarly for y. We have

$$\mathbf{E}\left[\|F_\xi(z)\|_*^2\right] = \mathbf{E}\left[\|A_{\xi_1:}\|_\infty^2 + \|A_{:\xi_2}\|_\infty^2\right] \leqslant 2\|A\|_\infty^2 =: G^2,$$

since for every i and j, $\|A_{i:}\|_\infty \leqslant \|A\|_\infty$ and $\|A_{:j}\|_\infty \leqslant \|A\|_\infty$. Here $\|A\|_\infty = \max_{i,j}|A_{ij}|$.

ℓ_2-geometry. We want to bound

$$\mathbf{E}\left[\|F_\xi(z)\|_*^2\right] = \mathbf{E}\left[\|A_{\xi_1:}y\|_2^2 + \|A_{:\xi_2}x\|_2^2\right].$$

Let us choose $p_j = \frac{x_j^2}{\|x\|_2^2}$ and $q_i = \frac{y_i^2}{\|y\|_2^2}$. In other words,

$$A_{\xi_2}x = \frac{1}{p_j}A_{:j}x_j, \qquad \mathbf{P}[\xi_2 = j] = p_j$$

and similarly for y. In particular, we have

$$\mathbf{E}\left[\|A_{:\xi_2}x\|_2^2\right] = \sum_{j=1}^n \frac{x_j^2}{p_j}\|A_{:j}\|_2^2 = \|x\|_2^2\|A\|_F^2 \leqslant \|A\|_F^2,$$

where we used that $\|x\|_2^2 \leqslant \|x\|_1^2 = 1$. The estimate for $\mathbf{E}\left[\|A_{\xi_1:}y\|_2^2\right]$ is similar. Hence, we can conclude that

$$\mathbf{E}\left[\|F_\xi(z)\|_*^2\right] = \mathbf{E}\left[\|A_{\xi_1:}y\|_2^2 + \|A_{:\xi_2}x\|_2^2\right] \leqslant 2\|A\|_F^2 =: G^2.$$

13.2.2 Comparison

We can summarize the obtained results in the following table

	ℓ_2/ℓ_2-geometry	ℓ_1/ℓ_1-geometry
Convergence rate	$O\left(\frac{\|A\|_F^2}{\sqrt{K}}\right)$	$O\left(\frac{\|A\|_\infty^2 \log mn}{\sqrt{K}}\right)$

It is clear that in most cases, convergence in the ℓ_1/ℓ_1-case is much better (log gives a small constant and typically $\|A\|_\infty \ll \|A\|_F$). However, remember that this is a worst-case analysis and may not always reflect the behavior in specific instances.

Now, the complexity of both methods must take into account an additional factor $O(m+n)$ (vector-vector operations). This contrasts with deterministic algorithms that have the same convergence rate but with a cost per iteration of $O(mn)$ (matrix-vector multiplications).

Comments

References

Lecture 14: MAX-CUT

14.1 Introduction

Let $G = (V, E)$ be a finite simple graph (no loops and double edges).

Definition 14.1 Given a partition of V into two disjoint sets, the *cut* is called the number of edges crossing between these two sets. The *maximum cut*, $\mathrm{MAXCUT}(G)$, is the cut with the maximum number of edges.

We are interested in approximating the value $\mathrm{MAXCUT}(G)$; computing it exactly is NP-hard.

Let $n = |V|$ and label vertices $1, \dots, n$. The *adjacency matrix* is an $n \times n$ matrix A with entries $a_{ij} = 1$ if vertices i and j are connected and $a_{ij} = 0$ otherwise.

Every partition of V can be encoded by a binary vector $x = (x_1, \dots, x_n)$ with $x_i \in \{-1, 1\}$; the sign of x_i indicates to which subset of vertices i belongs. Given such x, we write $\mathrm{CUT}(G, x)$ to denote the corresponding cut. We have the following:

$$\mathrm{CUT}(G, x) = \frac{1}{2} \sum_{i,j : x_i x_j = -1} a_{ij} = \frac{1}{4} \sum_{i,j=1}^{n} a_{ij}(1 - x_i x_j). \tag{14.1}$$

And hence,

$$\mathrm{MAXCUT}(G) = \frac{1}{4} \max \left\{ \sum_{i,j=1}^{n} a_{ij}(1 - x_i x_j) \colon x_i = \pm 1 \right\}. \tag{14.2}$$

Lemma 14.1 0.5-approximation of MAX-CUT. Let us partiton V into two sets uniformly at random (among all 2^n subsets) and let x be a correspondent binary encoding of that partition. Then

$$\mathbf{E}[\mathrm{CUT}(G, x)] \geqslant \frac{1}{2}|E| \geqslant \frac{1}{2}\mathrm{MAXCUT}(G).$$

Proof. The uniform partition means that $x \sim \mathrm{Unif}(\{-1, 1\}^n)$ with independent entries. Thus, if $i \neq j$, then $\mathbf{E}\left[x_i x_j\right] = 0$. Conversely, if $i = j$, then $a_{ij} = 0$. Altogether, we conclude

$$\mathbf{E}[\mathrm{CUT}(G, x)] = \frac{1}{4} \sum_{i,j=1}^{n} a_{ij}(1 - \mathbf{E}\left[x_i x_j\right]) = \frac{1}{2}|E| \geqslant \frac{1}{2}\mathrm{MAXCUT}(G).$$

■

14.1.1 Interlude: linear and semidefinite programming

Linear programming. The most basic class of optimization problems is called a *linear programming*. These are problems whose both objective and constraints are affine/linear:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \ & \langle c, x\rangle \\ \text{subject to } & \langle a_i, x\rangle \leqslant b_i, \quad i = 1,\dots,m \\ & x \geqslant 0. \end{aligned}$$

This class of problems is well-understood and we have efficient algorithms for solving them, such as the simplex method and interior point methods.

Semidefinite programming. Similarly to linear programming, we can consider the problem when X is a matrix. In this case, we use the usual inner product for matrices, $\langle A, B\rangle = \operatorname{tr}(A^\top B)$. This yields a semidefinite program:

$$\begin{aligned} \min_{X\in\mathbb{R}^{n\times n}} \ & \langle C, X\rangle \\ \text{subject to } & \langle A_i, X\rangle \leqslant b_i, \quad i = 1,\dots,m \\ & X \succcurlyeq 0. \end{aligned}$$

Again similarly to linear programming, semidefinite problems can be efficiently solved by interior point methods[1]

14.2 Semidefinite relaxation

Problem (14.2) is a discrete optimization problem. As is often the case with discrete optimization, we will relax the problem to make it continuous and then apply methods from continuous optimization to approximate its solution.

The idea of a relaxation is to use vectors $X_i \in \mathbb{R}^n$ instead of numbers x_i. Since before we had that $|x_i| = 1$, we will assume that $\|X_i\|_2 = 1$. This yields

$$\operatorname{SDP}(G) = \frac{1}{4}\max\left\{\sum_{i,j=1}^{n} a_{ij}(1 - \langle X_i, X_j\rangle) \colon \|X_i\|_2 = 1\right\}. \tag{14.3}$$

We immediately have that

$$\operatorname{MAXCUT}(G) \leqslant \operatorname{SDP}(G).$$

Exercise 14.1 Explain why the inequality above holds true. ∎

Let us explain why problem (14.3) is indeed a semidefinite program. Let X be an $n \times n$ matrix with $X_{ij} = \langle X_i, X_j\rangle$, such matrix is called the Gram matrix of vectors $X_1, \dots, X_n$. We know that $X \succcurlyeq 0$ and that $X_{ii} = 1$. Hence, (14.3) can be reformulated as

$$\begin{aligned} \max_{X\in\mathbb{R}^{n\times n}} \ & -\frac{1}{4}\langle A, X\rangle + \frac{1}{2}|E| \\ \text{subject to } & X_{ii} = 1, \quad i = 1,\dots,n \\ & X \succcurlyeq 0. \end{aligned}$$

It is clear that $X_{ii} = 1$ are linear constraints (if not, show why). Hence, (14.3) is a semidefinite problem and thus can be solved efficiently.

[1] As always, efficient may change its meaning when dimension n increases.

Alternatively, one can come up with the same relaxation using the following arguments. Problem (14.2) can be rewritten as

$$\mathrm{MAXCUT}(G) = \frac{1}{4}\max\left\{\sum_{i,j=1}^{n} a_{ij}(1-x_i x_j)\colon x_i = \pm 1\right\} = \frac{1}{4}\max\left\{2|E| - \langle Ax, x\rangle\colon x_i = \pm 1\right\}.$$

Thus, ultimately, we are interested in minimizing $\langle Ax, x\rangle$. Note that

$$\langle Ax, x\rangle = \langle A, xx^\top\rangle.$$

What can we say about $xx^\top$? It is a symmetric positive semidefinite matrix with all ones on the main diagonal and, importantly, it is a rank-one matrix. The latter is a difficult constraint, so we relax it to all positive semidefinite matrices whose diagonal elements are all ones. This yields the same semidefinite program as above.

Now, given a solution X to $\mathrm{SDP}(G)$, how do we recover the actual partition x, and what can we say about its cut? Recovery is done by the following procedure:

Randomized rounding:

1. Solve $\mathrm{SDP}(G)$ and let X be its solution.
2. Find $M = [X_1|\ldots|X_n]$ such that $MM^\top = X$ (Cholesky's decomposition).
3. Generate a random vector $g \sim N(0, I_n)$.
4. Set $x_i = \operatorname{sign}\langle X_i, g\rangle$ for $i \in [n]$.

In other words, we generate a random hyperplane whose normal is g. All vectors X_i above it and below define respectively two subsets of the partition.

Finally, the main result says that the SDP relaxation provides a significantly better approximation than the previous 0.5-approximation.

Theorem 14.1 **0.878-approximation algorithm for maximum cut.** Let $x = (x_i)$ be the result of a randomized rounding of the solution of the semidefinite program (14.3). Then

$$\mathbf{E}[\mathrm{CUT}(G, x)] \geqslant 0.878 \cdot \mathrm{SDP}(G) \geqslant 0.878 \cdot \mathrm{MAXCUT}(G).$$

The second inequality is obvious.

Lemma 14.2 **Grothendieck's identity.** Let $g \sim N(0, I_n)$. Then for any fixed u, v, we have

$$\mathbf{E}[\operatorname{sign}\langle u, g\rangle \operatorname{sign}\langle v, g\rangle] = \frac{2}{\pi}\arcsin\langle u, v\rangle.$$

Proof. Recall that a Gaussian distribution is invariant under rotation: g and Ug have the same distribution for any orthogonal matrix U. Using this fact, we can reduce the problem to g lying in the same plane as u and v, with its angle uniformly distributed. This simplifies the problem to 2D. ■

In the previous lemma, the function arcsin is difficult to work with, so instead we use the following bound.

Lemma 14.3 For every $t \in [-1, 1]$ it holds

$$1 - \frac{2}{\pi}\arcsin t = \frac{2}{\pi}\arccos t \geqslant 0.878(1-t).$$

Proof of Theorem 14.1. First, we have

$$\mathbf{E}[\mathrm{CUT}(G,x)] = \frac{1}{4}\sum_{i,j=1}^{n} a_{ij}(1-\mathbf{E}\big[x_i x_j\big]).$$

By the definition of x_i, followed by Lemma 14.2 and 14.3, we have

$$1-\mathbf{E}\big[x_i x_j\big] = 1-\mathbf{E}\big[\operatorname{sign}\langle X_i, g\rangle \operatorname{sign}\langle X_j, g\rangle\big] = 1-\frac{2}{\pi}\arcsin\langle X_i, X_j\rangle \geqslant 0.878\cdot(1-\langle X_i, X_j\rangle).$$

Hence,

$$\mathbf{E}[\mathrm{CUT}(G,x)] \geqslant 0.878\cdot\frac{1}{4}\sum_{i,j=1}^{n} a_{ij}(1-\langle X_i, X_j\rangle) = 0.878\cdot\mathrm{SDP}(G).$$

■

Comments

This result was proposed by Goemans and Williamson in [1]. I mostly followed the exposition in [2].

References

[1] Michel X Goemans and David P Williamson. "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming". In: *Journal of the ACM (JACM)* 42.6 (1995), pages 1115–1145.

[2] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*. Volume 47. Cambridge university press, 2018.

Lecture 15: Randomized Power Method

15.1 Power method

Let A be an $n \times n$ symmetric positive semidefinite matrix with eigenvalues $\lambda_1 \geqslant \ldots \geqslant \lambda_n \geqslant 0$. We would like to estimate λ_1. One of the most basic methods for this is the power method:

Algorithm 5 Power Method

Initialization: $y_0 = w \in \mathbb{R}^n$.
for $k = 1, 2 \ldots,$ **do**
  $y_k = \frac{Ay_{k-1}}{\|Ay_{k-1}\|}$.
  $\xi_k = \langle y_k, Ay_k \rangle$
end for

Clearly, we have that $\xi_k \in [0, \lambda_1]$. We will need the notion of the *relative error*

$$\mathrm{err}(\xi_k) = \frac{\lambda_1 - \xi_k}{\lambda_1} \in [0, 1].$$

For simplicity, we assume throughout the lecture the following.

Assumptions:
1. A is a diagonal matrix $A = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$.
2. $\lambda_1 = 1$.

The first assumption, while it may seem far-fetched, is achieved without loss of generality by eigendecomposition $A = U\Lambda U^\top$, with orthogonal U and diagonal Λ. This means we are merely changing the basis in which we work. Note that the algorithm won't have access to this basis; it is used only to simplify the analysis. The second assumption is easily satisfied by rescaling the matrix.

15.1.1 Relative error

Using that $\lambda_1 = 1$, we have

$$\begin{aligned}\operatorname{err}(\xi_k) = 1 - \langle y_k, Ay_k\rangle &= 1 - \frac{\langle w, A^{2k+1}w\rangle}{\langle w, A^{2k}w\rangle} = \frac{\langle w, A^{2k}(I-A)w\rangle}{\langle w, A^{2k}w\rangle} \\ &= \frac{\sum_{i\in[n]} w_i^2\lambda_i^{2k}(1-\lambda_i)}{\sum_{i\in[n]} w_i^2\lambda_i^{2k}} \qquad // \ A = \operatorname{diag}(\lambda_1,\dots,\lambda_n) \\ &= \frac{\sum_{i>1} w_i^2\lambda_i^{2k}(1-\lambda_i)}{w_1^2 + \sum_{i>1} w_i^2\lambda_i^{2k}}. \end{aligned} \tag{15.1}$$

15.1.2 Classic analysis

Theorem 15.1 Let A be a symmetric psd matrix with $\lambda_1 > \lambda_2 > \lambda_3$. Suppose that $w \in \mathbb{R}^n$ with $w_1 \neq 0$. Then the power method satisfies

$$\frac{\operatorname{err}(\xi_{k+1})}{\operatorname{err}(\xi_k)} \to \left(\frac{\lambda_2}{\lambda_1}\right)^2, \qquad \text{as } k \to \infty.$$

This is a rather weak result. We have to make many assumptions, and in the end, we don't even get an explicit convergence rate, only a limit.

Proof. We use (15.1) twice to get

$$\frac{\operatorname{err}(\xi_{k+1})}{\operatorname{err}(\xi_k)} = \frac{\sum_{i>1} w_i^2\lambda_i^{2(k+1)}(1-\lambda_i)}{\sum_{i>1} w_i^2\lambda_i^{2k}(1-\lambda_i)} \cdot \frac{w_1^2 + \sum_{i>1} w_i^2\lambda_i^{2k}}{w_1^2 + \sum_{i>1} w_i^2\lambda_i^{2(k+1)}}.$$

Since $1 = \lambda_1 > \lambda_2 > \lambda_3$, the limit of the first fraction is λ_2^2 and the second is 1. Since $\lambda_1 = 1$, the statement follows. Note that if $\lambda_1 \neq 1$, we had to divide all terms over λ_1^{2k+1} and the same arguments can be applied. ■

15.2 Randomized power method

In general, the power method may not converge to the largest eigenvalue. This happens if we start from $y_0 = w$ that is an eigenvector that corresponds to λ_i, $i > 1$. This is why, in Theorem 15.1 we asked $w_1 \neq 0$. And this is why in practice we often select the initial point $y_0 = w$ at random.

However, we will show[1] that this approach has significant implications, particularly in the convergence rates of the method.

We refer to Algorithm 5 as the randomized power method if we select $y_0 = w \sim N(0, I_n)$.

15.2.1 RPM with a spectral gap

By a spectral gap we mean that there is a strict inequality $\lambda_1 > \lambda_2$.

Theorem 15.2 Let A be an $n \times n$ symmetric psd matrix with $\lambda_1 > \lambda_2$. Then the randomized power method satisfies

$$\mathbf{E}[\operatorname{err}(\xi_k)] \leqslant \sqrt{\frac{(n-1)\pi}{2}} \cdot \left(\frac{\lambda_2}{\lambda_1}\right)^k.$$

[1] All the proofs below are not required for the exam.

By introducing $\gamma = \frac{\lambda_1 - \lambda_2}{\lambda_1}$, we can rewrite the above inequality as

$$\mathbf{E}[\mathrm{err}(\xi_k)] \leqslant \sqrt{\frac{(n-1)\pi}{2}} \cdot (1-\gamma)^k \leqslant \sqrt{\frac{(n-1)\pi}{2}} \cdot e^{-\gamma k}.$$

This means that RPM converges linearly. First we need an auxiliary lemma. Its proof requires only some basic calculus.

Lemma 15.1 Let $g \sim N(0,1)$. Then for any $c > 0$,

$$\mathbf{E}\left[\frac{1}{g^2 + c}\right] = \frac{1}{\sqrt{c}} e^{c/2} \int_{\sqrt{c}}^{\infty} e^{-t^2/2} dt \leqslant \min\left\{\sqrt{\frac{\pi}{2c}}, \frac{1}{c}\right\}.$$

Proof of Theorem 15.2. Since $w \sim N(0, I_n)$, each coordinate $w_i \sim N(0,1)$ and by tower rule and Lemma 15.1,

$$\mathbf{E}[\mathrm{err}(\xi_k)] = \mathbf{E}\left[\mathbf{E}_{w_1}\left[\frac{\sum_{i>1} w_i^2 \lambda_i^{2k}(1-\lambda_i)}{w_1^2 + \sum_{i>1} w_i^2 \lambda_i^{2k}}\right]\right] \leqslant \sqrt{\frac{\pi}{2}} \mathbf{E}\left[\frac{\sum_{i>1} w_i^2 \lambda_i^{2k}(1-\lambda_i)}{\left(\sum_{i>1} w_i^2 \lambda_i^{2k}\right)^{1/2}}\right].$$

By Cauchy-Schwarz,

$$\sum_{i>1} w_i^2 \lambda_i^{2k}(1-\lambda_i) \leqslant \left(\sum_{i>1} w_i^2 \lambda_i^{2k}(1-\lambda_i)^2\right)^{1/2} \left(\sum_{i>1} w_i^2 \lambda_i^{2k}\right)^{1/2}.$$

Hence, we can continue

$$\begin{aligned}
\mathbf{E}[\mathrm{err}(\xi_k)] &\leqslant \sqrt{\frac{\pi}{2}} \mathbf{E}\left(\sum_{i>1} w_i^2 \lambda_i^{2k}(1-\lambda_i)^2\right)^{1/2} \\
&\leqslant \sqrt{\frac{\pi}{2}} \left(\sum_{i>1} \mathbf{E}\left[w_i^2\right] \lambda_i^{2k}(1-\lambda_i)^2\right)^{1/2} && // \ \mathbf{E}[f(x)] \leqslant f(\mathbf{E}x) \text{ for a concave } f \\
&\leqslant \sqrt{\frac{\pi}{2}} \left((n-1) \max_{i>1} \lambda_i^{2k}(1-\lambda_i)^2\right)^{1/2} && // \ \mathbf{E}\left[w_i^2\right] = 1 \\
&\leqslant \sqrt{\frac{(n-1)\pi}{2}} \lambda_2^k \\
&= \sqrt{\frac{(n-1)\pi}{2}} \left(\frac{\lambda_2}{\lambda_1}\right)^k = \sqrt{\frac{(n-1)\pi}{2}} (1-\gamma)^k,
\end{aligned}$$

which concludes the proof. ■

15.2.2 RPM without a spectral gap

Without assuming a spectral gap, the rate obviously becomes worse. But importantly that we are still able to have it.

Theorem 15.3 Let A be an $n \times n$ symmetric psd matrix. Then the randomized power method satisfies

$$\mathbf{E}[\mathrm{err}(\xi_k)] \leqslant \frac{1}{k}\left(1 + \log\sqrt{\frac{(n-1)\pi}{2}} + \log k\right).$$

The proof is a more refined version of the proof of Theorem 15.2.

Proof. Let $X = \sum_{i>1} w_i^2 \lambda_i^{2k}$. Then, as before, we have

$$\mathbf{E}[\mathrm{err}(\xi_k)] = \mathbf{E}\left[\mathbf{E}_{w_1}\left[\frac{\sum_{i>1} w_i^2 \lambda_i^{2k}(1-\lambda_i)}{w_1^2 + X}\right]\right].$$

Using the full version of the bound in Lemma 15.1, we get

$$\mathbf{E}[\mathrm{err}(\xi_k)] \leqslant \mathbf{E}\left[\sum_{i>1} w_i^2 \lambda_i^{2k}(1-\lambda_i) \cdot \min\left\{\sqrt{\frac{\pi}{2X}}, \frac{1}{X}\right\}\right].$$

The idea is now to estimate terms in the sum differently depending on whether they are larger than $1-\beta$ or not. We choose any $\beta > 0$ and will optimize it later. By splitting the sum, we obtain

$$\sum_{i>1} w_i^2 \lambda_i^{2k}(1-\lambda_i) \leqslant \sum_{\lambda_i \leqslant 1-\beta} w_i^2 \lambda_i^{2k} + \beta \sum_{\lambda_i > 1-\beta} w_i^2 \lambda_i^{2k} \leqslant \sum_{\lambda_i \leqslant 1-\beta} w_i^2 \lambda_i^{2k} + \beta X.$$

Using this bound, we can continue with $\mathbf{E}[\mathrm{err}(\xi_k)]$:

$$\begin{aligned}
\mathbf{E}[\mathrm{err}(\xi_k)] &\leqslant \mathbf{E}\left[\left(\sum_{\lambda_i \leqslant 1-\beta} w_i^2 \lambda_i^{2k} + \beta X\right) \cdot \min\left\{\sqrt{\frac{\pi}{2X}}, \frac{1}{X}\right\}\right] \\
&\leqslant \mathbf{E}\left[\left(\sum_{\lambda_i \leqslant 1-\beta} w_i^2 \lambda_i^{2k}\right)\sqrt{\frac{\pi}{2X}}\right] + \beta \\
&\leqslant \sqrt{\frac{\pi}{2}}\mathbf{E}\left[\left(\sum_{\lambda_i \leqslant 1-\beta} w_i^2 \lambda_i^{2k}\right)^{1/2}\right] + \beta \\
&\leqslant \sqrt{\frac{\pi}{2}}\left(\sum_{\lambda_i \leqslant 1-\beta} \lambda_i^{2k}\right)^{1/2} + \beta \qquad \text{// Jensen and } \mathbf{E}\, w_i^2 = 1 \\
&\leqslant \sqrt{\frac{(n-1)\pi}{2}}(1-\beta)^k + \beta \\
&\leqslant \sqrt{\frac{(n-1)\pi}{2}} e^{-\beta k} + \beta.
\end{aligned}$$

Minimizing last expression over β, we obtain the desired inequality. ■

All in all, we obtain an $O\left(\frac{\log k}{k}\right)$ convergence rate. It is possible to eliminate the $\log k$ dependency through more advanced analysis.

15.2.3 Discussion

One could notice that the rates we derived are familiar; they are almost the same as those for gradient descent in the convex and strongly convex cases.

Let $B = I - A$ and consider the following problem:

$$\min_{\|x\|_2=1} f(x) := \frac{1}{2}\langle Bx, x\rangle.$$

Clearly, this problem finds the smallest eigenvalue of B, which is equivalent to finding the largest eigenvalue of A. Matrix B has eigenvalues between $[0, 1]$, so we can apply the projected gradient method with $\alpha = 1$ to this problem:

$$x_{k+1} = P_{S^{n-1}}(x_k - \alpha \nabla f(x_k)) = P_{S^{n-1}}(x_k - Bx_k) = \frac{Ax_k}{\|Ax_k\|}.$$

This is exactly the power method we consider! While we used the projection onto the unit sphere S^{n-1} (a nonconvex set!), it is not essential. The power method uses normalization only for numerical purposes. The question then is how to interpret the results of the randomized power method through the language of gradient descent. What is the role of a spectral gap there?

Comments

The analysis of the randomized power method are from [1]. In our exposition, we mostly followed lecture notes [2].

References

[1] Jacek Kuczyński and Henryk Woźniakowski. "Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start". In: *SIAM journal on matrix analysis and applications* 13.4 (1992), pages 1094–1122.

[2] Joel A Tropp. *Randomized algorithms for matrix computations*. 2020.