Optimization Methods for Data Science (2023W) 2-5 Oct 2023

Lecture 1

Lecturer: Yurii Malitskyi

1 Foundations

Optimization problems can be *unconstrained* as in $\min_{x\in\mathbb{R}^n} f(x)$ or *constrained* as $\min_{x\in C} f(x)$, where $C\subset\mathbb{R}^n$ is some set. For now we only focus on unconstrained minimization. We often refer to the function f as to the *objective function* or just *objective*. Although it is common to write $\min_x f(x)$, most of the time we are actually interested in $\operatorname{argmin}_x f(x)$.

A point $x\in\mathbb{R}^n$ is called a *global minimum*, if $f(x)\leqslant f(y)$ for all $y\in\mathbb{R}^n$. A point $x\in\mathbb{R}^n$ is called a *local minimum*, if there is a neighborhood $B(x,r)$ of x such that $f(x)\leqslant f(y)$ for all $y\in B(x,r)$.

Notation. We use x^* or x_* to denote a solution of the optimization problem. We use $\langle x,y\rangle = x^\top y$ to denote the scalar product between vectors x and y. Unless otherwise stated, $\|x\| = \langle x,x\rangle^{1/2}$ denotes the standard Euclidean norm. For $x,y\in\mathbb{R}^n$ we write $[x,y]$ to denote the segment between x and y, that is $[x,y]=\{(1-t)x+ty\colon t\in[0,1]\}$.

An *iterative algorithm* is a map $\mathscr{A}_f$ that takes already computed iterates as an input and outputs a new one
$$x_{k+1}=\mathscr{A}_f(x_k,\dots,x_0).$$
Our goal is to construct such an algorithm $\mathscr{A}$. For this a few factors should be taken into account: (i) what properties the objective f satisfies and (ii) what information about f is available to us. For instance, f may be differentiable or not, convex, strongly convex, nonconvex, etc. We may know about f its gradient $\nabla f(x)$, its Hessian $\nabla^2 f(x)$, only its value $f(x)$, etc.

In general, we are interested in an approximate solution. Various approximate optimality measures can be used
$$\|x_k-x_*\|\leqslant\varepsilon \quad\text{or}\quad f(x_k)-f_*\leqslant\varepsilon \quad\text{or}\quad \|\nabla f(x_k)\|\leqslant\varepsilon,$$
where ε is the target accuracy and x_k is the output of the algorithm.

1.1 Differentiability

Gradients. We say that $f\colon\mathbb{R}^n\to\mathbb{R}$ is *differentiable at* x if there is a vector $u\in\mathbb{R}^n$ such that for all $d\in\mathbb{R}^n$ it holds that
$$f(x+d)=f(x)+\langle u,d\rangle+o(\|d\|),$$
where notation $o(t)$ reads as $\lim_{t\to 0}\frac{o(t)}{t}=0$. Such a vector u is called the *gradient* of f at x and is denoted by $\nabla f(x)$. Thus, we can rewrite the above as
$$f(x+d)=f(x)+\langle\nabla f(x),d\rangle+o(\|d\|) \tag{1.1}$$
By choosing d as the basis vector e_i for $i=1,\dots,n$, it is easy to see that
$$\nabla f(x)=\left(\frac{\partial f(x)}{\partial x_1},\dots,\frac{\partial f(x)}{\partial x_n}\right)^\top.$$
We say that f is differentiable on $\mathbb{R}^n$ if it is differentible at every point $x\in\mathbb{R}^n$.

Instead of varying d, we can substitute in (1.1) $d := td$ and vary $t \in \mathbb{R}$. This yields

$$f(x+td) = f(x) + t\langle \nabla f(x), d\rangle + o(t) \tag{1.2}$$

The last equation is our recipe for computing ∇f analytically in practice, that is we represent $f(x+td)$ as

$$f(x+td) = f(x) + t\langle \text{some vector}, d\rangle + o(t)$$

and identify some vector as the gradient $\nabla f(x)$.

One can interpret (1.2) as a consequence of the standard single variable derivative of $\varphi(t) = f(x+td)$. If you are not very familiar with multivariable calculus, you can recover most formulae by using φ instead of f. For instance, we know that

$$\varphi(1) = \varphi(0) + \int_0^1 \varphi'(t)\,\mathrm{d}t.$$

It is fairly easy to see that this is nothing else as *Taylor's formula* with an integral remainder

$$f(x+d) = f(x) + \int_0^1 \langle \nabla f(x+td), d\rangle\,\mathrm{d}t. \tag{1.3}$$

Similarly, we can prove the *mean value theorem*: for a differentiable f and any x, d there exists $z \in [x, x+d]$ such that

$$f(x+d) = f(x) + \langle \nabla f(z), d\rangle. \tag{1.4}$$

Hessians. We say that $f: \mathbb{R}^n \to \mathbb{R}$ is twice differentiable at x if f is differentiable at x and there is a symmetric matrix $H \in \mathbb{R}^{n\times n}$ such that for all $d \in \mathbb{R}^n$ it holds that

$$f(x+d) = f(x) + \langle \nabla f(x), d\rangle + \frac{1}{2}\langle Hd, d\rangle + o\left(\|d\|^2\right).$$

The matrix H is called the *Hessian* of f at x and is denoted by $\nabla^2 f(x)$. As before, it is not difficult to see that $\nabla^2 f(x)$ is the matrix of second partial derivatives

$$\nabla^2 f(x) = \left[\frac{\partial^2 f(x)}{\partial x_i \partial x_j}\right]_{i,j=1}^n.$$

Gradients are of utmost importance in optimization, and this is due to several key reasons.

1. $\boxed{\text{Gradients define the direction of the local fastest increase of } f\text{:}}$

$$\frac{\nabla f(x)}{\|\nabla f(x)\|} = \operatorname*{argmax}_{\|d\|=1} \lim_{\alpha\to 0} \frac{f(x+\alpha d) - f(x)}{\alpha} = \operatorname*{argmax}_{\|d\|=1} \langle \nabla f(x), d\rangle. \tag{1.5}$$

   This follows directly from the Cauchy-Schwarz inequality.

2. $\boxed{\text{Gradients are orthogonal to the level sets of } f.}$

   Given $c \in \mathbb{R}$, a *level set* is $L_f(c) = \{x : f(x) = c\}$.

   A vector is orthogonal to a set at a point x_0 means that the vector is orthogonal to the tangent to this set that goes through x_0. With this at hand, this property becomes not difficult to check.

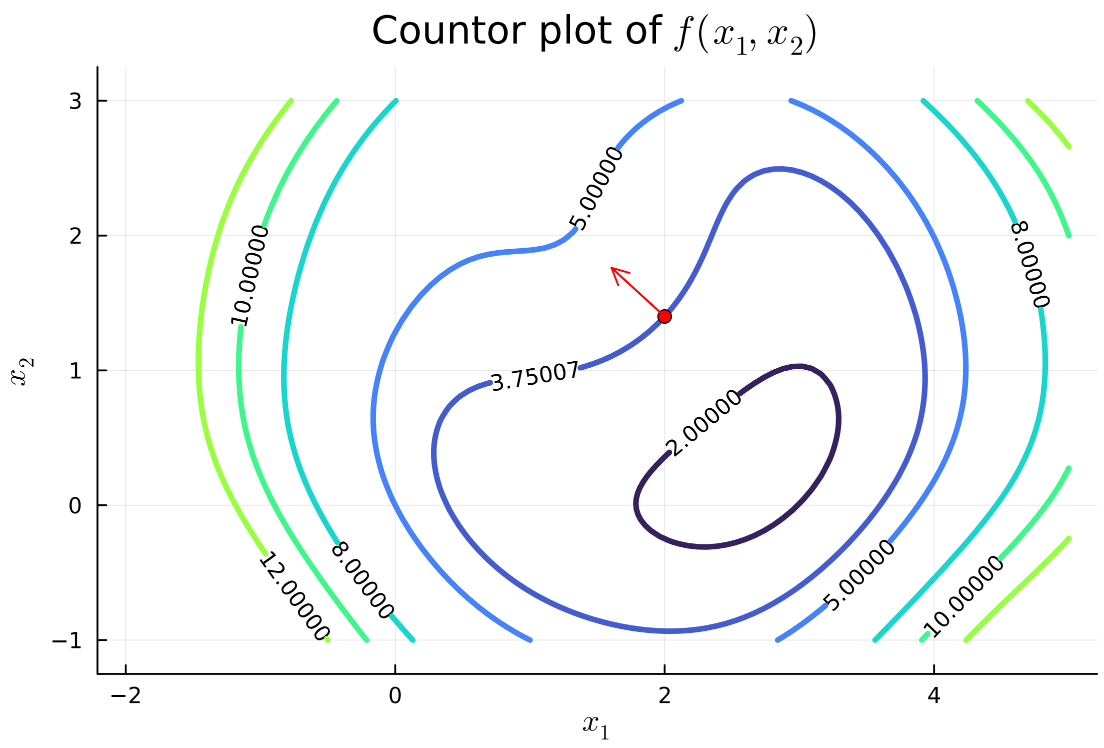

Figure 1: Countor plot of $f : \mathbb{R}^2 \to \mathbb{R}$. The red arrow depicts the gradient at the given point

3. If x is a local minimum, then $\nabla f(x) = 0$.

   Indeed, we have
$$\lim_{\alpha \to 0} \frac{f(x + \alpha d) - f(x)}{\alpha} = \langle \nabla f(x), d \rangle.$$
   If $\nabla f(x) \neq 0$, then we can take $d = -\nabla f(x)$ and α small enough so that $x + \alpha d \in B(x, r)$. By tending α to 0, we obtain a contradiction.

The last property is the most important one and is called the *first-order optimality condition*. It is a necessary optimality condition. Hessians provide a sufficient optimality condition, but we will cover it later if at all.

1.2 Convexity

A set $C \subset \mathbb{R}^n$ is *convex* if
$$\{x, y\} \in C \quad \Longrightarrow \quad [x, y] \in C.$$
A function $f : \mathbb{R}^n \to \mathbb{R}$ is called *convex* if
$$\lambda f(x) + (1 - \lambda) f(y) \geqslant f(\lambda x + (1 - \lambda) y) \qquad \text{for all } x, y \text{ and for all } \lambda \in [0, 1] \tag{1.6}$$
To understand why the same word is used for such seemingly unrelated objects, it is useful to have in mind an alternative definition. The *epigraph* epi(f) of the function f is the set
$$\operatorname{epi}(f) = \{(x, t) \in \mathbb{R}^n \times \mathbb{R} : f(x) \leqslant t\},$$
in other words, all those points that are on or above the graph of f.

Lemma 1.1. A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if epi(f) is a convex set.

There exist more refined classes of convex functions. We call $f : \mathbb{R}^n \to \mathbb{R}$

- *strictly convex* if we have a strict inequality in (1.6), that is

$$\lambda f(x) + (1-\lambda)f(y) > f(\lambda x + (1-\lambda)y) \qquad \text{for all } x \neq y \text{ and for all } \lambda \in (0,1) \tag{1.7}$$

- μ-*strongly convex* if

$$\lambda f(x)+(1-\lambda)f(y) \geqslant f(\lambda x+(1-\lambda)y)+\frac{\mu\lambda(1-\lambda)}{2}\|x-y\|^2 \qquad \text{for all } x, y \text{ and for all } \lambda \in [0,1] \tag{1.8}$$

We would like to have alternative characterization of convexity for some classes of functions.

Lemma 1.2. For a differentiable $f : \mathbb{R}^n \to \mathbb{R}$ the following holds:

- f is convex if and only if

$$f(y) \geqslant f(x) + \langle \nabla f(x), y - x\rangle \qquad \text{for all } x, y; \tag{1.9}$$

- f is μ-strongly convex if and only if

$$f(y) \geqslant f(x) + \langle \nabla f(x), y - x\rangle + \frac{\mu}{2}\|y-x\|^2 \qquad \text{for all } x, y. \tag{1.10}$$

Proof. We only prove the first equivalence. The second one follows seamlessly. Suppose f is convex. Then we can rewrite the definition (1.6) as

$$f(y) - f(x) \geqslant \frac{f(\alpha y + (1-\alpha)x) - f(x)}{\alpha} = \frac{f(x+\alpha(y-x)) - f(x)}{\alpha}.$$

By tending α to 0, we obtain the desired inequality.

Conversely, suppose that (1.9) holds. Let $z = (1-\alpha)x + \alpha y$. Then

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

Multiplying the first inequality by $(1-\alpha)$ and the second one by α and adding them up gives the desired inequality (1.6). ■

Lemma 1.3. For a twice differentiable $f : \mathbb{R}^n \to \mathbb{R}$, the following holds

- f is convex if and only if

$$\nabla^2 f(x) \succcurlyeq 0 \qquad \text{for all } x; \tag{1.11}$$

- f is μ-strong convexity if and only if

$$\nabla^2 f(x) \succcurlyeq \mu I \qquad \text{for all } x; \tag{1.12}$$

In other words, twice differentiable f is convex iff its Hessian is a positive semidefinite matrix and it is strongly convex if the Hessian is positive definite.

Proof. Again, we prove only the first equivalence. Suppose f is convex. Then we have

$$f(x+td) = f(x) + t\langle \nabla f(x)d, d\rangle + \frac{t^2}{2}\langle \nabla^2 f(x)d, d\rangle + o\left(t^2\right).$$

By Lemma 1.2 we know that

$$f(x+td) - f(x) - t\langle \nabla f(x)d, d\rangle \geqslant 0.$$

Hence,

$$\frac{t^2}{2}\langle \nabla^2 f(x)d, d\rangle + o\left(t^2\right) \geqslant 0.$$

After dividing over t^2 and tending t to 0, we obtain $\langle \nabla^2 f(x)d, d\rangle \geqslant 0$. Hence $\nabla^2 f(x)$ is a psd matrix.

The converse follows from the Taylor formula

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

where $z \in [x, y]$ and Lemma 1.2. ∎

It is difficult to overstate how important both Lemmas 1.2 and 1.3 are. We will use those alternative definitions much more often than the original definitions in (1.6) and (1.8).

Why do we care about convexity? Next statement partially explains this.

Lemma 1.4. Let $f : \mathbb{R}^n \to \mathbb{R}$. The following holds:

(i) If f is convex, then every local minimum is a global one and the set of minima is convex.

(ii) If f is strictly convex, then it has at most one minimum.

(iii) If f is strongly convex, then minimum always exists.

Another aspect why convexity is important is that better optimization algorithms can be developed for such functions.

EXERCISES[1]

Exercise 1. Make sure you understand how to derive formula (1.4) using function $\varphi(t) = f(x+td)$.

Exercise 2. Make sure you understand what equation (1.5) means and why that derivation is correct.

Exercise 3. Find the gradient and the Hessian of $f(x) = \frac{1}{2}\langle Qx, x\rangle - \langle b, x\rangle + c$, where $x, b \in \mathbb{R}^n$, $Q \in \mathbb{R}^{n\times n}$, and $c \in \mathbb{R}$.

Exercise 4. When is the function defined in the previous exercise (i) convex; (ii) strongly convex?

Exercise 5. Prove equivalence in (ii) in Lemma 1.2.

Exercise 6. One may expect that in Lemma 1.3, strict convexity is equivalent to $\nabla^2 f(x) \succ 0$. Show that this is not true.

[1]Do them to make sure you understand the material.

Optimization Methods for Data Science (2023W) 9-12 Oct 2023

Lecture 2

Lecturer: Yurii Malitskyi

2 Gradient descent

Most optimization methods fit in the following framework. We choose some initial $x_0 \in \mathbb{R}^n$ and generate (x_k) by

$$x_{k+1} = x_k + \alpha_k d_k, \tag{2.1}$$

where $\alpha_k > 0$ is called a stepsize (or a learning rate) and $d_k \in \mathbb{R}^n$ defines a direction.

Choosing $d_k = -\nabla f(x_k)$ in (2.1) leads to the *gradient descent* method

$$\boxed{x_{k+1} = x_k - \alpha_k \nabla f(x_k).} \tag{2.2}$$

What was our motivation to choose $d_k = -\nabla f(x_k)$? There are many ways to motivate this method. For instance,

1. We know that gradient can be defined alternatively as the direction of the local fastest increase of f:

$$\frac{\nabla f(x)}{\|\nabla f(x)\|} = \operatorname*{argmax}_{\|d\|=1} \lim_{\alpha \to 0} \frac{f(x+\alpha d) - f(x)}{\alpha} = \operatorname*{argmax}_{\|d\|=1} \langle \nabla f(x), d \rangle.$$

   Likewise, anti-gradient, $-\nabla f(x)$, defines the direction of the fastest descent of f

$$-\frac{\nabla f(x)}{\|\nabla f(x)\|} = \operatorname*{argmin}_{\|d\|=1} \lim_{\alpha \to 0} \frac{f(x+\alpha d) - f(x)}{\alpha} = \operatorname*{argmin}_{\|d\|=1} \langle \nabla f(x), d \rangle.$$

   Hence, given x_k, the most natural direction to decrease $f(x_k)$ is to move along $-\nabla f(x_k)$.

2. Minimizing f is hard, so let's minimize its Taylor's approximation around x_k. Consider the first-order approximation

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

   This approximation is linear over x, so minimizing it doesn't make much sense — we always obtain $-\infty$. To prevent this, we can add some regularization and thus, we define x_{k+1} as

$$x_{k+1} = \operatorname*{argmin}_x \Big\{ f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2\alpha_k} \|x - x_k\|^2 \Big\}. \tag{2.3}$$

   It is each to check that this will lead to GD as in (2.2).

To analyze GD, we need to indroduce another class of funtions.

***L*-smooth functions.** We say that $f : \mathbb{R}^n \to \mathbb{R}$ is *L-smooth* if its gradient ∇f is *L-Lipschitz continuous*, that is

$$\|\nabla f(x) - \nabla f(y)\| \leqslant L\|x - y\| \quad \text{for all } x, y \in \mathbb{R}^n. \tag{2.4}$$

If f is additionally twice differentible, then the above inequality is equivalent to

$$\nabla^2 f(x) \preccurlyeq LI \quad \iff \quad \lambda_{\max}(\nabla^2 f(x)) \leqslant L. \tag{2.5}$$

Lemma 2.1 (Descent lemma). If f is L-smooth, then for all x, y it holds

$$f(y) \leqslant f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|y - x\|^2. \tag{2.6}$$

Proof. We prove it only for twice differentiable functions. By the Taylor formula, for any x, y there exists $z \in [x, y]$ such that

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

Now, by L-smoothness of f, we have that $\nabla^2 f(z) \preccurlyeq L$ and the proof is complete. ∎

Combining Lemma 1.3 and equation (2.5), we obtain the following characterization of convex, L-smooth and twice differentiable function f:

$$\boxed{0 \preccurlyeq \nabla^2 f(x) \preccurlyeq LI \quad \text{for all } x.}$$

The best example of such functions is a convex quadratic functions

$$f(x) = \frac{1}{2}\langle Qx, x\rangle - \langle b, x\rangle, \tag{2.7}$$

where $Q \succcurlyeq 0$ is a positive semidefinite matrix. Convex quadratic functions are important both on their own right and as second-order approximation of general convex functions.

An important subclass of positive semidefinite matrices is positive definite matrices, for which we use notation $Q \succ 0$. If $\lambda_1 \leqslant \lambda_2 \leqslant \ldots \leqslant \lambda_n$ are eigenvalues of Q, then Q is positive definite if $\lambda_1 > 0$. The number

$$\kappa(Q) = \frac{\lambda_n}{\lambda_1}$$

is called the *condition number* of the matrix Q. Roughly speaking, the larger this number is, the more difficult the problem of $\min_x f(x)$.

One can analyze GD for different classes of functions:

- convex, L-smooth;
- μ-strongly convex, L-smooth;
- nonconvex, L-smooth.

For all these cases we can establish a descent property of one step of the GD.

Lemma 2.2 (One step of GD). Suppose that f is L-smooth. Then one step of GD with the stepsize α satisfies

$$f(x_{k+1}) \leqslant f(x_k) - \frac{\alpha(2 - \alpha L)}{2}\|\nabla f(x_k)\|^2. \tag{2.8}$$

Proof. By inequality (2.6) 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 (2.8), 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 $\alpha = \frac{1}{L}$[2]. With this choice, we obtain

$$f(x_{k+1}) \leqslant f(x_k) - \frac{1}{2L}\|\nabla f(x_k)\|^2. \tag{2.9}$$

[2]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 might be too loose.

GD for nonconvex functions. In general, for nonconvex functions we cannot guarantee convergence to a global minimum, nor even to a local minimum. Without further assumptions, we can only talk about *stationary points* that are points where the gradient vanishes.

Theorem 2.1 (GD for nonconvex functions)**.** *Suppose that f is L-smooth and lower-bounded by f_{low}, that is $f(x) \geqslant f_{\mathrm{low}}$ for all x. Then with $\alpha_k = \frac{1}{L}$ we have that*

$$\min_{i=0,\ldots,k} \|\nabla f(x_i)\|^2 \leqslant \frac{2L(f(x_0) - f_{\mathrm{low}})}{k}. \tag{2.10}$$

Proof. From (2.8) with $\alpha = \frac{1}{L}$, we have that

$$f(x_{k+1}) \leqslant f(x_k) - \frac{1}{2L}\|\nabla f(x_k)\|^2,$$

which we can rewrite as

$$\frac{1}{2L}\|\nabla f(x_k)\|^2 \leqslant f(x_k) - f(x_{k+1}).$$

Since this holds for all $k \geqslant 0$, we can sum this inequality for all $i = 0, \ldots, k$, which gives

$$\frac{1}{2L}\sum_{i=0}^{k} \|\nabla f(x_i)\|^2 \leqslant f(x_0) - f(x_{k+1}) \leqslant f(x_0) - f_{\mathrm{low}}.$$

Then the desired bound follows easily

$$(k+1) \min_{i=0,\ldots,k} \|\nabla f(x_i)\|^2 \leqslant 2L(f(x_0) - f_{\mathrm{low}}).$$

Just to make the final bound slightly simpler, we can use that $k \leqslant k+1$ and write it as

$$k \min_{i=0,\ldots,k} \|\nabla f(x_i)\|^2 \leqslant 2L(f(x_0) - f_{\mathrm{low}}).$$

■

What is important to remember here is that the convergence rate of GD in the nonconvex case is

$$\min_{i=0,\ldots,k} \|\nabla f(x_i)\| = O\left(\frac{1}{\sqrt{k}}\right).$$

Given accuracy $\varepsilon > 0$, how many iterations of GD do we need to ensure that we found a point $\bar{x}$ with $\|\nabla f(\bar{x})\| \leqslant \varepsilon$? This means that we want to find the smallest k such that

$$\frac{2L(f(x_0) - f_{\mathrm{low}})}{k} \leqslant \varepsilon^2.$$

This implies that

$$k \geqslant \frac{2L(f(x_0) - f_{\mathrm{low}})}{\varepsilon^2}.$$

Hence, the first integer k that satisfies this bound is our answer. For us, it is enough to remember that the number of iterations is $O\left(\frac{1}{\varepsilon^2}\right)$. This is a slow rate: say for a moderate accuracy $\varepsilon = 10^{-3}$, we need of order 10^6 iterations.

GD for convex functions. This is probably the most important case. The proof is slightly more complicated than in the previous case, but still relatively simple.

Theorem 2.2 (Gradient descent for convex functions)**.** *Suppose that f is convex and L-smooth. Then with $\alpha_k = \frac{1}{L}$, we have that*

$$f(x_k) - f_* \leqslant \frac{L\|x_0 - x^*\|^2}{2k}. \tag{2.11}$$

Thus, convergence rate of GD is

$$f(x_k) - f_* \leqslant O\left(\frac{1}{k}\right).$$

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

$$\frac{L\|x_0 - x^*\|^2}{2k} \leqslant \varepsilon \qquad \Longleftrightarrow \qquad k \geqslant \frac{L\|x_0 - x^*\|^2}{2\varepsilon}.$$

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.

GD for strongly convex functions. Now we analyze GD, when f is μ-strongly convex and L-smooth (the most favorable case). The first property has the following consequence.

Lemma 2.3. Let f be μ-strongly convex and x^* be the solution. Then

$$\frac{1}{2\mu}\|\nabla f(x)\|^2 \geqslant f(x) - f_* \quad \text{for all } x. \tag{2.12}$$

Proof. This is a direct consequence of (1.10):

$$\begin{aligned} f(x_*) &\geqslant f(x) + \langle \nabla f(x), x^* - x\rangle + \frac{\mu}{2}\|x - x^*\|^2 \\ &\geqslant f(x) - \frac{1}{2\mu}\|\nabla f(x)\|^2 - \frac{\mu}{2}\|x - x^*\|^2 + \frac{\mu}{2}\|x - x^*\|^2 \\ &\geqslant f(x) - \frac{1}{2\mu}\|\nabla f(x)\|^2, \end{aligned}$$

where in the second inequality we used that $\langle a, b\rangle \leqslant \frac{1}{2\mu}\|a\|^2 + \frac{\mu}{2}\|b\|^2$. ■

Theorem 2.3 (GD for strongly convex functions)**.** *Suppose that f is μ-strongly convex and L-smooth. Then with $\alpha_k = \frac{1}{L}$, we have that*

$$f(x_k) - f_* \leqslant \left(1 - \frac{\mu}{L}\right)^k (f(x_0) - f_*). \tag{2.13}$$

Proof. Setting $\alpha = \frac{1}{L}$ in (2.8) yields

$$\begin{aligned} f(x_{k+1}) &\leqslant f(x_k) - \frac{1}{2L}\|\nabla f(x_k)\|^2 \\ &\leqslant f(x_k) - \frac{\mu}{L}(f(x_k) - f_*) \qquad //\text{By Lemma 2.3.} \end{aligned}$$

Hence, $f(x_{k+1}) - f_* \leqslant \left(1 - \frac{\mu}{L}\right)(f(x_k) - f_*)$. ■

We call such convergence *linear*. In terms of iterations to reach the minimum, we need to find k such that

$$\left(1 - \frac{\mu}{L}\right)^k (f(x_0) - f_*) \leqslant \varepsilon.$$

Of course, we could solve this inequality, but the resulting expression will be messy. Instead, we use a slightly cruder inequality that will result in a simpler final bound. Using $1 - t \leqslant e^{-t}$, we want to find k such that

$$\left(1 - \frac{\mu}{L}\right)^k (f(x_0) - f_*) \leqslant e^{-k\mu/L}(f(x_0) - f_*) \leqslant \varepsilon,$$

which gives us

$$k \geqslant \frac{L}{\mu}\log\left(\frac{f(x_0) - f_*}{\varepsilon}\right).$$

Here, the dependency on ε is much better than before. Similar to the case of quadratic functions with positive definite matrices, we call the number $\frac{L}{\mu}$ the *condition number* of the function f.

Stepsize. What are the ways to choose the stepsize α_k?

- Fixed stepsize $\alpha_k = \alpha < \frac{2}{L}$. For this f must be L-smooth and we should know L.
- Use linesearch. In every iteration we choose a trial stepsize α_k, check a special condition. If this condition is satisfied, we proceed to the next iteration and decrease α_k otherwise and repeat calculations.
- Adaptive. Compute α_k on the fly using all available information about f and previous iterates $x_0, \dots, x_k$.
- Steepest descent. Choose stepsize α_k that provides the smallest value of f:

  $$\alpha_k = \operatorname*{argmin}_{\alpha \geqslant 0} f(x_k - \alpha\nabla f(x_k)).$$

  This approach is not very practical, since in every iteration we have to solve another minimization problem.

Exercises

Exercise 7. Show that the update in (2.3) is indeed equivalent to the GD update.

Exercise 8. Show that $\alpha = \frac{1}{L}$ is indeed the maximum of $\alpha(2 - \alpha L)$.

Exercise 9. Prove that f in (2.7) is convex if and only if $Q \succcurlyeq 0$.

Exercise 10. Prove that f in (2.7) is L-smooth, with $L = \|Q\| = \lambda_{\max}(Q)$.

Exercise 11. Explain all the steps in the proof of Lemma 2.3.

Exercise 12. Prove that $1 - t \leqslant e^{-t}$ using only convexity arguments.

Optimization Methods for Data Science (2023W) 16 Oct 2023

Lecture 3

Lecturer: Yurii Malitskyi

3 Acceleration. Emprical risk minimization

3.1 Convex functions

In the previous lecture, we learned that for a convex and L-smooth function f, gradient descent enjoys the following rate

$$f(x_k) - f_* \leqslant \frac{L\|x_0 - x^*\|^2}{2k}.$$

It's worth noting that this result is not only accurate but also the best possible, as there exists a function within our class of convex and L-smooth functions where this rate is exact.

Now, the question arises: can we improve upon this result within our defined framework? To be more precise, let's define the class of functions as convex and L-smooth functions, and the class of algorithms as *first-order algorithms*, which are limited to using function values and gradients. Our question now becomes: can we develop a better first-order algorithm for this class of functions? A crucial result asserts that within this class of functions, no algorithm can achieve a rate faster than $O\left(\frac{1}{k^2}\right)$[3]. As there is still a gap between $O\left(\frac{1}{k}\right)$ and $O\left(\frac{1}{k^2}\right)$, we must explore whether an algorithm with the latter rate is attainable.

Polyak's heavy-ball method. In 1964, Polyak introduced a modification of gradient descent, known as the *heavy-ball* method:

$$x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta(x_k - x_{k-1}). \tag{3.1}$$

In this equation, $\alpha > 0$ serves as the stepsize, $\beta > 0$ is a parameter, and we set $x^{-1} = x^0$. When appropriately tuned, this method has been shown to converge much faster than GD on quadratic problems with positive definite matrices. In more general convex cases, the theoretical guarantees of the *heavy-ball* method may not be as strong, but it often performs well in practice. The term we added, $\beta(x_k - x_{k-1})$, is commonly referred to as *momentum* and will play a significant role in our later lectures.

Nesterov's accelerated method. In 1983, Nesterov, who was Polyak's student, proposed another modification of GD, very much in the spirit of (3.1):

$$\begin{aligned} y_k &= x_k + \beta_k(x_k - x_{k-1}) \\ x_{k+1} &= y_k - \alpha \nabla f(y_k), \end{aligned} \tag{3.2}$$

where $k \geqslant 0$ and $x_{-1} = x_0$. For this scheme, one can prove the following convergence result.

Theorem 3.1 (Nesterov's accelerated method)**.** *Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is convex and L-smooth. Then iterates of* (3.2) *with $\alpha = \frac{1}{L}$ and $\beta_k = \frac{k}{k+3}$ satisfy*

$$f(x_k) - f_* \leqslant \frac{2L\|x_0 - x^*\|^2}{k^2}.$$

We will not prove this theorem, although the proof is not very difficult.

To conclude, we have shown that the Nesterov accelerated method achieves the lover bound $O\left(\frac{1}{k^2}\right)$ on the class of convex L-smooth functions.

[3]The formal statement is a bit more complicated, typically requiring the dimension n to satisfy $n \geqslant 2k+1$, but we'll omit those details for simplicity.

3.2 Strongly convex functions

The same question can be asked for the class of L-smooth and μ-strongly convex functions. First recall that GD on this class enjoys the rate

$$f(x_k) - f_* \leqslant \left(1 - \frac{\mu}{L}\right)^k (f(x_0) - f_*)$$

or, if we want to express it in ε notation, as

$$f(x_k) - f_* \leqslant \varepsilon \quad \text{if} \quad k \geqslant \frac{L}{\mu} \log\left(\frac{f(x_0) - f_*}{\varepsilon}\right).$$

Indeed, we can improve upon this result. And a good thing is that we don't need to consider new algorithms.

Theorem 3.2 (Nesterov's accelerated method for strongly convex functions)**.** *Suppose that* $f : \mathbb{R}^n \to \mathbb{R}$ *is* μ*-strongly convex and* L*-smooth. Then iterates of* (3.2) *with* $\alpha = \frac{1}{L}$ *and* $\beta_k = \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}$ *satisfy*

$$f(x_k) - f_* \leqslant \left(1 - \sqrt{\frac{\mu}{L}}\right)^k \left(f(x_0) - f_* + \frac{\mu}{2}\|x_0 - x^*\|^2\right).$$

Obviously, by transforming this rate in ε-notation, we obtain

$$f(x_k) - f_* \leqslant \varepsilon \quad \text{if} \quad k \geqslant \sqrt{\frac{L}{\mu}} \log\left(\frac{f(x_0) - f_* + \frac{\mu}{2}\|x_0 - x^*\|^2}{\varepsilon}\right).$$

If $\frac{L}{\mu}$ is a big number, acceleration provides us a great improvement.

3.3 Empirical risk minimization

Consider a labeled dataset $(a_1, b_1), \dots, (a_m, b_m)$, where $a_i \in \mathbb{R}^d$ represents the main objects of interest (text, image, video, etc.). The coordinates of a_i are referred to as *features* and b_i is the corresponding *label* or *observation*. For example, $a_1, \dots, a_m$ could represent a collection of images containing a car or not, with b_i taking values of ± 1 for these cases.

The objective of a data analyst is to find a function φ such that $\varphi(a_i) \approx b_i$ for $i = 1, \dots, m$. This process of finding such a function is commonly known as *learning* or *training*.

It's evident that our problem is inherently challenging: we have a finite dataset but an infinite number of potential functions φ. Furthermore, we could define a function that trivially satisfies the equation $\varphi(a_i) = b_i$ by definition. However, this function is not useful because our goal is not merely to match $\varphi(a_i) \approx b_i$ for $i = 1, \dots, m$. Our true aim is to discover a φ that performs well on unseen data. We assume that our initial dataset $(a_i, b_i)_{i=1}^m$ is a representative sample of the data that interests us. Using this limited dataset, we seek to find a φ that can predict labels for new data points effectively; in other words, we aim for it to *generalize* well.

For a given φ, we measure whether our prediction is good or not, by using a *loss* function $\ell \colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ and measure how large is $\ell(\varphi(a_i), b_i)$. In other words, how far apart (in some metric) our predection $\widehat{b}_i = \varphi(a_i)$ and the true label b_i are. It is common to have

$$\varphi(\widehat{b}_i, b_i) = 0 \quad \iff \quad \widehat{b}_i = b_i.$$

For a given φ, we assess the quality of our predictions using a *loss* function $\ell \colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$. This function quantifies the disparity between our prediction $\widehat{b}_i = \varphi(a_i)$ and the actual label b_i. Often we choose ℓ to have

$$\ell(\widehat{b}_i, b_i) = 0 \quad \iff \quad \widehat{b}_i = b_i.$$

Now we are ready to have our first important definition.

Definition 1. Given a dataset $(a_i, b_i)_{i=1}^m$ and the loss function ℓ, the *empirical risk* of a predictor φ is

$$R(\varphi) = \frac{1}{m}\sum_{i=1}^m \ell(\varphi(a_i), b_i).$$

Now *empirical risk minimization* problem is naturally defined as

$$\min_{\varphi} R(\varphi) := \frac{1}{m}\sum_{i=1}^m \ell(\varphi(a_i), b_i). \tag{3.3}$$

However, this problem remains intractable due to the vast number of potential functions φ. To address this, we can constrain our search to a specific class of functions parametrized by a parameter $x \in \mathbb{R}^n$.

For a more concrete example, imagine that when $d = 1$ we aim to find our predictor φ within the class of quadratic functions. Each quadratic function can be described by three parameters (p, q, r) as follows:

$$\varphi(a) = pa^2 + qa + r.$$

Therefore, finding φ is equivalent to finding the parameter vector $x = (p, q, r) \in \mathbb{R}^3$. This is what we mean by parametrization.

In the discussion above, we've chosen notation that aligns with optimization literature. In machine learning and statistics, it's more common to represent a dataset as (x_i, y_i) and parametrize the predictor φ with θ or w instead of x. However, this is a matter of notation, and the choice of variable names doesn't impact the concepts.

Least squares. The best known example of the above problem in statistics and data science is the least squares problem.

We have data points $(a_1, b_1), \dots, (a_m, b_m) \in \mathbb{R}^d \times \mathbb{R}$ and we want to solve

$$\min_x \frac{1}{2m}\|Ax - b\|^2 = \frac{1}{m}\sum_{i=1}^m \frac{1}{2}(a_i^\top x - b_i)^2.$$

In the previous notation, it means that we chose for φ the simplest function — linear, that is $\varphi(a_i) = a_i^\top x$, and the loss function is $\ell(\widehat{b}_i, b_i) = \frac{1}{2}(\widehat{b}_i - b_i)^2$. The least squares problem is also known as a *linear regression*.

In this type of problem, it's common to include a regularization term that encourages desirable properties for the predictor φ. For example, when adding an ℓ_2-norm penalty, we obtain the following optimization problem

$$\min_x \frac{1}{2m}\|Ax - b\|^2 + \frac{\lambda}{2}\|x\|^2.$$

Here, the parameter $\lambda > 0$ controls the strength of the regularization. This optimization problem is referred to as *ridge regression* and it's known to yield a more stable solution x with respect to perturbations in the data (a_i, b_i).

If we add ℓ_1-norm penalty, we get the *lasso* problem

$$\min_x \frac{1}{2m}\|Ax - b\|^2 + \lambda\|x\|_1.$$

Regularizer $\lambda\|x\|_1$ promotes sparsity, that is it makes a solution x to have more zeros.

Optimization Methods for Data Science (2023W) 23 Oct 2023

Lecture 4

Lecturer: Yurii Malitskyi

4 Stochastic Gradient Descent

Probability theory refresher

Let X be a random variable that takes values $x_1, \dots, x_m$ with probabilities $p_1, \dots, p_m$, respectively. Then the *expectation* of X (or *mean*) is

$$\mathbb{E}[X] = \sum_{i=1}^{m} p_i x_i.$$

In other words, if we sample X many-many times, then in average the value we get will be $\mathbb{E}[X]$. For example, if we roll a die with faces $1, \dots, 6$, in average we obtain 3.5.

It is easy to see that expectation is a linear operation, that is

$$\mathbb{E}[aX + bY] = a\mathbb{E}[X] + b\mathbb{E}[Y].$$

SGD

As before, we want to minimize $\min_{x \in \mathbb{R}^n} f(x)$ with a smooth function f. Suppose that instead of the exact value $\nabla f(x)$ at a given x, we can only access its stochastic approximation

$$g(x, \xi) = \nabla f(x) + \xi,$$

where ξ is a random vector (noise). This means that we have access to the *stochastic* oracle, but not to deterministic one. If $\mathbb{E}[g(x, \xi)] = \nabla f(x)$, we say that this estimate is *unbiased*.

By analogy with GD, stochastic gradient descent takes the form

$$\begin{aligned} x_{k+1} &= x_k - \alpha_k g(x_k, \xi_k) \\ &= x_k - \alpha_k (\nabla f(x_k) + \xi_k), \end{aligned} \tag{4.1}$$

where ξ_k is a noise in the k-th iteration. This idea goes back to Robbins & Monro, 1951, where they were interested in solving nonlinear equations with stochastic estimates.

What is our motivation to consider update (4.1)? It can be outlined as follows.

1. **Escaping saddle points:** Adding noise to the gradient estimate can help us escape saddle points. With deterministic first-order methods, once $\nabla f(x) = 0$, we may become stuck at a saddle point. However, when we introduce random noise to $\nabla f(x)$, there is a high probability that we will eventually discover a direction to descend, allowing us to move away from the saddle point.

2. **Practicality:** In some cases, we may not have access to a more accurate estimate of the gradient. For example, consider an objective function of the form:

   $$f(x) = \mathbb{E}_\xi[F(x, \xi)],$$

   where ξ is a random variable. In general, computing f and ∇f may be infeasible. Instead, we can work with the stochastic estimates of the latter:

   $$g(x, \xi) = \nabla F(x, \xi).$$

   Random noise is also may be added during gradients transmission or we can add it intentionally to protect our data.

3. **Better complexity in some cases:** This is the main motivation for us that we consider in detail.

Recall that the empirical risk minimization problem asks to find a predictor φ that is a solution of

$$\min_{\varphi} R(\varphi) := \frac{1}{m}\sum_{i=1}^{m} \ell(\varphi(a_i), b_i). \tag{4.2}$$

If we parametrize φ by some vector $x \in \mathbb{R}^n$ and denote $f_i(x) = \ell(\varphi(a_i), b_i)$, we arrive at our primary objective:

$$\min_{x\in\mathbb{R}^n} f(x) := \frac{1}{m}\sum_{i=1}^{m} f_i(x). \tag{4.3}$$

Assuming that f is L-smooth and convex, we can run our favorite GD on it. We know that in the worst case we need $O(\frac{1}{\varepsilon})$ iterations to reach ε-accuracy in terms of function values. What about complexity, that is how much will it cost us? One can measure complexity in various ways: arithmetic operations, matrix-vector multiplications, etc. But for us the most natural is the number of gradient evaluations. In particular, we will count the number of ∇f_i assuming that all $f_1, \dots, f_m$ incur similar costs. Evaluating one $\nabla f(x)$ costs m gradients ∇f_i. Since we need $O(\frac{1}{\varepsilon})$ iterations, the total complexity is

$$O\left(\frac{m}{\varepsilon}\right) \text{ of gradients } \nabla f_i.$$

If m is very large, $O(\frac{m}{\varepsilon})$ can become overwhelming. Thus, we desire a more favorable dependence on m, and that's where SGD comes into play.

Looking at problem (4.3), it is natural to choose a stochastic estimate of $\nabla f(x)$ as $\nabla f_i(x)$, where $i \in \{1, \dots, m\}$ is chosen uniformly at random. Thus, for problem (4.3), the main update (4.1) takes the form

Algorithm 4.1 Stochastic gradient descent

```
1: Input: $x_0 \in \mathbb{R}^n$, $(\alpha_k) \subset \mathbb{R}_{++}$
2: for $k = 0, 1, \dots$ do
3:     Sample $i_k \sim \text{Unif}\{1, \dots, m\}$
4:     Update $x_{k+1} = x_k - \alpha_k \nabla f_{i_k}(x_k)$
5: end for
```

It is easy to see that our estimate of the gradient is unbiased, that is

$$\mathbb{E}\left[\nabla f_{i_k}(x)\right] = \nabla f(x).$$

In this algorithm there are too many random variables $i_1, \dots, i_k$. They in turn make other vectors $x_1 \dots, x_k$ random as well. In order to distinguish about which randomness we talk, it is useful to introduce *conditional* expectation. In the k-th iteration, we can consider two types of expectations: one is conditioned on random variables $i_1, \dots, i_{k-1}$, denoted as $\mathbb{E}_k[\cdot]$, and another is total, or conditioned on all random variables that have appeared $i_1, \dots, i_k$, denoted simply as $\mathbb{E}[\cdot]$. In other words, when we take $\mathbb{E}_k[X]$, we only care about randomness caused by i_k and we treat all the previous information as deterministic. For us the following formula will be useful, known as the *law of total expectation*:

$$\mathbb{E}[X] = \mathbb{E}[\mathbb{E}_k[X]].$$

As a simple consequence, this formula says that SGD update is equivivalent to the GD one in expectation, that is

$$\mathbb{E}[x_{k+1}] = \mathbb{E}\left[\mathbb{E}_k\left[x_k - \alpha_k \nabla f_{i_k}(x_k)\right]\right] = \mathbb{E}[x_k - \alpha_k \nabla f(x_k)].$$

For brevity, we will write SGD update as

$$x_{k+1} = x_k - \alpha_k g_k, \quad \text{that is} \quad g_k := \nabla f_{i_k}(x_k).$$

Theorem 4.1 (Convex case). *Suppose that $f_i\colon \mathbb{R}^n \to \mathbb{R}$ is differentiable, f is convex, and $\mathbb{E}\left[\|g_k\|^2\right] \leqslant G^2$ for all k. Then for iterates of Algorithm 4.1 it holds*

$$\mathbb{E}[f(\hat{x}_K) - f_*] \leqslant \frac{\|x_0 - x^*\|^2}{2\sum_{k=0}^K \alpha_k} + \frac{\sum_{k=0}^K \alpha_k^2 G^2}{2\sum_{k=0}^K \alpha_k}, \tag{4.4}$$

where $\hat{x}_K = \frac{\alpha_0 x_1 + \cdots + \alpha_K x_k}{\alpha_0 + \cdots + \alpha_K}$.

Proof. Let's expand the norm

$$\|x_{k+1} - x^*\|^2 = \|x_k - x^* - \alpha_k g_k\|^2 = \|x_k - x^*\|^2 - 2\alpha_k \langle g_k, x_k - x^*\rangle + \alpha_k^2 \|g_k\|^2$$

By taking expectation from both sides and using that

$$\mathbb{E}[\langle g_k, x^* - x_k\rangle] = \mathbb{E}[\mathbb{E}_k[\langle g_k, x^* - x_k\rangle]] = \mathbb{E}[\langle \nabla f(x_k), x^* - x_k\rangle] \leqslant \mathbb{E}[f(x_k) - f_*],$$

we have that

$$2\alpha_k \mathbb{E}[f(x_k) - f_*] \leqslant \mathbb{E}\left[\|x_k - x^*\|^2\right] - \mathbb{E}\left[\|x_{k+1} - x^*\|^2\right] + \alpha_k^2 \mathbb{E}\left[\|g_k\|^2\right].$$

Summing this inequality over $k = 0, \ldots, K$ and using that $\mathbb{E}\left[\|g_k\|^2\right] \leqslant G^2$, we obtain

$$\begin{aligned} 2\sum_{k=0}^K \alpha_k \mathbb{E}[f(x_k) - f_*] &\leqslant \|x_0 - x^*\|^2 - \mathbb{E}\left[\|x_{k+1} - x^*\|^2\right] + \sum_{k=0}^K \alpha_k^2 \mathbb{E}\left[\|g_k\|^2\right] \\ &\leqslant \|x_0 - x^*\|^2 + G^2 \sum_{k=0}^K \alpha_k^2. \end{aligned}$$

Applying Jensen's inequality to the LHS, we get

$$\sum_{k=0}^K \alpha_k \mathbb{E}[f(x_k)] \geqslant \mathbb{E}\left[\sum_{k=0}^K \alpha_k f(\hat{x}_K)\right].$$

Hence,

$$\mathbb{E}[f(\hat{x}_K) - f_*] \leqslant \frac{\|x_0 - x^*\|^2}{2\sum_{k=0}^K \alpha_k} + \frac{\sum_{k=0}^K \alpha_k^2 G^2}{2\sum_{k=0}^K \alpha_k}.$$

■

Looking at (4.4), we see that it is desirable to ask (α_k) to satisfy

$$\sum_{k=0}^{+\infty} \alpha_k = +\infty \quad \text{and} \quad \frac{\sum_{k=0}^K \alpha_k^2}{\sum_{k=0}^K \alpha_k} \to 0.$$

It is not very difficult to prove that the "best" sequence (α_k) has to behave like $\alpha_k \sim \frac{1}{\sqrt{k}}$. In particular, for $\alpha_k = \frac{C}{\sqrt{k+1}}$ we have that

$$\sum_{k=0}^K \alpha_k > C\sqrt{K+1} > C\sqrt{K}$$

$$\sum_{k=0}^K \alpha_k^2 \approx C^2 \log(K+1).$$

To summarize, we obtain that

$$\mathbb{E}[f(\hat{x}_K) - f_*] \leqslant \frac{\|x_0 - x^*\|^2}{2C\sqrt{K}} + \frac{G^2 C^2 \log(K+1)}{2\sqrt{K}} = O\left(\frac{\log K}{\sqrt{K}}\right).$$

Of course, this result is in expectation, which is worse than the deterministic one. Now, let's compare complexity of GD and SGD. For the problem we study, GD, in order to reach ε-accuracy, needs $O\left(\frac{m}{\varepsilon}\right)$ evaluations of ∇f_i. At the same time, for SGD[4] we have $O\left(\frac{1}{\varepsilon^2}\right)$ evaluations of ∇f_i. This is the most striking about SGD: its total complexity doesn't depend on the number of data points m!

This comparison is not particularly rigorous because we used different assumptions for the analysis of GD and SGD. However, it's worth noting that we can also analyze SGD without the restrictive assumption $\mathbb{E}\left[\|g_k\|^2\right] \leqslant G^2$, relying solely on the L-smoothness of f instead. In this scenario, the analysis becomes slightly more intricate, but the fundamental conclusion remains the same: the complexity of SGD is $O\left(\frac{1}{\varepsilon^2}\right)$ in terms of the number of evaluations of ∇f_i. Aside from the specifics of the SGD update, this is a key takeaway from today's lecture.

Remark 1. The name "stochastic gradient descent" is a bit of a misnomer, there is usually no descent property; the name rather conveys that we made gradient descent method stochastic.

Mini-batching

Instead of sampling one i_k per iteration, we can draw a subset from $\{1, \dots, m\}$. This is known as *mini-batching*. In this case, the update is

```
for $k = 0, 1, \dots$
    Draw a random subset $S_k \subset \{1, \dots, m\}$
    $x_{k+1} = x_k - \alpha_k \frac{1}{|S_k|} \sum_{i \in S_k} \nabla f_i(x_k)$
```

Here $|S|$ denotes the number of elements in the set S.

- If $|S_k| = 1$, then it is the same SGD as we had before (Algorithm 4.1).
- If $|S_k| \approx m$, then it is known as a *full-batch* regime, which is close to the true GD update.
- If $|S_k| \ll m$, then it is known as a *mini-batch* regime.

In practice, determining the ideal mini-batch size is more of an art than a science.

We call an *epoch* a number of iterations that roughly represent m calculations of ∇f_i.

Exercises

Exercise 13. Why in the proof of Theorem 4.1 we didn't write $\mathbb{E}\left[\|x_0 - x^*\|^2\right]$?

Exercise 14. What is the complexity of SGD with mini-batching (a) $|S_k| = \frac{m}{100}$, (b) $|S_k| = 100$?

Exercise 15. Let $m = 10^6$. How many iterations (epochs) of SGD with mini-batching 100 do we need to approximate the cost of 1000 iterations of full-batch SGD?

[4]For simplicity we ignore $\log K$ dependency. Actually, there is a tighter analysis of SGD with decreasing steps without this log factor.

Optimization Methods for Data Science (2023W) 6-13 Nov 2023

Lecture 5

Lecturer: Yurii Malitskyi

5 Quadratic minimization

Consider the most important problem of numerical mathematics — a linear system

$$Ax = b, \tag{5.1}$$

where A is an $m \times n$ matrix and $b \in \mathbb{R}^m$. This linear system may have 0 solutions, 1 solution, or infinitely many. If $m > n$ the system is called *over-determined*, and if $m < n$ it is called *under-determined*. Given $x \in \mathbb{R}^n$, a vector $r = Ax - b$ is called a *residual*.

In general, for over-determined systems there is no solution — we have more constraints than variables. In other words, there is no x such that $r = Ax - b = 0$. In this case, the most natural generalization is to consider the *least squares* problem

$$\min_x f(x) := \frac{1}{2}\|Ax - b\|^2. \tag{5.2}$$

In other words, we want to find x such that the residual is as small as possible. Problem (5.2) is also called a *regression* problem. Of course, it is an instance of the quadratic minimization and after expanding the norm, we see that

$$f(x) = \frac{1}{2}\langle Qx, x\rangle - \langle c, x\rangle + \frac{1}{2}\|b\|^2,$$

where $Q = A^\top A$ and $c = A^\top b$. It is also useful to realize that

$$f(x) = \frac{1}{2}\|Ax - b\|^2 = \frac{1}{2}\sum_{i=1}^m (a_i^\top x - b_i)^2.$$

By definition, matrix $Q = A^\top A$ is positive semidefinite. This means that the problem (5.2) is convex. Let $0 \leqslant \lambda_1 \leqslant \ldots \leqslant \lambda_n$ be eigenvalues of Q. If $\lambda_1 > 0$, then Q is positive definite, and in this case f is λ_1-strongly convex. (Why?) For $\lambda_1 > 0$, $\kappa = \frac{\lambda_n}{\lambda_1}$ is called a *condition number* of $A^\top A$. The smaller this number is, the better numerical properties the matrix $A^\top A$ has. The matrix A with large κ is called *ill-conditioned*: small changes to elements of A or b lead to large changes in the solution x.

Before we proceed, recall some basic facts from linear algebra. We use the following notation

$$\ker(A) = \{x : Ax = 0\} \quad \text{and} \quad \operatorname{range}(A) = \{Ax : x \in \mathbb{R}^n\}.$$

The kernel is also called the null-space of the matrix and the range is called the column space. It shouldn't be difficult to show that $\ker(A)^\perp = \operatorname{range}(A^\top)$. We can define $\operatorname{rank}(A) = \dim(\operatorname{range}(A))$, that is the maximum number of linearly independent columns of A.

Lemma 5.1. For any matrix $A \in \mathbb{R}^{m\times n}$, one has

$$\ker(A^\top A) = \ker(A) \quad \text{and} \quad \operatorname{range}(A^\top A) = \operatorname{range}(A^\top).$$

Its proof is not important for us, but I include it in the case you have questions.

Proof. If $x \in \ker(A)$, then $Ax = 0$ and hence $A^\top Ax = 0$. Therefore, $x \in \ker(A^\top A)$. The opposite direction: let $x \in \ker(A^\top A)$. Then $A^\top Ax = 0$. By multiplying this equation by x, we obtain

$$\langle A^\top Ax, x\rangle = \langle Ax, Ax\rangle = 0.$$

Hence, $Ax = 0$ and $x \in \ker(A)$.

The second identity follows directly from the the first:

$$\operatorname{range}(A^\top A) = \ker(A^\top A)^\perp = \ker(A)^\perp = \operatorname{range}(A^\top).$$

■

Problem (5.2), in contrast to (5.1), always has a solution.

Theorem 5.1. *Problem (5.2) always has a solution. Every solution x^* is a solution of the normal equation*

$$A^\top Ax^* = A^\top b.$$

Moreover, if $m \geqslant n$ and $\operatorname{rank}(A) = n$, then the solution is unique:

$$x^* = (A^\top A)^{-1}A^\top b.$$

Proof. The objective f in (5.2) is convex and differentiable. Hence, x^* must be characterized by

$$\nabla f(x^*) = 0 \qquad \Longleftrightarrow \qquad A^\top(Ax^* - b) = 0.$$

The existence of x^* now follows from Lemma 5.1, since $A^\top b \in \operatorname{range}(A^\top A)$.

Let $A \in \mathbb{R}^{m\times n}$, $m \geqslant n$. Then from the same Lemma 5.1 it follows that $\operatorname{rank}(A^\top A) = \operatorname{rank}(A) = n$. This implies that the matrix $A^\top A$ is invertible. ■

If $m < n$, it is typical for a linear system to have infinitely many solutions. But which solution is the "best"? One possible way is to look at the solution with the smallest norm, that is to solve

$$\min_x \frac{1}{2}\|x\|^2 \quad \text{s.t.} \quad Ax = b. \tag{5.3}$$

This is our first instance of constrained optimization problem. And although the objective is convex and differentiable, the first-order optimality condition is different. Consider a more general problem

$$\min_x g(x) \quad \text{s.t.} \quad Ax = b. \tag{5.4}$$

Theorem 5.2. *Suppose that g is a differentiable function. If x^* is a local minimizer of (5.4), then*

$$\begin{aligned} Ax^* &= b \\ \nabla g(x^*) &\perp \ker(A). \end{aligned} \tag{5.5}$$

If g is also a convex function, then any x^* that satisfies above two conditions must be a solution.

Condition $\nabla g(x^*) \perp \ker(A)$ means that

$$\langle \nabla g(x^*), u\rangle = 0 \quad \text{for any } u \text{ s.t. } Au = 0.$$

Since, we know that $\ker(A)^\perp = \operatorname{range}(A^\top)$, we can rewrite the condition (5.5) as

$$\begin{aligned} Ax^* &= b \\ \nabla g(x^*) &\in \operatorname{range}(A^\top) \end{aligned} \tag{5.6}$$

Proof. Let x^* be a minimizer of (5.4). Then $Ax^* = b$. Suppose that the second condition doesn't hold, that is there exists $h \in \ker(A)$ such that $\langle \nabla g(x^*), h\rangle < 0$ (why can we assume this?). Then by differentiability of g, we have that

$$g(x^* + \alpha h) = g(x^*) + \alpha\langle \nabla g(x^*), h\rangle + o(\alpha).$$

Hence, for all α small enough we get $g(x^* + \alpha h) < g(x^*)$ that contradicts that x^* is a local minimizer.

Now assume that g is convex and that $Ax^* = b$, $\nabla g(x^*) \perp \ker(A)$. For any feasible x, that is such that $Ax = b$, we have that $x - x^* \in \ker(A)$. Then by inequality (1.9),

$$g(x) \geqslant g(x^*) + \langle \nabla g(x^*), x - x^*\rangle = g(x^*),$$

which concludes that x^* is a minimizer. ■

For simplicity, we assume that the matrix A is full rank, that is its all rows are linearly independent. Now we can apply the above theory to find the solution of (5.3). We write "the solution" because we know that $\|x\|^2$ is a strongly convex function, hence similarly as it was in the unconstrained case, the solution must be unique. Applying condition (5.6) to (5.3), we obtain

$$\begin{aligned} Ax^* &= b \\ x^* &= A^\top z \end{aligned}$$

for some z. Hence, $AA^\top z = b$. Since A is full rank, $AA^\top$ must be as well, and thus it is invertible. Therefore, $z = (AA^\top)^{-1} b$ and

$$x^* = A^\top (AA^\top)^{-1} b.$$

5.1 Algorithms

Now we are going to see how the algorithms we have learned behave on the least square problem

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

Also, let's summarize what we know about f. It is convex and differentiable with the gradient $\nabla f(x) = A^\top(Ax - b) = Qx - c$. If $\lambda_1(A^\top A) = \lambda_1(Q) > 0$, the function f is also λ_1-strongly convex. This follows from the identity

$$f(y) - f(x) = \langle \nabla f(x), y - x\rangle + \frac{1}{2}\|A(y-x)\|^2 = \langle \nabla f(x), y - x\rangle + \frac{1}{2}\langle Q(y-x), y-x\rangle. \quad (5.7)$$

How one can prove this identity literally without writing any formula?

Gradient descent. The gradient ∇f is L-Lipschitz with $L = \lambda_n(A^\top A) = \lambda_n(Q)$. We know from the general theory about GD that stepsize α_k must satisfy $\alpha_k \leqslant \frac{2}{L} = \frac{2}{\lambda_n}$. We will now show this additionally for quadratic case.

GD iteration is

$$\begin{aligned} x_{k+1} &= x_k - \alpha \nabla f(x_k) \\ &= x_k - \alpha A^\top (Ax_k - b). \end{aligned}$$

By Theorem 5.1, a solution x^* must satisfy $A^\top A x^* = A^\top b$. Hence,

$$\begin{aligned} x_{k+1} - x^* &= x_k - \alpha A^\top (Ax_k - b) - x^* \\ &= x_k - \alpha A^\top A(x_k - x^*) - x^* \\ &= (I - \alpha A^\top A)(x_k - x^*). \end{aligned}$$

From this we obtain that $\|x_{k+1}-x^*\| \leqslant \|I-\alpha_k A^\top A\|\|x_k-x^*\|$. If $\alpha_k < \frac{2}{L_f} = \frac{2}{\lambda_n}$.

Unrolling this recursion, we get

$$x_k - x^* = (I-\alpha Q)^k(x_0-x^*). \tag{5.8}$$

Consider two cases.

Case I: $\lambda_1 > 0$. In this case, f is strongly convex, hence x^* is the unique solution and we can expect $\|x_k-x^*\| \to 0$. We have

$$\|x_k-x^*\| \leqslant \|(I-\alpha Q)^k\|\|x_0-x^*\| = \|I-\alpha Q\|^k\|x_0-x^*\|.$$

By the definition of the spectral norm, for a symmetric matrix $I-\alpha Q$,

$$\|I-\alpha Q\| = \lambda_{\max}(|I-\alpha Q|) = \max_{\lambda_1\leqslant\lambda\leqslant\lambda_n}|1-\alpha\lambda|$$

Thus, we need to find α such that $\max_{\lambda_1\leqslant\lambda\leqslant\lambda_n}|1-\alpha\lambda| < 1$. Obviously, it is sufficient to consider two extreme cases for λ and thus, we obtain

$$\max\{|1-\alpha\lambda_1|, |1-\alpha\lambda_n|\}. \tag{5.9}$$

There are several popular choices for α:

- $\alpha = \frac{2}{\lambda_1+\lambda_n}$. This is an optimal choice for α that makes (5.9) smaller than 1. (To see this, just draw two plots: $\alpha \mapsto |1-\alpha\lambda_1|$ and $\alpha \mapsto |1-\alpha\lambda_n|$). With this choice

  $$\max\{|1-\alpha\lambda_1|, |1-\alpha\lambda_n|\} = \frac{\lambda_n-\lambda_1}{\lambda_1+\lambda_n} = \frac{\kappa-1}{\kappa+1} = 1-\frac{2}{\kappa+1}$$

  and we obtain

  $$\|x_k-x^*\| \leqslant \left(1-\frac{2}{\kappa+1}\right)^k\|x_0-x^*\|. \tag{5.10}$$

  The drawback of this choice is that we need to know both λ_1 and λ_n.

- $\alpha = \frac{1}{L} = \frac{1}{\lambda_n}$. With this choice, we get

  $$\|x_k-x^*\| \leqslant \left(1-\frac{1}{\kappa}\right)^k\|x_0-x^*\|, \tag{5.11}$$

  which is of the same order as (5.10).

- Steepest descent choice (see the paragraph in the end of this lecture for more details). Let $r_k = \nabla f(x_k) = Qx_k - c$ and $\alpha = \frac{\langle r_k, r_k\rangle}{\langle Qr_k, r_k\rangle}$. With this, we obtain the same inequality as in (5.10)

  $$\|x_k-x^*\| \leqslant \left(1-\frac{2}{\kappa+1}\right)^k\|x_0-x^*\|,$$

  but the advantage is that it does not require any knowledge about the eigenvalues of Q.

With any α as above, we obtain a linear rate for $\|x_k-x^*\|$. In particular, given accuracy $\varepsilon>0$, we can find the number of iterations that ensure $\|x_k-x^*\| \leqslant \varepsilon$ from

$$\|x_k-x^*\| \leqslant \left(1-\frac{1}{\kappa}\right)^k\|x_0-x^*\| \leqslant e^{-k/\kappa}\|x_0-x^*\| \leqslant \varepsilon,$$

which give us $k = \kappa\log\left(\frac{\|x_0-x^*\|}{\varepsilon}\right)$. Hence, at most after this number of iterations we can guarantee $\|x_k-x^*\| \leqslant \varepsilon$. If we care about the rate for function values, we can easily derive it by applying (5.7)

$$f(x)-f_* = \frac{1}{2}\|A(x-x^*)\|^2 = \frac{1}{2}\langle Q(x-x^*), x-x^*\rangle. \tag{5.12}$$

With it, we have

$$\begin{aligned}
f(x_k) - f_* &= \frac{1}{2}\langle Q(x_k - x^*), x_k - x^*\rangle \\
&= \frac{1}{2}\langle Q(I-\alpha Q)^k(x_0 - x^*), (I-\alpha Q)^k(x_0 - x^*)\rangle \\
&= \frac{1}{2}\langle (I-\alpha Q)^k Q(x_0 - x^*), (I-\alpha Q)^k(x_0 - x^*)\rangle \qquad \text{//Why?} \\
&= \frac{1}{2}\langle (I-\alpha Q)^{2k} Q(x_0 - x^*), x_0 - x^*\rangle \\
&\leqslant \|I-\alpha Q\|^{2k}(f(x_0) - f_*).
\end{aligned} \tag{5.13}$$

Thus, for any of α as above, we get at least

$$f(x_k) - f_* \leqslant \left(1 - \frac{1}{\kappa}\right)^{2k}(f(x_0) - f_*).$$

This is almost the same rate, as we obtained for GD in the case of the μ-strongly convex and L-smooth function. Why almost? Is there a contradiction?

Case II: $\lambda_1 = 0$. As we have already said, in this case we cannot obtain the rate for $x_k - x^*$, at least because x^* is not unique. To analyze the function values, we do the same as before and use that

$$\begin{aligned}
f(x_k) - f_* &= \frac{1}{2}\langle (I-\alpha Q)^{2k} Q(x_0 - x^*), x_0 - x^*\rangle \\
&\leqslant \|(I-\alpha Q)^{2k} Q\| \|x_0 - x^*\|.
\end{aligned}$$

Let $\alpha \leqslant \frac{1}{\lambda_n}$. We know that

$$\|(I-\alpha Q)^{2k} Q\| = \max_{0\leqslant\lambda\leqslant\lambda_n}(1-\alpha\lambda)^{2k}\lambda \leqslant e^{-2k\alpha\lambda}\lambda \leqslant \frac{1}{2ek\alpha} \leqslant \frac{1}{4k\alpha},$$

where the second last inequality follows simply from maximizing a scalar function $\lambda \mapsto e^{-2k\alpha\lambda}\lambda$. Therefore, we got

$$f(x_k) - f_* \leqslant \frac{1}{4k\alpha}\|x_k - x^*\|^2. \tag{5.14}$$

Is this consistent with the previous result for GD for L-smooth function?

Implicit bias of GD. By unrolling the recursion of GD, we have that

$$x_{k+1} - x_0 = x_k - x_0 - \alpha A^\top(Ax_k - b) = -\alpha\sum_{i=0}^{k} A^\top(Ax_i - b).$$

This means that $x_{k+1} - x_0 \in \operatorname{range}(A^\top)$, which in turn implies that $x_{k+1} - x_0 \perp \ker(A)$. From the general convergence of GD, for any $\alpha < \frac{2}{\lambda_n}$, the sequence (x_k) converges to some solution $\hat{x}$, so that $AA^\top\hat{x} = A^\top b$. Since for any k, $x_{k+1} - x_0 \perp \ker(A)$, by tending $k \to \infty$ we obtain that

$$\hat{x} - x_0 \perp \ker(A).$$

Let's summarize what we obtain. On one hand, $\hat{x}$ is a solution of the least squares problem, that is $A^\top A\hat{x} = A^\top b$. On the other, $\hat{x} - x_0 \perp \ker(A)$. If we consider the problem

$$\min_x \|x - x_0\|^2 \quad \text{s.t.} \quad A^\top Ax = A^\top b,$$

by Theorem 5.2, $\hat{x}$ must be its solution. This is quite a remarkable result: among all solutions of (5.2), the iterates of GD choose the closest point to x_0 to converge. In machine learning literature this phenomenon is called an *implicit bias*: without our intervention iterates choose "the best" point to converge.

Steepest Descent. For GD, there is a particular way to choose stepsize α_k in every iteration

$$\alpha_k = \min_{\alpha} f(x_k - \alpha \nabla f(x_k)).$$

In general, this is not a very good idea, since finding α_k is already a non-trivial problem and we have to solve it in every iteration. But for the case of quadratic minimization it can be done efficiently.

For a quadratic problem when $f(x) = \frac{1}{2}\langle Qx, x\rangle - \langle c, x\rangle$ with $Q \succ 0$, the subproblem to find α_k can be solved analytically:

$$\langle \nabla f(x_k - \alpha \nabla f(x_k)), \nabla f(x_k)\rangle = 0,$$

which gives us an explicit expression $\alpha = \frac{\langle r_k, r_k\rangle}{\langle Q r_k, r_k\rangle}$, where we denote $r_k = \nabla f(x_k) = Qx_k - c$.

The complete algorithm is given in Algorithm 5.1.

Algorithm 5.1 Steepest descent for quadratic problem

```
1: Input: $x_0 \in \mathbb{R}^n$.
2: for $k = 0, 1 \ldots$ do
3:     $r_k = Qx_k - c$
4:     $\alpha_k = \frac{\langle r_k, r_k\rangle}{\langle Q r_k, r_k\rangle}$
5:     $x_{k+1} = x_k - \alpha_k r_k$
6: end for
```

Exercise 16. How to implement this algorithm efficiently? In the form above each iteration requires two matrix-vector multiplications. Can we reduce this number?

Optimization Methods for Data Science (2023W) 20 Nov 2023

Lecture 6

Lecturer: Yurii Malitskyi

6 Newton method

Newton's method, also known as the Newton-Raphson method, was originally proposed for solving nonlinear equations. But since unconstrained optimization can be casted as one of this via $\nabla f(x) = 0$, it can be equally well applied for optimization problems. It is a *local* method, meaning it converges only in some neighborhood of a solution. But if it converges, it converges very fast.

As we have already seen for the gradient method, a good idea for minimizing a function f is to build some approximation of f around current point x_k. This time, let's consider the second-order approximation around x_k

$$f(x) \approx f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{2}\langle \nabla^2 f(x_k)(x - x_k), x - x_k \rangle =: f_k(x). \tag{6.1}$$

Minimizing f_k instead of f yields us a new method

$$x_{k+1} = \operatorname*{argmin}_x f_k(x).$$

We didn't need to extra regularize it as before for GD, since the quadratic term $\langle \nabla^2 f(x_k)(x - x_k), x - x_k \rangle$ ensures existence of a solution and with $\nabla^2 f(x_k) \succ 0$ we will have its uniqueness.

Now writing explicitly, we obtain Newton's method

$$x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k) \tag{6.2}$$

Original motivation for Newton's method comes from solving nonlinear equations in $\mathbb{R}$. Suppose $p \colon \mathbb{R} \to \mathbb{R}$ and we are seeking a solution of $p(t) = 0$. The Newton method is based on a linear approximation of p. If t is close to the current iterate t_k, then we can write

$$p(t) = p(t_k) + p'(t_k)(t - t_k) + o(|t - t_k|).$$

Hence, equation $p(t) = 0$ can be approximated by $p(t_k) + p'(t_k)(t - t_k) = 0$. Solving the latter leads to the iterative scheme

$$t_{k+1} = t_k - \frac{p(t_k)}{p'(t_k)}. \tag{6.3}$$

By the same logic, we can extend this scheme to the vector case $F(x) = 0$ with $F \colon \mathbb{R}^n \to \mathbb{R}^n$ as

$$x_{k+1} = x_k - [F(x_k)]^{-1} F(x_k). \tag{6.4}$$

We could directly use the latter scheme to obtain (6.2), since unconstrained minimization is equivalent to a nonlinear vector equation $\nabla f(x) = 0$.

Example 1. *Let us approximate $\sqrt{2}$. We apply Newton's method to solve $x^2 - 2 = 0$. Let $f(x) = x^2 - 2$ and $x_0 = 1$. Then*

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{x_k^2 - 2}{2x_k} = \frac{x_k}{2} + \frac{1}{x_k}.$$

This is the basis of how calculators compute $\sqrt{a}$.

Newton's method as steepest descent in local norm. Suppose that f is strictly convex. We have already seen that GD is characterized by the direction

$$d = \operatorname*{argmin}_{\|u\|=1} \langle \nabla f(x), u\rangle.$$

In this definition, we use the standard Euclidean norm $\|\cdot\|$ induced by the inner product $\langle\cdot,\cdot\rangle$. However, minimizing some abstract function f has little to do with $\|\cdot\|$ and $\langle\cdot,\cdot\rangle$. Instead, it is tempting to use norm, tailored to the geometry of f at given point x. This leads to the definition of *local norm*:

$$\langle u, v\rangle_x := \langle \nabla^2 f(x)u, v\rangle \qquad \|u\|_x := \langle u, u\rangle_x^{1/2} = (\langle \nabla^2 f(x)u, u\rangle)^{1/2}.$$

Now let us define the direction d as

$$d = \operatorname*{argmin}_{\|u\|_x=1} \langle \nabla f(x), u\rangle.$$

We claim that $d = -\alpha[\nabla^2 f(x)]^{-1}\nabla f(x)$ with $\alpha > 0$ that normalizes d. Indeed, with a shortcut $H = \nabla^2 f(x)$ we have

$$\langle \nabla f(x), u\rangle = \langle H^{1/2}H^{-1}\nabla f(x), H^{1/2}u\rangle \geqslant -\|H^{1/2}H^{-1}\nabla f(x)\| \cdot \|H^{1/2}u\| = -\|H^{-1}\nabla f(x)\|_x \cdot \|u\|_x,$$

with an equality if and only if $H^{1/2}H^{-1}\nabla f(x) = -\alpha H^{1/2}u$ for some $\alpha > 0$. This proves the above claim.

Preliminaries. Let's recall some standard inequalities for the second-order Taylor approximation.

Lemma 6.1. Suppose that $\nabla^2 f$ is H-Lipschitz, that is

$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \leqslant H\|x - y\|.$$

Then

$$\|\nabla f(y) - \nabla f(x) - \nabla^2 f(x)(y - x)\| \leqslant \frac{H}{2}\|x - y\|^2.$$

Its proof is not essential for us, but I put it here for completeness. In general, the proof is very similar to the one of decent lemma.

Proof. In the vector case, for a differentiable $g : \mathbb{R}^n \to \mathbb{R}^n$ we have

$$g(x + y) = g(x) + \int_0^1 g'(x + \tau y)y \, d\tau.$$

Since

$$\begin{aligned}\nabla f(x + y) &= \nabla f(x) + \int_0^1 \nabla^2 f(x + \tau y)y \, d\tau \\ &= \nabla f(x) + \nabla^2 f(x)y + \int_0^1 \left(\nabla^2 f(x + \tau y) - \nabla^2 f(x)\right) y \, d\tau,\end{aligned}$$

we have that

$$\begin{aligned}\left\|\nabla f(x + y) - \nabla f(x) - \nabla^2 f(x)y\right\| &\leqslant \left\|\int_0^1 \left(\nabla^2 f(x + \tau y) - \nabla^2 f(x)\right) y \, d\tau\right\| \\ &\leqslant \int_0^1 H\tau\|y\|^2 \, d\tau \\ &= \frac{H}{2}\|y\|^2.\end{aligned}$$

Redefining y to $y - x$ gives the desired. ∎

Convergence.

Theorem 6.1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be μ-strongly convex and twice differentible and with $\nabla^2 f$ being H-Lipschitz. Suppose that initial iterate x_0 satisfies*

$$q = \frac{H}{2\mu^2}\|\nabla f(x_0)\| < 1.$$

Then Newton's method converges to the global minimum x^* quadratically:

$$\|x_k - x^*\| \leqslant \frac{2\mu}{H} q^{2^k}.$$

Proof. By Lipschitzness of $\nabla^2 f$, we have that

$$\|\nabla f(x_{k+1}) - \nabla f(x_k) - \nabla^2 f(x_k)(x_{k+1} - x_k)\| \leqslant \frac{H}{2}\|x_{k+1} - x_k\|^2$$

and taking into account the definition of x_{k+1}, we can write it as

$$\|\nabla f(x_{k+1})\| \leqslant \frac{H}{2}\|[\nabla^2 f(x_k)]^{-1}\nabla f(x_k)\|^2.$$

From strong convexity it follows that $\nabla^2 f(x_k) \succcurlyeq \mu I$ or $[\nabla^2 f(x_k)]^{-1} \preccurlyeq \frac{1}{\mu} I$. Hence,

$$\|\nabla f(x_{k+1})\| \leqslant \frac{H}{2\mu^2}\|\nabla f(x_k)\|^2.$$

Iterating inequality $a_{k+1} \leqslant b a_k^2$ gives us $b a_{k+1} \leqslant (b a_k)^2 \leqslant \ldots \leqslant (b a_0)^{2^{k+1}}$. Therefore, we conclude that

$$\|\nabla f(x_k)\| \leqslant \frac{2\mu^2}{H} q^{2^k}.$$

It only remains to apply $\|\nabla f(x_k)\| \geqslant \mu\|x_k - x_*\|$ (why is this true?). ∎

Affine invariance. Another thing that makes the Newton method great is its affine invariance with respect to linear transformation of the variables. Let A be a nonsingular $n \times n$ matrix and consider $\varphi(y) = f(Ay)$. For Newton's method there is no difference which function to minimize — f or φ.

Lemma 6.2. Let (x_k) be a sequence generated by Newton's method for f:

$$x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1}\nabla f(x_k).$$

Let (y_k) be a sequence generated by Newton's method for φ:

$$y_{k+1} = y_k - [\nabla^2 \varphi(y_k)]^{-1}\nabla \varphi(y_k)$$

with $y_0 = A^{-1}x_0$. Then $y_k = A^{-1}x_k$ for all $k \geqslant 0$.

Exercises

Exercise 17. Verify that minimimization of $f_k(x)$ in (6.1) indeed leads to (6.2).

Exercise 18. Let x^* be a minimizer of the μ-strongly convex function f. Then

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

Exercise 19. Apply Newton's method for minimizing $f(x) = \sqrt{1 + x^2}$.

Exercise 20. Solve a system of equations by Newton's method

$$\begin{cases} 3x^2 y + y^2 &= 1 \\ x^4 + xy^3 &= 1 \end{cases}$$

Exercise 21. Let $p \in \mathbb{R}[x]$ be a polynomial with real coefficients whose roots are all real $u_{\max} \geqslant \ldots \geqslant u_{\min}$. Prove that if we initialize $x_0 \geqslant u_{\max}$, then the Newton method applied to $p(x) = 0$ converges to $u_{\max}$.

Optimization Methods for Data Science (2023W) 27 Nov 2023

Lecture 7

Lecturer: Yurii Malitskyi

7 Quasi-Newton methods

Newton's method has great advantages over GD in terms of speed, but also a major disadvantage — the need to know and invert Hessians in every iteration. Quasi-Newton methods are designed to mitigate the latter issue. These are methods with a variable metric

$$x_{k+1} = x_k - \alpha_k H_k \nabla f(x_k), \tag{7.1}$$

where a $n \times n$ matrix H_k is updated in a certain way.

Consider two approximations of f at x_k:

$$\begin{aligned}\varphi_{\mathrm{GD}}(x) &= f(x_k) + \langle \nabla f(x_k), x - x_k\rangle + \frac{1}{2}\langle x - x_k, x - x_k\rangle \\ \varphi_{\mathrm{NM}}(x) &= f(x_k) + \langle \nabla f(x_k), x - x_k\rangle + \frac{1}{2}\langle \nabla^2 f(x_k)(x - x_k), x - x_k\rangle\end{aligned}$$

As we have seen, minimizing φ_{GD} leads to the gradient descent and minimizing φ_{NM} — to the Newton method. Now let us devise another quadratic approximation of f around x_k, which are better than φ_{GD} and less expensive than φ_{NM}. Let an $n \times n$ matrix $G_k \succ 0$ and consider

$$\varphi_k(x) = f(x_k) + \langle \nabla f(x_k), x - x_k\rangle + \frac{1}{2}\langle G_k(x - x_k), x - x_k\rangle.$$

Then minimizing φ_k leads to

$$x_{k+1} = x_k - G_k^{-1}\nabla f(x_k).$$

Such methods are called *variable metric* methods. Ideally, we would like to have $G_k - \nabla^2 f(x_k) \to 0$.

Let $H_k = G_k^{-1}$. We can formulate the basic variable metric method as

$x_{k+1} = x_k - \alpha_k H_k \nabla f(x_k)$,
where α_k is found by some linesearch
Update H_k to H_{k+1}

Hence, to be fully defined, such methods require two things: stepsizes α_k and the way to update H_k to H_{k+1}.

Quasi-Newton methods are a particular case of variable metric methods with several options to update H_k. A sensible requirement from our model φ_{k+1}

$$\varphi_{k+1}(x) = f(x_{k+1}) + \langle \nabla f(x_{k+1}), x - x_{k+1}\rangle + \frac{1}{2}\langle G_{k+1}(x - x_{k+1}), x - x_{k+1}\rangle$$

can be the guarantee that gradients of f and φ_{k+1} coincide at $x = x_k$ and $x = x_{k+1}$. This leads to the condition

$$\nabla f(x_{k+1}) - \nabla f(x_k) = G_{k+1}(x_{k+1} - x_k).$$

Now since $G_k^{-1} = H_k$, we have that

$$H_{k+1}(\nabla f(x_{k+1}) - \nabla f(x_k)) = x_{k+1} - x_k. \tag{7.2}$$

This is known as the *secant equation* or *quasi-Newton rule*.

For brevity, let $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ and $s_k = x_{k+1} - x_k$. That is the secant equation is $H_{k+1}y_k = s_k$. Notice that for a non-convex f this rule is problematic, since for $H_k \succ 0$ and $y_k \neq 0$, we have

$$0 < \langle H_{k+1}y_k, y_k\rangle = \langle s_k, y_k\rangle = \langle \nabla f(x_{k+1}) - \nabla f(x_k), x_{k+1} - x_k\rangle,$$

which we cannot guarantee in general.

What do we want from the update of H_k? We boldly ask for it to be simple, cheap, and effective. Several options are available.

Rank-one update. Arguably, the simplest update is

$$H_{k+1} = H_k + auu^\top.$$

From the quasi-Newton rule $H_{k+1}y_k = s_k$, it follows

$$H_k y_k + auu^\top y_k = s_k.$$

As $uu^\top y_k = \langle u, y_k\rangle u$, it must be that u and $s_k - H_k y_k$ are parallel. And since we have an extra scalar factor a, we can just set $u = s_k - H_k y_k$. Then a is given by $a = \frac{1}{u^\top y_k} = \frac{1}{y_k^\top(s_k - H_k y_k)}$. Finally, we have

$$\text{(rank-one update)} \qquad \boxed{H_{k+1} = H_k + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^\top}{y_k^\top(s_k - H_k y_k)}.} \tag{7.3}$$

Unfortunately, this update has some drawbacks:

(i) H_k is not necessary positive definite;

(ii) Denominator may be zero.

Davidon-Fletcher-Powell update. Alas, the one-rank update was not very successful, so we do the most natural think — consider a two-rank update:

$$H_{k+1} = H_k + auu^\top + bvv^\top.$$

Again we find a, b, u, v from the quasi-Newton rule $H_{k+1}y_k = s_k$. This time, we have more flexibility in how to choose u and v, but to keep it simple, we set $u = s_k$ and $v = H_k y_k$. Then we can find a and b from $au^\top s_k = 1$ and $bv^\top s_k = -1$ and obtain the final update as

$$\text{(DFP update)} \qquad \boxed{H_{k+1} = H_k - \frac{H_k y_k y_k^\top H_k}{y_k^\top H_k y_k} + \frac{s_k s_k^\top}{y_k^\top s_k}.} \tag{7.4}$$

This update is known as the Davidon-Fletcher-Powell update. In this case positive-definiteness of H_k can be established.

Lemma 7.1. If $H_0 \succ 0$ and $\langle y_k, s_k\rangle > 0$ for all k, then DFP update guarantees that $H_k \succ 0$ for all k.

As we have already discussed, the condition $\langle y_k, s_k\rangle > 0$ is quite sensible, as it holds for any strictly convex function.

Proof. We use induction. Suppose $H_k \succ 0$ and let $H_k = LL^\top$. Then, for $x \neq 0$ we have

$$\begin{aligned}\langle x, H_{k+1}x\rangle &= \langle x, H_k x\rangle - \frac{x^\top H_k y_k y_k^\top H_k x}{y_k^\top H_k y_k} + \frac{x^\top s_k s_k^\top x}{y_k^\top s_k} \\ &= \|Lx\|^2 - \frac{\langle Lx, Ly_k\rangle^2}{\|Ly_k\|^2} + \frac{(x^\top s_k)^2}{y_k^\top s_k} \geqslant 0,\end{aligned}$$

where we applied the Cauchy-Schwarz inequality. In fact, we have strict inequality, because equality can only happen when $Lx = tLy_k$ for $t > 0$ (equality for Cauchy-Schwarz) and $x^\top s_k = 0$ (the last term is zero), which together lead to $x = 0$. Hence, H_{k+1} is indeed positive definite. ∎

Broyden-Fletcher-Goldfarb-Shanno update. Strangely, but we can get good results using the same logic to update G_k instead. Let

$$G_{k+1} = G_k + auu^\top + bvv^\top.$$

Since the quasi-Newton rule can be written with G_k as $G_{k+1}s_k = y_k$, we get the dual update of G_k similar to (7.4)

$$G_{k+1} = G_k - \frac{G_k s_k s_k^\top G_k}{s_k^\top G_k s_k} + \frac{y_k y_k^\top}{y_k^\top s_k}. \tag{7.5}$$

Now the most interesting bit: how to recover the matrix H_k? After all, we do not want to solve a linear system $G_k d_k = \nabla f(x_k)$.

Lemma 7.2 (Sherman–Morrison formula)**.** If $A \in \mathbb{R}^{n\times n}$ is an invertible matrix and $u, v \in \mathbb{R}^n$, then $A + uv^\top$ is invertible iff $1 + v^\top A^{-1}u \neq 0$ and in this case

$$(A+uv^\top)^{-1} = A^{-1} - \frac{A^{-1}uv^\top A^{-1}}{1+v^\top A^{-1}u}$$

The proof of this lemma is not important for us, I put it here for completeness.

Proof. It is easy to check this identity with a simple algebra. But it is not very instructive in my opinion. Let's think how we can come up with such an identity.

First, notice that since

$$(A+uv^\top)^{-1} = (I + A^{-1}uv^\top)^{-1}A^{-1},$$

for simplicity we can set $A := I$ and $u := A^{-1}u$. Thus, we must prove that

$$(I+uv^\top)^{-1} = I - \frac{uv^\top}{1+v^\top u}.$$

The eigenvalues of $I + uv^\top$ are $(1 + v^\top u)$ and $n-1$ times 1, which correspond to the eigenvector u and any other $n-1$ vectors $u_1, \dots, u_{n-1}$ which form the basis in the subspace orthogonal to u. Since eigenvectors are preserved under inversion, by the same logic we should expect to have $(I + uv^\top)^{-1} = I + \alpha uv^\top$. This is true, because a matrix is fully characterized by its action on eigenvectors. For $u_1, \dots, u_{n-1}$ this is obvious and the last eigenvector provides us condition on α. Indeed, we should have

$$\left(I + \alpha uv^\top\right)u = (1 + \alpha v^\top u)u = \frac{1}{1+u^\top v}u,$$

from which we deduce that $\alpha = -\frac{1}{1+v^\top u}$. ∎

In other words, if we know A^{-1}, then it is easy to compute the inverse of rank-1 perturbation of A. Most importantly, it says that the rank-1 update is preserved under inversion. This is true because $A^{-1}uv^\top A^{-1} = (A^{-1}u)(A^{-\top}v)^\top$ is a rank-1 matrix.

Since G_{k+1} is a rank-2 update of G_k, we have to apply Sherman-Morrison identity two times and overall computation will be a bit cumbersome. Instead we use the following generalization of Woodbury.

Lemma 7.3 (Sherman–Morrison-Woodbury formula)**.** Let $A \in \mathbb{R}^{n\times n}$ be an invertible matrix and U, C, V be conformable matrices to have $A + UCV$ well-defined. Then $A + UCV$ is invertible iff $C^{-1} + VA^{-1}U$ is nonsingular and in this case

$$(A+UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}.$$

Again, its proof is not important for us, it is just a simple algebraic manipulation.

Proof. Algebraic proof. Since

$$(I+A^{-1}UCV)^{-1}A^{-1} = \big(I - A^{-1}U(I+(CV)(A^{-1}U))^{-1}CV\big)A^{-1}.$$

it is sufficient to prove a particular case where we set $A := I$, $U := A^{-1}U$, and $V := CV$, that is

$$(I+UV)^{-1} = I - U(I+VU)^{-1}V.$$

For $U \in \mathbb{R}^{m\times n}$ and $V \in \mathbb{R}^{n\times m}$ we have the following chain of straightforward identities:

$$(UV)U = U(VU) \implies (I+UV)U = U(I+VU) \implies U(I+VU)^{-1} = (I+UV)^{-1}U \implies$$
$$U(I+VU)^{-1}V = (I+UV)^{-1}UV = (I+UV)^{-1}(I+UV-I) = I-(I+UV)^{-1}$$

and the proof is complete. ■

With such an identity, we proceed to derive an explicit form for H_k update. Be careful, next derivations are only if you're really passionate about math.

We drop all indices k for simplicity and write the update for G_k as

$$G_{k+1} = G - \frac{Gss^\top G}{s^\top Gs} + \frac{yy^\top}{y^\top s} = G + \underbrace{\begin{bmatrix} Gs & y \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} -\frac{1}{s^\top Gs} & 0 \\ 0 & \frac{1}{y^\top s} \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} s^\top G \\ y^\top \end{bmatrix}}_{V}.$$

Then using $HG = GH = I$ many times below, we have

$$\begin{aligned}
H_{k+1} &= H - H\begin{bmatrix} Gs & y \end{bmatrix}\left(\begin{bmatrix} -s^\top Gs & 0 \\ 0 & y^\top s \end{bmatrix} + \begin{bmatrix} s^\top G \\ y^\top \end{bmatrix} H \begin{bmatrix} Gs & y \end{bmatrix}\right)^{-1} \begin{bmatrix} s^\top G \\ y^\top \end{bmatrix} H \\
&= H - \begin{bmatrix} s & Hy \end{bmatrix}\left(\begin{bmatrix} -s^\top Gs & 0 \\ 0 & y^\top s \end{bmatrix} + \begin{bmatrix} s^\top G \\ y^\top \end{bmatrix}\begin{bmatrix} s & Hy \end{bmatrix}\right)^{-1} \begin{bmatrix} s^\top \\ y^\top H \end{bmatrix} \\
&= H - \begin{bmatrix} s & Hy \end{bmatrix}\begin{bmatrix} 0 & s^\top y \\ s^\top y & s^\top y + y^\top Hy \end{bmatrix}^{-1} \begin{bmatrix} s^\top \\ y^\top H \end{bmatrix} \\
&= H + \frac{1}{(s^\top y)^2}\begin{bmatrix} s & Hy \end{bmatrix}\begin{bmatrix} s^\top y + y^\top Hy & -s^\top y \\ -s^\top y & 0 \end{bmatrix}\begin{bmatrix} s^\top \\ y^\top H \end{bmatrix} \qquad \text{//inverse of } 2\times 2 \text{ matrix} \\
&= H + \frac{1}{(s^\top y)^2}\begin{bmatrix} s & Hy \end{bmatrix}\begin{bmatrix} s^\top y s^\top + y^\top Hys^\top - s^\top yy^\top H \\ -s^\top ys^\top \end{bmatrix} \\
&= H + \frac{1}{s^\top y}(ss^\top - sy^\top H - Hys^\top) + \frac{1}{(s^\top y)^2} sy^\top Hys^\top \\
&= H + \left(1 + \frac{y^\top Hy}{s^\top y}\right)\frac{ss^\top}{s^\top y} - \frac{sy^\top H + Hys^\top}{s^\top y} \\
&= \left(I - \frac{sy^\top}{s^\top y}\right) H \left(I - \frac{sy^\top}{s^\top y}\right) + \frac{ss^\top}{s^\top y},
\end{aligned}$$

where the last two equations are mere rewriting. Both appear in various textbooks. We can summarize it as

$$\text{(BFGS update)} \qquad \boxed{H_{k+1} = \left(I - \frac{s_k y_k^\top}{s_k^\top y_k}\right) H_k \left(I - \frac{s_k y_k^\top}{s_k^\top y_k}\right) + \frac{s_k s_k^\top}{s_k^\top y_k}.} \qquad (7.6)$$

7.1 Stepsize

We have learned a few ways how to update the matrix H_k. The last piece is how to find the stepsize α_k. Our update is $x_{k+1} = x_k - \alpha_k H_k \nabla f(x_k) = x_k - \alpha_k d_k$, with $d_k = H_k \nabla f(x_k)$. The most common condition for it is the Wolfe condition, which tells us that in the k-th iteration we have to choose α_k such that

$$\begin{aligned} f(x_k - \alpha_k d_k) &\leqslant f(x_k) - c_1 \alpha_k \langle \nabla f(x_k), d_k \rangle \quad \text{and} \\ \langle \nabla f(x_k - \alpha_k d_k), d_k \rangle &\leqslant c_2 \langle \nabla f(x_k), d_k \rangle \end{aligned}$$

for some $0 < c_1 < c_2 < 1$. In practice we run linesearch to find α_k. It means that we start from some value α and decrease it in some way until the above condition is met; this last value of α is our α_k.

Exercises

Exercise 22. Prove that if H_k is positive definite, then the descent property holds, that is there exist $\alpha > 0$ such that $f(x_k - \alpha H_k \nabla f(x_k)) \leqslant f(x_k)$.

Exercise 23. Prove that for a strictly convex differentiable function f one has

$$\langle \nabla f(x) - \nabla f(y), x - y \rangle > 0, \qquad \forall x, y.$$

Optimization Methods for Data Science (2023W) 4 Dec 2023

Lecture 8

Lecturer: Yurii Malitskyi

8 Algorithms for deep learning

Our problem of interest is $\min_x f(x)$, where sometimes we suppose that

For this problem we know gradient descent, heavy-ball method, and accelerated gradient method. Let's recall them.
Gradient descent:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k). \tag{8.1}$$

Heavy-ball method:

$$x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta(x_k - x_{k-1}). \tag{8.2}$$

Accelerated GD:

$$\begin{aligned} y_k &= x_k + \beta_k(x_k - x_{k-1}) \\ x_{k+1} &= y_k - \alpha \nabla f(y_k), \end{aligned} \tag{8.3}$$

Vector $\beta(x_k - x_{k-1})$ is known as *momentum*.

I won't bother you by repeating the theoretical guarantees for the above methods. We should only remember that at least in the convex case, adding momentum often helps.

Now if the objective f is given by

$$f(x) = \frac{1}{m}\sum_{i=1}^{m} f_i(x)$$

with large f, computing a full ∇f may be expensive and thus we would rather use SGD:

$$\begin{aligned} i_k &\sim \operatorname{Unif}\{1, \dots, m\} \\ x_{k+1} &= x_k - \alpha_k \nabla f_{i_k}(x_k). \end{aligned} \tag{8.4}$$

From this moment, we will use notation g_k for the current "gradient" vector, whether it is stochastic or not. For instance, SGD main update will be written as

$$x_{k+1} = x_k - \alpha_k g_k.$$

Notation g_k is more or less standard in the ML literature, but the main iterate usually is denoted by letters θ or w instead of x.

For simplicity, here and below we sample only one index i_k, that is our mini-batch is of size 1. In practice, usually one takes a larger mini-batch. This means that we sample a random subset $S_k \in \{1, \dots, m\}$ and define g_k as

$$g_k = \frac{1}{|S_k|}\sum_{i \in S_k} \nabla f_i(x_k).$$

Now, an obvious idea is to try to incorporate "momentum" into SGD. This leads to the SGD with momentum, also known as stochastic heavy ball method.

$$\begin{aligned} i_k &\sim \operatorname{Unif}\{1, \dots, m\} \\ g_k &= \nabla f_{i_k}(x_k) \\ x_{k+1} &= x_k - \alpha_k g_k + \beta_k(x_k - x_{k-1}). \end{aligned} \tag{8.5}$$

For some unclear reasons to me, you rarely see this method written in the form as above. Instead, it is more often given by

$$\begin{aligned} i_k &\sim \mathrm{Unif}\{1,\dots,m\} \\ g_k &= \nabla f_{i_k}(x_k) \\ m_k &= \beta_k m_{k-1} + g_k \\ x_{k+1} &= x_k - \alpha_k m_k, \end{aligned} \tag{8.6}$$

where $m_0 = 0$.

Let's prove that these forms are indeed equivalent. We use $x'_k, \alpha'_k, \beta'_k$ to denote all the ingredients of (8.6), so that we can distinguish them from the same in (8.5).

Lemma 8.1. Algorithms (8.5) and (8.6) are equivalent. In particular, if $x_0 = x'_0$ is the same initial point, all i_k are the same for both algorithms, and $\alpha'_k = \alpha_k$, $\beta'_k = \frac{\alpha_{k-1}\beta_k}{\alpha_k}$, then $x'_k = x_k$ for all k.

Proof. Suppose that $x'_t = x_t$, for $t = 1,\dots,k$. let's prove this for the $k+1$-th iteration:

$$\begin{aligned} x'_{k+1} &= x_k - \alpha'_k m_k = x_k - \alpha'_k(\beta'_k m_{k-1} + g_k) = x_k - \alpha'_k g_k - \frac{\alpha'_k\beta'_k}{\alpha'_{k-1}}(x_{k-1} - x_k) \\ &= x_k - \alpha_k g_k + \beta_k(x_k - x_{k-1}) = x_{k+1}, \end{aligned}$$

where we used that $\alpha'_k = \alpha_k$ and $\beta'_k = \frac{\alpha_{k-1}\beta_k}{\alpha_k}$. ■

To summarise, **idea 1** is to use momentum in SGD.

AdaGrad. We have already seen that enhancing first-order methods by some additional matrix, such as

$$x_{k+1} = x_k - \alpha G_k^{-1} g_k$$

may be advantageous in terms of convergence. Such matrix G is called a preconditioner, meaning that we use it to decrease the condition number of f. As we know, the best preconditioner is the Hessian: $G_k = \nabla^2 f(x)$, but this would lead to an expensive iteration. AdaGrad suggest to do the following[5]

$$\begin{aligned} i_k &\sim \mathrm{Unif}\{1,\dots,m\} \\ g_k &= \nabla f_{i_k}(x_k) \\ G_k &= G_{k-1} + g_k g_k^\top \\ x_{k+1} &= x_k - \alpha G_k^{-1/2} g_k, \end{aligned} \tag{8.7}$$

where $G_{-1} = \varepsilon I$ is an ε-scaling of the identity matrix, $\varepsilon > 0$. This algorithm is also expensive, as it requires computing $G^{-1/2}$. Instead, a more popular version is when we only maintain the diagonal of this matrix. This algorithm is given below

$$\begin{aligned} i_k &\sim \mathrm{Unif}\{1,\dots,m\} \\ g_k &= \nabla f_{i_k}(x_k) \\ v_k &= v_{k-1} + g_k^2 \\ x_{k+1} &= x_k - \alpha \frac{g_k}{\sqrt{\varepsilon + v_k}}, \end{aligned} \tag{8.8}$$

where $v_{-1} = 0$. All operations like g_k^2, $\sqrt{v_k}$, or $\frac{g_k}{\sqrt{\varepsilon + v_k}}$ have to be understood elementwise. AdaGrad was incredibly influential, especially in practical applications. Now, we will see its few important modifications.

[5]We formulate it in the stochastic version, the deterministic variant is straightforward.

RMSProp. The abbreviation stands for Root Mean Square Propagation, which is an unpublished algorithm proposed by Martens and Sutskever

$$\begin{aligned} i_k &\sim \mathrm{Unif}\{1,\dots,m\} \\ g_k &= \nabla f_{i_k}(x_k) \\ v_k &= \beta v_{k-1} + (1-\beta) g_k^2 \\ x_{k+1} &= x_k - \alpha \frac{g_k}{\sqrt{\varepsilon + v_k}}, \end{aligned} \tag{8.9}$$

where $v_{-1} = 0$. In other words, we used a more conservative update for v_k.

Adam. This time, let's incorporate the idea of stochastic heavy ball method into RMSProp.

$$\begin{aligned} i_k &\sim \mathrm{Unif}\{1,\dots,m\} \\ g_k &= \nabla f_{i_k}(x_k) \\ m_k &= \beta_1 m_{k-1} + g_k \\ v_k &= \beta_2 v_{k-1} + g_k^2 \\ x_{k+1} &= x_k - \alpha \frac{m_k}{\sqrt{\varepsilon + v_k}}, \end{aligned} \tag{8.10}$$

where $v_{-1} = 0$. To date, Adam has collected over 160000 citations — an incredible number for a paper in any field of research.

What is even more surprising is that this algorithm does not converge even in the convex case, see Reddi et al..

Optimization Methods for Data Science (2023W) 11 Dec 2023

Lecture 9

Lecturer: Yurii Malitskyi

9 Constrained Optimization

Today, we switch to another class or optimization problems — constrained optimization

$$\min_{x\in C} f(x), \tag{9.1}$$

where $C \subset \mathbb{R}^n$ and f is differentiable over C.

Similarly, to what we had before in the unconstrained case, a point $x \in C$ is called a *global minimum*, if $f(x) \leqslant f(y)$ for all $y \in C$. A point $x \in C$ is called a *local minimum*, if there is a neighborhood $B(x,r)$ of x such that $f(x) \leqslant f(y)$ for all $y \in B(x,r)\cap C$.

When dealing with constrained problems, we say that x is *feasible* if $x \in C$ and *infeasible* otherwise.

Our goal is to develop the first-order optimality condition for this type of problem. In the unconstrained case, it was easy: just look at $\nabla f(x) = 0$. In the constrained case this, evidently, won't work. Just consider the problem $\min_{x\in[1,2]} x^2$. The point $x = 1$ is the global minimum, but there is no obvious quantity that becomes 0.

Considering (9.1) in full generality may be not very helpful. So we assume that our set C is defined as

$$C = \left\{x \in \mathbb{R}^n \colon g_i(x) \leqslant 0, h_j(x) = 0, \forall i \in 1,\dots,m, \forall j \in 1,\dots,r\right\},$$

where all g_i and h_j are also differentiable. In other words, now our problem of interest is

$$\begin{aligned} &\min_x f(x) \\ &\text{s.t. } g_i(x) = 0, \qquad i = 1,\dots,m, \\ &\qquad h_j(x) \leqslant 0, \qquad j = 1,\dots,r, \end{aligned} \tag{9.2}$$

where all functions f, g_i, h_j are differentiable over $\mathbb{R}^n$. As expected, the constraints $g_i(x) = 0$ and $h_j(x) \leqslant 0$ are called equality and inequality constraints, respectively.

We are approaching our goal incrementally, and for some time we will focus only on the equality constraints.

Equality constraints. Our problem of interest is

$$\begin{aligned} &\min_x f(x) \\ &\text{s.t. } g_i(x) = 0, \qquad i = 1,\dots,m. \end{aligned} \tag{9.3}$$

We define a Lagrangian $L\colon \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ as

$$L(x,\lambda) = f(x) + \sum_{i=1}^m \lambda_i g_i(x) = f(x) + \langle \lambda, g(x)\rangle,$$

where the last equation assumes that $g(x) = (g_1(x),\dots,g_m(x))$.

The importance of L can be seen from the following informal statement:

solutions of (9.2) are among critical point of L.

In other words, we can find all critical points of an unconstrained function L and by examining each of them we will find a solution of (9.3).

Example 2. *Consider*

$$\begin{aligned} &\min x^2 + 4y^2 + 16z^2 \\ &s.t.\ xy = 1 \end{aligned}$$

The objective function tends to $+\infty$ if any variable is unbounded (we say that the function is coercive*). Thus, the problem must have a solution. We form Lagrangian:*

$$L(x, y, z, \lambda) = x^2 + 4y^2 + 16z^2 + \lambda(xy - 1).$$

Critical points of L can be found from

$$\begin{aligned} 0 &= \nabla_x L = 2x + \lambda y \\ 0 &= \nabla_y L = 8y + \lambda x \\ 0 &= \nabla_z L = 32z \\ 0 &= \nabla_\lambda L = xy - 1. \end{aligned}$$

This implies that $z = 0$, $2xy + \lambda y^2 = 0$ and $8xy + \lambda x^2 = 0$. And by solving it, we get $x = \pm\sqrt{2}$ and $y = \pm 1/\sqrt{2}$. The objective at two points $(\sqrt{2}, 1/\sqrt{2}, 0)$ and $(-\sqrt{2}, -1/\sqrt{2}, 0)$ has the same value, so these two points are both global minimizers.

Let us denote the optimal value of (9.3) by p_*. It is not difficult to see that

$$p_* = \min_x \max_\lambda L(x, \lambda) = \min_x \max_\lambda f(x) + \sum_{i=1}^m \lambda_i g_i(x) = \min_x \begin{cases} f(x) & \text{if } g_i(x) = 0 \text{ for all } i \\ +\infty & \text{otherwise.} \end{cases} \tag{9.4}$$

Definition 2. A vector $\lambda^* \in \mathbb{R}^m$ is called a *Lagrange multiplier* for (9.3) if

$$p_* = \inf_x L(x, \lambda^*).$$

We need an important and simple fact,

Lemma 9.1. For any function φ, we have that[6]

$$\max_y \min_x \varphi(x, y) \leqslant \min_x \max_y \varphi(x, y).$$

Proof.

$$\forall x, y\ \varphi(x, y) \leqslant \max_y \varphi(x, y) \implies \forall y\ \min_x \varphi(x, y) \leqslant \min_x \max_y \varphi(x, y).$$

Since the latter inequality holds for any y, we can take the maximum over the left-hand side. ■

Hence, going back to (9.4), we have that

$$p_* = \min_x \max_\lambda L(x, \lambda) \geqslant \max_\lambda \min_x L(x, \lambda) =: d_*$$

That is it is always true that $p_* \geqslant d_*$. We call the first problem $\min_x \max_\lambda L(x, \lambda)$ as primal, which is equivalent by (9.4) to our original problem (9.3). And we call the second problem $\max_\lambda \min_x L(x, \lambda)$ as dual. With the notation $F(\lambda) = \min_x L(x, \lambda)$, we can also write the dual problem as

$$\max_\lambda \left\{ F(\lambda) := \min_x L(x, \lambda) \right\} \tag{9.5}$$

Notice that the dual problem is always concave! (Why?)

[6] We assume that both min and max are attained. Otherwise we must write inf and sup.

Theorem 9.1 (Weak duality)**.** *For any feasible solution x and any λ, we have that $F(\lambda) \leqslant f(x)$ and $p_* \geqslant d_*$.*

Under some additional assumptions we might have $p_* = d_*$, which is called *strong duality*. If $p_* > d_*$, then we say that there is a *duality gap* between primal and dual problems. If there is a duality gap, the set of Lagrange multipliers is empty.

If x^* and λ^* are solutions of primal and dual problems respectively, we have that

$$L(x^*, \lambda) \leqslant L(x^*, \lambda^*) \leqslant L(x, \lambda^*),$$

in other words (x^*, λ^*) is a saddle point of the Lagrangian L.

If λ^* exists then it is obvious that our previous informal statement made sense.

Regarding the primal problem it is usually easy to establish the existence of a solution x^*. Unfortunately, for the dual problem, we usually have to impose some extra conditions, which are called *constraint qualifications* condition. The next theorem is one such example.

Theorem 9.2. *If x^* is a (local) solution of (9.2) and $\{\nabla g_i(x^*) : i = 1, \dots, m\}$ are linearly independent, then there exists $\lambda^* \in \mathbb{R}^m$ such that*

$$0 = \nabla_x L(x^*, \lambda^*) = \nabla f(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(x^*) = 0$$

Example 3. *Consider the following problem*

$$\min x \quad \textit{subject to } y^2 - x^3 = 0.$$

It is obvious that the solution of this problem is $(x^*, y^*) = (0, 0)$. But if we form the Lagrangian $L(x, \lambda) = x + \lambda(y^2 - x^3)$ and consider the corresponding system

$$\begin{cases} 1 - 3x^2\lambda & = 0 \\ 2\lambda y & = 0 \\ y^2 - x^3 & = 0, \end{cases}$$

we will see that this system does not have a solution. Hence, we cannot apply Theorem 9.2 here. The reason is that indeed at $(x^*, y^*) = (0, 0)$, we have that $\nabla g_1(x^*, y^*) = 0$. In other words, it is not linearly independent.

Optimization Methods for Data Science (2023W) 8 Jan 2024

Lecture 10

Lecturer: Yurii Malitskyi

10 Constrained Optimization: inequality constraints and general form

Let's recall what we learnt last lecture. Let's illustrate what we obtain with an example.

Example 4. *Consider the problem of finding the least-squares solution of a linear system:*

$$\min_x \frac{1}{2}\|x\|^2 \quad \textit{subject to} \quad Ax = b,$$

where we assume that the system is feasible (that is at least one such x exists). For simplicity, we assume that the rows of A are linear independent, so $m \leqslant n$. If $m = n$, the problem is not interesting, since it has only one feasible point $x = A^{-1}b$. So we assume that $m < n$, that is the system $Ax = b$ is under-determined.

Let us form the Lagrangian:

$$L(x,\lambda) = \frac{1}{2}\|x\|^2 + \langle \lambda, Ax - b\rangle.$$

Then it is easy to find the optimal $x^* = -A^\top\lambda^*$ from $\nabla_x L(x^*,\lambda^*) = 0$. We find λ^* from $Ax = b$, which implies that

$$-AA^\top\lambda^* = b.$$

Since $\operatorname{rank} A = m < n$, the matrix $AA^\top$ is invertible and thus,

$$\lambda^* = -(AA^\top)^{-1}b \quad x^* = A^\top(AA^\top)^{-1}b.$$

Inequality constraints. In the general case when inequalities constraints are also present as in (9.2) we have very similar results. Thus, consider

$$\begin{aligned} &\min_x f(x) \\ &\text{s.t. } g_i(x) = 0, \qquad i = 1,\dots,m \\ &\qquad h_j(x) \leqslant 0, \qquad j = 1,\dots,r, \end{aligned} \tag{10.1}$$

where each function is assumed to be differentiable.

For a feasible x, a constraint $h_j(x) \leqslant 0$ is called *active*, if $h_j(x) = 0$, and *inactive* otherwise. For a given feasible x, we denote the set of all active constraints as $A(x) = \{j : h_j(x) = 0\}$.

It is fairly easy to see that if x^* is a local minimum of (10.1), then x^* is also a local minimum of the same problem where we keep inequality constraints only from $A(x^*)$. In other words, inactive constraints do not matter. In the same vein, we can say that active constraints are like equalities constraints. Thus, if x^* is a local minimum of (10.1), then it is also a local minimum of equality constrained problem:

$$\begin{aligned} &\min_x f(x) \\ &\text{s.t. } g_i(x) = 0, \qquad i = 1, \dots, m \\ &\qquad h_j(x) = 0, \qquad j \in A(x^*) \end{aligned} \tag{10.2}$$

Now, if x^* is regular (meaning that some CQ condition holds), then there exist Lagrange multipliers $\lambda^* \in \mathbb{R}^m$ and μ_j^* for $j \in A(x^*)$ such that

$$\nabla f(x^*) + \sum_i \lambda_i^* \nabla g_i(x^*) + \sum_{j \in A(x^*)} \mu_j^* \nabla h_j(x^*) = 0.$$

It makes sense to set $\mu_j^* = 0$ for all inactive constraints $j \notin A(x^*)$. Hence, we get

$$\begin{aligned} &\nabla f(x^*) + \sum_i \lambda_i^* \nabla g_i(x^*) + \sum_{j=1}^r \mu_j^* \nabla h_j(x^*) = 0 \\ &\mu_j^* = 0 \qquad j \notin A(x^*). \end{aligned}$$

If we introduce the Lagrangian L as before

$$L(x, \mu) = f(x) + \sum_{i=1}^m \lambda_i g_i(x) + \sum_{j=1}^r \mu_j h_j(x),$$

we will see that now μ_j must be nonnegative. This can be seen from the following:

$$\begin{aligned} p_* &= \min_x \max_{\lambda \in \mathbb{R}^m \mu \in \mathbb{R}_+^r} L(x, \lambda, \mu) \\ &= \min_x \max_{\lambda \in \mathbb{R}^m \mu \in \mathbb{R}_+^r} f(x) + \sum_{i=1}^m \lambda_i g_i(x) + \sum_{j=1}^r \lambda_j h_j(x) = \min_x \begin{cases} f(x) & \text{if } g_i(x) = 0 \text{ and } h_j(x) \leqslant 0 \text{ for all } i, j \\ +\infty & \text{otherwise.} \end{cases} \end{aligned} \tag{10.3}$$

The weak duality holds because of the same reasons. The dual problem, however, now is constrained. In particular, let $F(\lambda, \mu) = \min_x L(x, \lambda, \mu)$, then the dual problem is

$$\max_{\lambda \in \mathbb{R}^m, \mu \in \mathbb{R}_+^r} F(\lambda, \mu).$$

As before, it is a concave problem.

Let us formulate the necessary optimality condition more rigorously. They are known as the Karush-Kuhn-Tucker or KKT condition.

Theorem 10.1 (KKT condition)**.** *Let x^* be a local minimum of* (10.1) *and assume that x^* is regular. Then there exist Lagrange multiplier vectors $\lambda^* \in \mathbb{R}^m$ and $\mu^* \in \mathbb{R}^r$ such that*

$$\begin{aligned} &\nabla f(x^*) + \sum_i \lambda_i^* \nabla g_i(x^*) + \sum_{j=1}^r \mu_j^* \nabla h_j(x^*) = 0 \\ &\mu_j^* \geqslant 0, \qquad j = 1, \dots, r \\ &\mu_j^* = 0, \qquad j \notin A(x^*). \end{aligned}$$

The last condition can be succinctly written as

$$\mu_j^* h_j(x^*) = 0, \qquad j = 1, \dots, r$$

and is known as *complementarity slackness condition*. If $h_j(x^*) \leqslant 0$ is slack, that is we have a strict inequality, then the constraint $\mu_j \geqslant 0$ cannot be slack and vise versa.

General constrained optimization. Often, instead of considering (10.1), it may be more useful (and simpler) an abstract formulation

$$\min_{x\in C} f(x), \tag{10.4}$$

where C is some set. For simplicity, we assume C to be nonempty closed and convex and f differentiable.

Theorem 10.2 (Optimality condition).

1. *If x^* is a local solution of (10.4), then*

$$\langle \nabla f(x^*), x - x^*\rangle \geqslant 0 \quad \textit{for all } x \in C.$$

2. *If f is convex, then it is also a sufficient condition.*

Proof. Necessary direction. Since x^* is a local minimizer, for any $x \in C$ there exists α_0 such that

$$f(x^* + \alpha(x - x^*)) - f(x^*) \geqslant 0 \quad \forall \alpha \in [0, \alpha_0]$$

Dividing it by α and tending it to 0, we obtain

$$\langle \nabla f(x^*), x - x^*\rangle \geqslant 0.$$

Sufficient direction. If f is convex, then for any x,

$$f(x) - f(x^*) \geqslant \langle \nabla f(x^*), x - x^*\rangle \geqslant 0.$$

■

Consider a particular instance of (10.4)

$$\min_{x\in C} \frac{1}{2}\|x - z\|^2. \tag{10.5}$$

We have the following:

(i) For any z, (10.5) has a unique solution. It is called a *metric projection of z onto C* and is denoted by $x^* = P_C z$.

(ii) $x^* = P_C z$ if and only if $\langle x^* - z, x - x^*\rangle \geqslant 0$ for all $x \in C$.

We do not provide the proof, as it is a direct application of Theorem 10.2. Now consider some examples.

1. Projection onto the ball $B_r(0)$. We have

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

2. Projection onto the hyperplane $H = \{y \colon \langle a, y\rangle = b\}$.

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

3. Projection onto the subspace $S = \{y \colon Ay = 0\}$, where A is a $m \times n$ matrix with $\operatorname{rank}(A) = m$. That is we have to solve

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

By KKT optimality condition, we have

$$\begin{aligned} x - z + A^\top \lambda &= 0 \\ Ax &= 0. \end{aligned}$$

Solving this system, we get

$$P_C z = x^* = \big(I - A^\top (AA^\top)^{-1} A\big) z.$$

In general, there are no explicit formula for most closed convex sets. But if such formula exists or if the projection can be computed efficiently, it makes sense to exploit this in the algorithmic development. For instance, by analogy to the gradient descent, one may consider *projected gradient method*

$$x_{k+1} = P_C(x_k - \alpha_k \nabla f(x_k)).$$

This method keeps most of the properties of the gradient descent and its analysis follows mostly the same lines. Similar extension exist for accelerated or stochastic versions of GD.