1 Introduction

In this course we are interested in solving optimization problems:

$$\min \quad f(x) \quad \text{subject to} \quad x \in X$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the *objective (or cost) function* and $X \subseteq \mathbb{R}^n$ is the *feasible set.* A *minimization* problem is *convex* if X is a convex set and f is a convex function.[1]

Optimization problems show up in many areas:

Applications of optimization

- Fitting/classification: *Least squares:* Given data points $(x_1, y_1), \ldots, (x_n, y_n)$ where $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$ we want to find $w \in \mathbb{R}^p$ and $b \in \mathbb{R}$ such that $y_i \approx w^T x_i + b$. A common way to find such a w, b is to solve

$$\min_{w \in \mathbb{R}^p, b \in \mathbb{R}} \quad \sum_{i=1}^n (w^T x_i + b - y_i)^2. \tag{1}$$

Having solved this optimization problem and obtained the optimal w, b, the predicted output $\bar{y}$ for a new data point $\bar{x}$ is $\bar{y} = w^T \bar{x} + b$. This is an unconstrained optimization problem. It is convex. In fact, solving (1) has a 'closed-form solution', and amounts to solve a positive definite system of linear equations.
Logistic loss: If $y_i \in \{-1, +1\}$ (classification problem), it is more common to use a logistic loss rather than a least-squares loss. This leads to the optimization problem

$$\min_{w \in \mathbb{R}^p, b \in \mathbb{R}} \quad \sum_{i=1}^n \log_2 \left(1 + e^{-y_i(w^T x_i + b)}\right). \tag{2}$$

This is an unconstrained optimization problem, which is again convex (prove it). However, we do not, in general have a closed form solution, and so we have to resort to iterative methods to approach the solution of (2). Having solved this optimization problem and obtained the optimal w, b, the predicted class $\bar{y}$ for a new data point $\bar{x}$ is $\bar{y} = \operatorname{sign}(w^T \bar{x} + b)$.
Nonlinear classification: Now assume we have a family of functions $F = \{f_w : w \in \mathbb{R}^p\}$ indexed by some real vector $w \in \mathbb{R}^p$. For example f_w could be a neural network with weight vector w. The training problem, with a logistic loss, then becomes

$$\min_{w \in \mathbb{R}^p} \quad \sum_{i=1}^n \log_2 \left(1 + e^{-y_i f_w(x)}\right).$$

This is again an unconstrained problem, but in general it can be nonconvex problem (depending on the parameterization $w \mapsto f_w$).

[1] It is important in the latter definition that we are dealing here with a minimization problem; maximizing a convex function subject to convex constraints is *not* considered a convex problem.

Remark 1. *A motivation for the logistic loss can be explained as follows: we put a model* $P(y_i = +1) = e^{w^T x_i + b}/(1 + e^{w^T x_i + b})$ *and* $P(y_i = -1) = 1/(1 + e^{w^T x_i + b})$*. The likelihood of a set of observations* $\{y_1, \ldots, y_n\}$ *is*

$$\prod_{i:y_i=1} \frac{e^{w^T x_i + b}}{1 + e^{w^T x_i + b}} \cdot \prod_{i:y_i=-1} \frac{1}{1 + e^{w^T x_i + b}}.$$

So the log likelihood is

$$\sum_{i:y_i=1} (w^T x_i + b) - \sum_{i=1}^{n} \log(1 + e^{w^T x_i + b}) \tag{3}$$

Note that (3) *is equal to*

$$-\sum_{i=1}^{n} \log(1 + e^{-y_i(w^T x_i + b)}) \tag{4}$$

since for $y_i = 1$ *we get* $\log(1 + e^{-y_i(w^T x_i + b)}) = \log(1 + e^{-(w^T x_i + b)}) = -(w^T x_i + b) + \log(1 + e^{w^T x_i + b})$.

- Geometry: given a cloud of point $x_1, \ldots, x_n \in \mathbb{R}^p$, we want to find the ellipsoid E of minimum volume that contains the points, i.e., we want to solve

  $$\min \quad \text{volume}(E) \quad \text{s.t.} \quad x_i \in E \quad \forall i = 1, \ldots, n.$$

  Assuming (for simplicity) that the ellipsoid is centered at the origin, we can write $E = \left\{z \in \mathbb{R}^p : z^T Q^{-1} z \leq 1\right\}$ where Q is a $p \times p$ real symmetric matrix that is positive definite. Then the volume of E is proportional to $\det(Q)^{1/2}$. Thus our problem can be written as

  $$\min \quad \det(Q) \quad \text{s.t.} \quad \begin{cases} Q \text{ is positive definite} \\ x_i^T Q^{-1} x_i \leq 1. \end{cases} \tag{5}$$

  This is a constrained optimization problem. As written, this problem is not convex, as the function $Q \mapsto \det(Q)$ is not convex. However (6) can be reformulated as a convex optimization problem, using the following two observations: if we do the change of variables $P = Q^{-1}$, and consider minimizing $\log\det Q$ (which is the same as minimizing $\det Q$, since log is monotonic), then $\log\det Q = -\log\det P$, and the latter is a convex function of P. Our problem is equivalent to

  $$\min \quad -\log\det(P) \quad \text{s.t.} \quad \begin{cases} P \text{ is positive definite} \\ x_i^T P x_i \leq 1. \end{cases} \tag{6}$$

  The objective function is convex in P, and the feasible set is convex (why?), thus this is a convex optimization problem.

- Graph theory: given a graph $G = (V, E)$ where $E \subset \binom{V}{2}$, a *stable set* of G is a subset S of vertices that are pairwise nonadjacent, i.e., $i, j \in S \Rightarrow \{i, j\} \notin E$. The maximum stable set problem asks for the largest stable set in a given graph G

  $$\max \quad |S| \quad \text{s.t.} \quad S \text{ stable set}.$$

  Such a problem can be reformulated as a constrained optimization over $\mathbb{R}^n$ by considering the characteristic vector x of S:

  $$\max_{x \in \mathbb{R}^n} \quad \sum_{i=1}^{n} x_i \quad \text{s.t.} \quad \begin{cases} x_i^2 = x_i & \forall i = 1, \ldots, n \\ x_i x_j = 0 & \forall \{i, j\} \in E. \end{cases}$$

Optimization on the cube To illustrate some of the concepts in this course consider the problem of minimizing a function $f : \mathbb{R}^n \to \mathbb{R}$ on $[0,1]^n$, i.e., to compute:

$$f^* = \min_{x \in [0,1]^n} f(x).$$

Our goal will be to find a solution with accuracy $\epsilon > 0$:

$$\text{Find } \bar{x} \text{ s.t. } f(\bar{x}) - f^* \leq \epsilon. \tag{*}$$

The algorithms have access to f through a *black box* which, given an input $x \in [0,1]^n$ returns the value $f(x) \in \mathbb{R}$. This is called an zeroth-order oracle model[2] The complexity of an algorithm on a given function f is the number of queries it makes to the oracle. So a general algorithm has the following form:

1. Query oracle at $x_0 \in [0,1]^n$ to get value $f_0 = f(x_0)$
2. Query oracle at $x_1 \in [0,1]^n$ (allowed to depend on f_0) to get value $f_1 = f(x_1)$
3. Query oracle at $x_2 \in [0,1]^n$ (allowed to depend on f_0, f_1) to get value $f_2 = f(x_2)$
4. $\ldots$
5. Query oracle at $x_{N-1} \in [0,1]^n$ (allowed to depend on $f_0, \ldots, f_{N-2}$) to get value $f_{N-1} = f(x_{N-1})$
6. Output $\bar{x}$ based on the gathered information about f

We will consider the class of functions that are L-Lipschitz with respect to ℓ_∞ norm

$$\mathcal{F}_L = \{f : [0,1]^n \to \mathbb{R} \text{ s.t. } |f(x) - f(y)| \leq L\|x - y\|_\infty \ \forall x, y \in [0,1]^n\}$$

where $\|x\|_\infty = \max_{i=1,\ldots,n} |x_i|$. We can prove the following:

Proposition 1.1. *There is an algorithm that can return an ϵ-accurate minimizer (in the sense of (*)) of any $f \in \mathcal{F}_L$ with a number of queries $\leq (\lfloor \frac{L}{2\epsilon} \rfloor + 2)^n$.*

Proof. Grid search. We discretize the cube $[0,1]^n$ using grid points that are equispaced by $2\epsilon/L$ in each dimension. Let $(x_i)_{i=1,\ldots,N}$ be the grid points; there are $N \leq (\lfloor \frac{L}{2\epsilon} \rfloor + 2)^n$ such grid points (we include points at coordinate 0 and coordinate 1, hence the $+2$). Let $\bar{x}$ be the grid point where the value of f is smallest, i.e.,

$$\bar{x} = \operatorname*{argmin}_{x \in \{x_1, \ldots, x_N\}} f(x).$$

We claim that this algorithm achieves the desired accuracy. Indeed, let x^* be a minimizer of f on $[0,1]^n$, and let $\tilde{x}$ be the closest grid point to x^* in the ℓ_∞ norm. Since the grid is equispaced by $2\epsilon/L$ it is not difficult to see that $\|x^* - \tilde{x}\|_\infty \leq \epsilon/L$. Then we have

$$f(\bar{x}) - f^* \leq f(\tilde{x}) - f^* \leq L\|\tilde{x} - x^*\|_\infty \leq \epsilon$$

as desired. □

[2] A first-order oracle returns the gradient of f at x, and a second-order oracle returns the Hessian of f at x. We will see this later...

The algorithm produced in the previous proposition is not great. For functions of large number of variables n the algorithm is not at all practical. Can we do better? The answer turns out to be no, if we want our algorithm to work for all $f \in \mathcal{F}_L$.

Proposition 1.2. *Assume $\mathcal{A}$ is an algorithm that returns an ϵ-accurate minimizer for all $f \in \mathcal{F}_L$. Then there is at least one function $f \in \mathcal{F}_L$ on which $\mathcal{A}$ does at least $\geq (\lfloor \frac{L}{3\epsilon} \rfloor)^n - 1$ queries.*

Proof. Recall that an algorithm $\mathcal{A}$ is given by a sequence of query points $x_0, x_1, \dots$ where each query point is allowed to depend on the answer received on the previous ones. We are going to simulate the algorithm on the function $f(x) \equiv 0$ (the function equal to zero everywhere). On such a function the algorithm will query certain (fixed) points $x_0, x_1, x_2, \dots, x_{N-1}$ all in $[0,1]^n$ before producing a point $\bar{x} \in [0,1]^n$. Let $S = \{x_0, \dots, x_{N-1}, \bar{x}\}$. We claim that necessarily $|S| \geq (\lfloor L/(3\epsilon) \rfloor)^n$. Fix $\eta = 3\epsilon/L$ and consider dividing $[0,1]^n$ into small boxes each of size η. We have at least $\lfloor 1/\eta \rfloor^n$ disjoint such boxes. Assuming for contradiction that $|S| < (\lfloor 1/\eta \rfloor)^n$, by the pigeonhole principle, there exists at least one box which does not contain any point from S. Let x^* be the center of that box and define the function

$$f(x) = \min(0, L\|x - x^*\|_\infty - \eta L/2).$$

Note that $f \in \mathcal{F}_L$, it is zero outside the box centered at x^* and its minimum is $-\eta L/2 = -3\epsilon/2$. If we run the algorithm on this function f we will get the same output as for the function that is identically zero (the $\bar{x} \in S$ from above). But this $\bar{x}$ is outside the box centered at x^* and so $f(\bar{x}) = 0$. This contradicts the assumption that the algorithm achieves ϵ accuracy on all functions in $\mathcal{F}_L$ because $f(\bar{x}) - f^* = 3\epsilon/2 > \epsilon$. Thus it must be that $|S| \geq \lfloor 1/\eta \rfloor^n = (\lfloor \frac{L}{3\epsilon} \rfloor)^n$. $\square$

We have thus shown that the following min-max quantity

$$\min_{\substack{\text{Algorithms } \mathcal{A} \text{ that achieve} \\ (*) \text{ for all functions in } \mathcal{F}_L}} \ \max_{f \in \mathcal{F}_L} \ \text{Complexity of } \mathcal{A} \text{ on } f$$

lies between $(\frac{L}{3\epsilon})^n$ and $(\frac{L}{2\epsilon} + 2)^n$.