Continuous Optimization

Chapter 1: Mathematical Preliminaries

1 Optimization problems

In this course we will focus on finite-dimensional continuous optimization problems

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0 \quad i = 1, \ldots, m \\ & h_j(x) = 0 \quad j = 1, \ldots, p \end{aligned} \tag{1}$$

- $x \in \mathbb{R}^n$: vector of variables to be chosen (n scalar variables $x_1, \ldots, x_n$)
- f: objective function to be minimized
- $g_1, \ldots, g_m$: inequality constraint functions to be satisfied
- $h_1, \ldots, h_p$: equality constraint functions to be satisfied

The feasible set can also be denoted by $S = \{x \in \mathbb{R}^n : g_i(x) \leq 0 \; i = 1, \ldots, m, \;\; h_j(x) = 0 \; j = 1, \ldots, p\}$.

1.1 Classification

Concerning the dimensionality, optimization problems can be

- infinite-dimensional: when the space of variables is infinite dimensional, e.g., elements can be vectors of functions of one ore more variables;
- finite-dimensional: when the space of variables is finite-dimensional, e.g., $\mathbb{R}^n$.

Concerning the granularity of the space, optimization problems can be

- discrete: when at least one variable is an integer;
- continuous: when all variables involved are continuous.

Concerning the hypothesis of linearity, optimization problems are also called

- linear programming: when the objective function is linear and the feasible set is defined by a system of linear equalities and inequalities;
- nonlinear problems: when the objective function is nonlinear and/or at least one constraint is defined by a nonlinear function.

Concerning the hypothesis of differentiability, we can distinguish between

- smooth optimization: when the functions defining the objective and the constraints are continuously differentiable;
- nonsmooth optimization problems: when the functions defining the objective and the constraints are continuous, but typically not differentiable in all points of the feasible set. Weaker notions of derivatives are used to solve this problem.
- derivative-free optimization: this branch of optimization consider cases in which evaluating the objective function/constraints is very costly, in particular the functions defining the objective and the constraints might not be continuous. A similar field is that of black-box optimization, in which the optimizer assumes to not have any information on the objective function or on the constraints.

Concerning the hypothesis of convexity, we can distinguish between

- convex optimization, when the functions defining the objective and the constraints are all convex;

- nonconvex optimization, when the functions defining the objective and the constraints might be nonconvex.

Example 1 (The transportation problem (Linear Programming)) *Consider a company producing and selling a certain commodity. The company has a set of N sources, where the commodity is produced and a set of M demand centers where the commodity is sold. At each source $i \in \{1, \dots, N\}$, a given quantity q_i of commodity is available. Each demand center $j \in \{1, \dots, M\}$ requires a given quantity d_j. We indicate by c_{ij} the cost per unit of transporting the commodity from source i to demand center j.*

The problem is to determine the quantity to be transported from each source to each demand center in such a way that

- *the availability constraints at the sources are satisfied;*
- *the requirements of the demand centers are satisfied;*
- *the total transportation cost is minimized.*

To formulate the problem, we indicate by x_{ij} the quantity of the commodity transported from source i to demand center j. Then the optimization problem is

$$\begin{aligned} \min \quad & \sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij} x_{ij} \\ s.t. \quad & \sum_{j=1}^{M} x_{ij} \leq q_i \quad i = 1, \dots, m \\ & h_j(x) = 0 \quad j = 1, \dots, p \end{aligned}$$

Example 2 (Training of Neural Networks (Unconstrained Optimization)) *Let us start from describing a single (artificial) neuron. In particular, this unit takes as an input a vector and performs a scalar product with the weights of the neuron. In a second step, this value passes through a thresholding function σ called activation function. Inspired by real neurons, also this unit propagates its output only if the weighted sum of its inputs (in the natural neuron, the potential difference) is over a certain threshold. Neural networks stack various neurons in parallel to build a layer and, in turn, concatenate various layer (deep networks). This model is very expressive, in fact already a wide-enough 2-layer neural network is able to approximate any possible function of the input.*

Neural networks are the most widely used machine learning model and they can be formally defined as follows.

- *$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{M}$ be a training set of M instances*
  - *$x_i \in \mathbb{R}^d$ d is the amount of features of the dataset (e.g., pixels)*
  - *$y_i \in \mathbb{R}$*
- *the j-th layer applied on a generic input n_{j-1}-dimensional input x:*
  $g_j(x) := \sigma(W_j x + b_j)$, $W_j \in \mathbb{R}^{n_{j-1} \times n_j}$, $b_j \in \mathbb{R}^{n_j}$, σ is applied component-wise,
  e.g., ReLU $\sigma(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{otherwise} \end{cases}$
- *the weights: $w = \begin{pmatrix} vec(W_1) \\ \vdots \\ vec(W_L) \end{pmatrix} \in \mathbb{R}^n$*
- *the network applied on the i-th instance: $h_i(w) := g_L \circ \cdots \circ g_1(x_i)$*
- *the losses: $f_i(w) := \mathcal{L}(h_i(w), y_i)$, e.g., $\mathcal{L}(h_i(w), y_i) = (h_i(w) - y_i)^2$*

$$\min_{w \in \mathbb{R}^n} f(w) = \frac{1}{M} \sum_{i=1}^{M} f_i(w) \tag{2}$$