Selected Methods for Modern Optimization in Data Analysis

Department of Statistics and Operations Research
UNC-Chapel Hill
Fall 2018

Instructor: Quoc Tran-Dinh Scriber: Quoc Tran-Dinh

Lecture 1: Introduction to Optimization Models

Index Terms: Optimization problem; mathematical formulation; convexity vs. nonconvexity; problem classification; convex models; nonconvex models.

1 Introduction

In this lecture, we first state the mathematical formulation of an optimization problem, both in unconstrained and constrained settings. Next, we recall some fundamental concepts used in mathematical optimization problems. After that we classify optimization problems into different classes based on structures of the problem components, and briefly discuss the aim of solution methods. Finally, we provide several optimization models in practice including convex and nonconvex models.

1.1 Mathematical formulation of optimization problems

Formulation: Let $f_i : \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ be multivalued functions and can take $+\infty$ for $i = 0, \cdots, m$, and let $\mathcal{X}$ be a nonempty set in $\mathbb{R}^p$. We consider the following formulation:

$$\left\{\begin{array}{ll} \min\limits_{\mathbf{x}\in\mathbb{R}^p} & f_0(\mathbf{x}) \\ \text{subject to} & f_i(\mathbf{x}) \le 0,\ i \in \mathcal{I}, \\ & f_i(\mathbf{x}) = 0,\ i \in \mathcal{E}, \\ & \mathbf{x} \in \mathcal{X}, \end{array}\right. \tag{1}$$

Here

- $\mathbf{x} \in \mathbb{R}^p$ is called a vector of decision variables (also referred to as a vector of optimization variables);
- f_0 is called the objective function;
- f_i, $(i = 1, \cdots, m)$, are called constrained functions; and
- $\mathcal{X}$ is called the domain of problem (1).

The inequality and equality index sets $\mathcal{I}$ and $\mathcal{E}$ are disjoint and subsets of $\mathcal{I}_m := \{1, \cdots, m\}$, and $\mathcal{I} \cup \mathcal{E} = \mathcal{I}_m$.

Quick examples: Here are some quick examples:

- **The Weber problem:** Given n separated points $\mathbf{a}^{(i)}$ in $\mathbb{R}^p$, we want to find a point $\mathbf{x}^\star$ in $\mathbb{R}^p$ such that it minimizes the total distance from this point to all the points $\mathbf{a}^{(i)}$ for $i = 1, \cdots, n$. Finding such a point $\mathbf{x}^\star$ can be formulated into the following problem:

$$\min_{\mathbf{x}\in\mathbb{R}^p} \sum_{i=1}^{n} \|\mathbf{x} - \mathbf{a}^{(i)}\|_2, \tag{2}$$

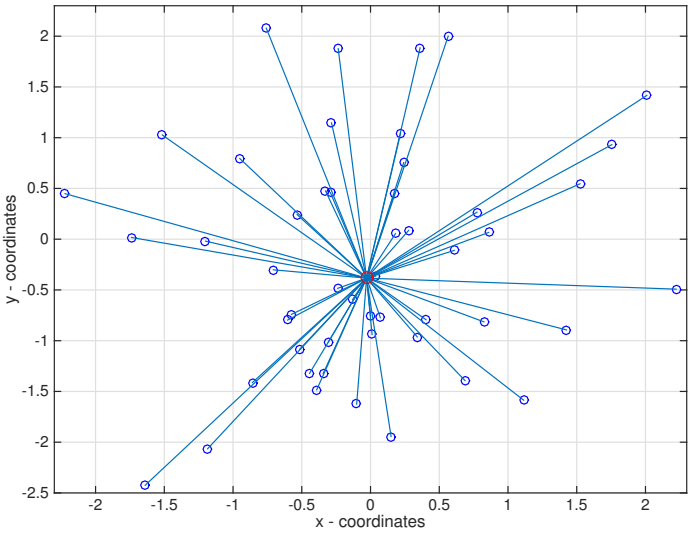


Figure 1: An illustration of the Weber solution with 50 random points

where $\|\cdot\|_2$ is the Euclidean norm in $\mathbb{R}^p$, i.e., $\|\mathbf{x}\|_2 := \sqrt{\sum_{i=1}^p \mathbf{x}_i^2}$. Figure 1 illustrates the solution of the Weber problem with 50 random points in a two-dimensional space:

Exercise. Transform this problem into a second order cone program studied below and solve it by using an available interior-point solver. Using CVX (http://cvxr.com) to solve this problem and visualize it in $\mathbb{R}^2$.

- **The projection onto a hyperplane:** Given a hyperplane $\mathcal{H} := \left\{\mathbf{x} \in \mathbb{R}^p \mid \mathbf{a}^T\mathbf{x} = b\right\}$ in $\mathbb{R}^p$ and a point $\mathbf{x}^0 \in \mathbb{R}^p$ ($\mathbf{x}^0$ is not on $\mathcal{H}$). The aim is to find the projection of $\mathbf{x}^0$ onto $\mathcal{H}$. This problem can be formulated into an optimization problem as

$$\min_{\mathbf{x}\in\mathcal{H}} \left\|\mathbf{x} - \mathbf{x}^0\right\|_2. \tag{3}$$

This problem can be analytically solved, and its solution can be denoted by $\mathrm{proj}_{\mathcal{H}}(\mathbf{x}^0)$.

Exercise. Compute explicitly the solution of this problem. Check the solution and visualize the result in $\mathbb{R}^2$.

- **Maximum eigenvectors:** Given a symmetric matrix $S \in \mathbb{R}^{p\times p}$, the goal is to find a vector $\mathbf{x} \in \mathbb{R}^p$ that solves the following problem:

$$\max_{\mathbf{x}\in\mathbb{R}^p} \left\{\mathbf{x}^\top S\mathbf{x} \mid \|\mathbf{x}\|_2 = 1\right\}. \tag{4}$$

Any vector $\mathbf{x}$ as an optimal solution of this problem is called the maximum eigenvector of $\mathbf{S}$, and its optimal value is called the maximum eigenvalue of S. This is a classical problem in numerical linear algebra. Besides direct methods such as eigen-decomposition, algorithms for approximating solution of this problem often rely on *the power method*. Similarly, we can define the minimum eigenvector problem by replacing the operator max by min. This problem is more difficult to solve than (4).

Feasible set and solution set: Let us define

$$\mathcal{F} := \{\mathbf{x} \in \mathcal{X} \mid f_i(\mathbf{x}) \le 0,\ i \in \mathcal{I},\ f_i(\mathbf{x}) = 0,\ i \in \mathcal{E}\}. \tag{5}$$

Then, $\mathcal{F}$ is a subset in $\mathbb{R}^p$, which is called the feasible set of (1). Any point $\mathbf{x} \in \mathcal{F}$ is called a feasible solution of (1). The min notation means that we want to find a feasible solution $\mathbf{x}^\star \in \mathcal{F}$ such that $f_0(\mathbf{x}^\star)$ is minimized in one of the following senses:

- **Local optimal solutions:** A point $\mathbf{x}^\star \in \mathcal{F}$ is called a local optimum (or local optimal solution) of (1) if there exists a neighborhood $\mathcal{N}(\mathbf{x}^\star)$ of $\mathbf{x}^\star$ such that

$$f_0(\mathbf{x}^\star) \le f_0(\mathbf{x}), \quad \forall\, \mathbf{x} \in \mathcal{F} \cap \mathcal{N}(\mathbf{x}^\star).$$

- **Global optimal solutions:** $\mathbf{x}^\star \in \mathcal{F}$ is called a global optimum (or global optimal solution) of (1) if

$$f_0(\mathbf{x}^\star) \leq f_0(\mathbf{x}), \quad \forall\, \mathbf{x} \in \mathcal{F}.$$

The subscript $\mathbf{x}$ in min indicates that $\mathbf{x}$ is a vector of variables that we want to optimize. The value $f^\star := f_0(\mathbf{x}^\star)$ is called the optimal value of (1). We denote by $\mathcal{X}^\star$ the global optimal solution set of (1), i.e.:

$$\mathcal{X}^\star := \{\mathbf{x}^\star \in \mathcal{F} \mid f_0(\mathbf{x}^\star) = f^\star\}.$$

Remark. Since we can write $\max\limits_{\mathbf{x}} f_0(\mathbf{x}) = -\min\limits_{\mathbf{x}} \{-f_0(\mathbf{x})\}$, we can convert a maximization problem to a minimization problem of the form (1). Hence, we assume that we always work with the minimization problem (1), and call it an optimization problem or a mathematical program.

Remark. In nonconvex optimization (see definition below), finding a local optimal solution is also not an easy task. We often find a stationary point of (1), which will be defined precisely in **Lecture 4**.

Remark. In this lecture, we only work with continuous domain $\mathcal{X}$ in $\mathbb{R}^p$, and focus on continuous optimization problem. When $\mathcal{X}$ is a discrete set, then (1) becomes a discrete optimization problem, which is not covered in this course. Students who are interested in this topic can find it in [3, 12, 17, 19, 21, 22].

1.2 Classification of optimization problems

We often **solve** an optimization problem by using one of the following approaches:

- Design an algorithm to solve this problem directly.
- Transform it into a canonical form (manually or by modeling software) then use available solvers.

Since each method or solver only aims at solving some classes of problems, we need to classify our problem into a certain class in order to choose a suitable solver or algorithm. There are several ways of classifying optimization problems. We will look at some of them.

1.2.1 Unconstrained vs. constrained optimization

An unconstrained optimization problem is defined by

$$\min_{\mathbf{x}\in\mathbb{R}^p} f_0(\mathbf{x}). \tag{6}$$

In this case, the variable $\mathbf{x}$ can take any value in the whole space $\mathbb{R}^p$, while minimizing the objective function f_0. Compare this setting and (1), we have $m = 0$ and $\mathcal{X} = \mathbb{R}^p$. Hence, (6) is a special case of (1).

We note that, we can consider the case $\mathcal{X}$ is not the whole space $\mathbb{R}^p$, but rather, $\mathcal{X}$ can be an open set in $\mathbb{R}^p$. In this case, we still refer to (6) as an unconstrained problem. When $m > 0$ and/or $\mathcal{X}$ is a closed and nonempty subset of $\mathbb{R}^p$, then problem (1) is called a constrained optimization problem.

Principally, solving a constrained optimization problem is harder than solving an unconstrained one. Hence, many optimization methods use algorithms for solving unconstrained problems as sub-routines to solve constrained problems. However, there is no borderline between unconstrained and constrained settings in terms of mathematical representations.

- *Any constrained problem can be reformulated as an unconstrained problem:* Indeed, we consider the following constrained problem

$$\min_{\mathbf{x}\in\mathcal{F}} f_0(\mathbf{x}),$$

where $\mathcal{F}$ is the feasible set. We define the following function (also called the indicator function of $\mathcal{F}$):

$$\delta_{\mathcal{F}}(\mathbf{x}) := \begin{cases} 0 & \text{if } \mathbf{x} \in \mathcal{F}, \\ +\infty & \text{otherwise.} \end{cases}$$

Then, the above constrained problem can be reformulated as an unconstrained problem of the form:

$$\min_{\mathbf{x}\in\mathbb{R}^p}\left\{f(\mathbf{x}) := f_0(\mathbf{x}) + \delta_{\mathcal{F}}(\mathbf{x})\right\}.$$

- *Any unconstrained problem can also be reformulated as a constrained one:* We consider the unconstrained problem (6). If we introduce a slack variable $t = f_0(\mathbf{x})$ then we can write (6) as

$$\min_{\mathbf{y}:=(\mathbf{x},t)} t \quad \text{s.t.} \quad f_0(\mathbf{x}) - t = 0.$$

1.2.2 Linear vs. nonlinear optimization

A function f is called an affine function if it can be written as $f(\mathbf{x}) = \langle \mathbf{a}, \mathbf{x}\rangle + b$, where $\mathbf{a}$ is a given vector in $\mathbb{R}^p$, $b \in \mathbb{R}$ is given, and $\langle\cdot,\cdot\rangle$ is the inner product in $\mathbb{R}^p$, i.e., $\langle \mathbf{x}, \mathbf{y}\rangle = \sum_{i=1}^p \mathbf{x}_i\mathbf{y}_i$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^p$. A subset $\mathcal{X}$ in $\mathbb{R}^p$ is called a polyhedron if it can be represented as $\mathcal{X} := \{\mathbf{x} \in \mathbb{R}^p \mid \mathbf{A}\mathbf{x} \le \mathbf{b}\}$, where $\mathbf{A} \in \mathbb{R}^{m\times p}$ is a given matrix, and $\mathbf{b} \in \mathbb{R}^m$ is a given vector.

If all the functions f_i are affine for $i = 0, \cdots, m$ and $\mathcal{X}$ is polyhedral in $\mathbb{R}^p$, then problem (1) reduces to a problem called linear program (LP). Otherwise, it is called a nonlinear program (NLP). More precisely, a linear program can be written as

$$\begin{cases} \min\limits_{\mathbf{x}\in\mathbb{R}^p} & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & \mathbf{C}\mathbf{x} \le \mathbf{d}. \end{cases}$$

Here $\mathbf{A}$ and $\mathbf{C}$ are matrices, and $\mathbf{c}, \mathbf{b}$ and $\mathbf{d}$ are vectors with appropriate dimension. The first group of constraints is called equality constraints, while the second one is called inequality constraints.

Theory and methods for linear programming is well-developed, and well understood. There are several software packages including open-source or commercial to solve linear programs. Two well-known methods for LPs are the simplex method invented by G. B. Dantzig in 1947 and the interior-point method (I.I. Dikin (1967), N. Karmarkar (1984)). These methods have been used in several LP solvers such as CPLEX, Gurobi, Mosek, XPress, etc.

1.2.3 Convex vs. nonconvex optimization

We first informally define what convex sets and convex functions are. Let $\mathcal{X}$ be a subset of $\mathbb{R}^p$, we say that $\mathcal{X}$ is convex if for any $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ and for any $\alpha \in [0, 1]$, the point $(1-\alpha)\mathbf{x} + \alpha\mathbf{y} \in \mathcal{X}$. In other words, $\mathcal{X}$ is convex if for any $\mathbf{x}, \mathbf{y} \in \mathcal{X}$, then the whole segment $[\mathbf{x}, \mathbf{y}] \subseteq \mathcal{X}$. Given a convex set $\mathcal{X} \subseteq \mathbb{R}^p$, we consider a multivariable function $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ which can also take $+\infty$ as its value. We say that f is convex if for any $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ and for any $\alpha \in [0, 1]$ we have $f((1-\alpha)\mathbf{x} + \alpha\mathbf{y}) \le (1-\alpha)f(\mathbf{x}) + \alpha f(\mathbf{y})$. Geometrically, any tangent line of f lies below the graph of f.

With these definitions of convex sets and convex functions, we can define the class of convex optimization problems in (1) by assuming that f_i are convex for $i = 0, \cdots, m$, and $\mathcal{X}$ is a convex set in $\mathbb{R}^p$. If this condition is not satisfied, then we have a nonconvex optimization problem. Clearly, any linear program is convex. In general, not any convex optimization problem can be solved efficiently, but there are various subclasses of convex optimization problems, which can be solved efficiently. We will go over these subclasses in detail in the sequel.

Here are some useful facts of convex optimization:

- Any local optimal solution is also a global one.
- Several subclasses of convex optimization problems such as linear, convex quadratic, and conic programming can be solved in polynomial-time[1] by interior-point methods. We say that these classes of problems are standard convex optimization or tractable convex optimization.

[1] In terms of worst-case complexity from the complexity theory, but not in practice.

- Various subclasses of nonstandard convex optimization problems can be converted into the standard one[2], and can be solved by interior-point methods. But as we will see later, this is not always a good idea, especially for large-scale applications.

1.2.4 Smooth vs. nonsmooth optimization

If $\mathcal{X} = \mathbb{R}^p$ and all the functions f_i $(i = 1, \cdots, m)$, are smooth (i.e., their derivatives exist and are continuous in some sense), then problem (1) is called smooth optimization problem. Otherwise, we have a nonsmooth optimization problem.

Smooth problems are much easier to handle by means of optimization methods that use derivatives, while nonsmooth problems require different type of information when we design optimization schemes. However, as we will see later, sole smoothness is not enough to design good algorithms. We need additional structures/assumptions on the functions f_i. Here are some observations:

- A true model is complicated. We often approximate it by a simpler one for our solution purpose. Nonsmooth functions are often "better" to model applications in terms of accurately capturing the true model.
- But, nonsmoothness also requires new solution techniques to efficiently handle it.
- In practice, we frequently design "tractable" nonsmooth functions to trade-off between modeling stage of the practical problem and numerical solution methods.

In this lecture, we also focus on nonsmooth problems but mainly for the convex case. For the nonconvex case, we only focus on certain cases. We refer the reader to the monograph [14] for nonsmooth optimization theory.

1.3 What do we mean by "numerical solution methods"?

Problem (1) or its subclasses are still too complicated to solve analytically. Classical analysis such as Fermat's rule or Lagrange multiplier methods cannot solve problems of the dimension greater than 3 (at least in the general cases). Hence, we often solve optimization problems of the form (1) numerically. Clearly, working with numerical methods, we encounter with errors, and do not get exact solutions for (1). Therefore, we accept approximate solutions of (1) up to a certain accuracy. For instance,

- we can accept a point $\tilde{\mathbf{x}}^\star \in \mathcal{X}$ as an approximate solution of (1) such that $|f_0(\tilde{\mathbf{x}}^\star) - f^\star| \le \varepsilon_0$ for the optimal value $f^\star$, and $f_i(\tilde{\mathbf{x}}^\star) \le \varepsilon_i$ for the constraint $f_i(\mathbf{x}) \le \varepsilon_i$, $(i = 1, \cdots, m)$, where ε_i $(i = 0, \cdots, m)$ are given desired accuracies. This accuracies can be the same or different.
- or we can measure the distance from the approximate solution $\tilde{\mathbf{x}}^\star$ to the true solution set $\mathcal{X}^\star$ as

$$\operatorname{dist}(\tilde{\mathbf{x}}^*, \mathcal{X}^\star) \le \varepsilon,$$

  where $\operatorname{dist}(\mathbf{x}, \mathcal{X}) := \min_{\mathbf{y}\in\mathcal{X}} \|\mathbf{y} - \mathbf{x}\|_2$ is the distance from $\mathbf{x}$ to $\mathcal{X}$.

These two criteria are often used in convex optimization. In nonconvex settings, they are two restrictive and are often very hard to achieve. Hence, we may only aim at finding an approximate stationary point or an approximate local minimum. We will discuss the nonconvex case in our last lecture.

Throughout this course, we aim at designing numerical methods that can produce an approximate solution to some subclasses of (1) up to a desired accuracy. We also emphasize that most of optimization schemes are iterative methods, which require to query information from component functions and constraint sets that form (1). Some methods have finite iterations and sometimes produce exact[3] solutions (e.g., simplex methods in linear programming), while others are infinite and we just terminate after a finite number of iterations to obtain an approximate solution (e.g., interior-point, or ellipsoid methods).

[2]This will be defined formally later.
[3]Up to machine precision.

2 Convex optimization problems

Our first primary goal of this course is to study convex optimization. Hence, we start our discussion by reviewing the convexity before defining problems.

2.1 Convex sets and convex functions

We repeat the definition of convexity formally here.

- Let $\mathcal{X}$ be a subset of $\mathbb{R}^p$. We say that $\mathcal{X}$ is convex if for any $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ and for any $\alpha \in [0,1]$,we have $(1-\alpha)\mathbf{x} + \alpha\mathbf{y} \in \mathcal{X}$. In other words, if $\mathbf{x}, \mathbf{y} \in \mathcal{X}$, then the whole segment $[\mathbf{x}, \mathbf{y}] \subseteq \mathcal{X}$.

Examples of convex sets include hyperplanes, half-planes, polyhedra, balls, and simplexes. Given a convex set $\mathcal{C}$, it is said to be a convex cone if for any $\mathbf{x}, \mathbf{y} \in \mathcal{C}$, and $\alpha, \beta \geq 0$, we have $\alpha\mathbf{x} + \beta\mathbf{y} \in \mathcal{C}$. For instance, $\mathbb{R}^p_+$ is a convex cone, and $\{(\mathbf{x}, t) \in \mathbb{R}^p \times \mathbb{R}_+ \mid \|\mathbf{x}\|_2 \leq t\}$ is also a convex cone.

- Given a convex set $\mathcal{X} \subseteq \mathbb{R}^p$, we consider a multivariable function $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ which is allowed it to take $+\infty$ as its value. We say that f is convex if for any $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ and for any $\alpha \in [0,1]$ we have

$$f((1-\alpha)\mathbf{x} + \alpha\mathbf{y}) \leq (1-\alpha)f(\mathbf{x}) + \alpha f(\mathbf{y}).$$

- If f is a convex function, then the sublevel set $\mathcal{L}_\gamma(f) := \{\mathbf{x} \in \mathcal{X} \mid f(\mathbf{x}) \leq \gamma\}$ for any γ is a convex set.
- We denote by $\mathrm{dom}\,(f) := \{\mathbf{x} \in \mathcal{X} \mid f(\mathbf{x}) < +\infty\}$ the domain of f.
- The set $\mathrm{epi}(f) := \{(\mathbf{x}, t) \in \mathcal{X} \times \mathbb{R} \mid f(\mathbf{x}) \leq t\}$ is called the epi-graph of f.
- We say that f is proper if for all $\mathbf{x} \in \mathbb{R}^p$ we have $f(\mathbf{x}) > -\infty$, and $\mathrm{dom}\,(f) \neq \emptyset$.
- We say that f is closed if its epi-graph is closed.

Examples of convex functions include affine functions ($\langle \mathbf{a}, \cdot \rangle + b$), norms ($\|\cdot\|$), and negative logarithmic functions ($-\ln(\cdot)$). From now on, we consider functions defined on $\mathcal{X} = \mathbb{R}^p$, without recalling it again. We denote by $\Gamma_0(\mathbb{R}^p)$ the class of all proper, closed, and convex functions from $\mathbb{R}^p$ to $\mathbb{R} \cup \{+\infty\}$. We also recall the following special convex functions, which will be used in the sequel.

- **Indicator function**: Let $\mathcal{X} \subseteq \mathbb{R}^p$ be a nonempty, closed, and convex set. We define

$$\delta_{\mathcal{X}}(\mathbf{x}) := \begin{cases} 0 & \text{if } \mathbf{x} \in \mathcal{X} \\ +\infty & \text{otherwise} \end{cases} \tag{7}$$

  as the indicator function of the set $\mathcal{X}$. It is convex and nonsmooth. In this case, $\mathrm{dom}\,(\delta_{\mathcal{X}}) = \mathcal{X}$.

- **Support function:** Associated with the set $\mathcal{X}$ above, we define

$$s_{\mathcal{X}}(\mathbf{x}) := \sup_{\mathbf{y} \in \mathcal{X}} \langle \mathbf{x}, \mathbf{y} \rangle \tag{8}$$

  as the support function of $\mathcal{X}$. This function is also convex and generally nonsmooth.

- **Fenchel's conjugate:** Given $f \in \Gamma_0(\mathbb{R}^p)$, we define

$$f^*(\mathbf{x}) := \sup_{\mathbf{y} \in \mathrm{dom}(f)} \{\langle \mathbf{x}, \mathbf{y} \rangle - f(\mathbf{y})\} \tag{9}$$

  as the Fenchel conjugate function of f. It is trivial to show that $s_{\mathcal{X}}(\cdot)$ is the Fenchel's conjugate of $\delta_{\mathcal{X}}(\cdot)$ and vice versa.

Remark. Alternative to convexity, we have concavity. A function f is said to be concave, if $-f$ is convex. Hence, instead of considering the problem of minimizing a convex function on a convex set, we can consider the problem of maximizing a concave function over a convex set. The latter is also called a convex optimization. However, for simplicity of our presentation, we only focus on the minimization of convex function over a convex feasible set. For concave functions, we can convert into convex ones and process them as we do in this lecture.

2.2 Convex optimization problem: from unconstrained to constrained settings

Let $f \in \Gamma_0(\mathbb{R}^p)$. We consider the following optimization problem:

$$f^\star := \min_{\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x}) \tag{10}$$

in the sense that we find a vector $\mathbf{x}^\star \in \mathrm{dom}\,(f)$ such that $f^\star = f(\mathbf{x}^\star) \le f(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^p$. Since $\mathbf{x}$ does not have any restriction, we call (10) a unconstrained problem. If $f^\star$ is finite and $\mathcal{X}^\star := \{\mathbf{x}^\star \in \mathrm{dom}\,(f) \mid f(\mathbf{x}^\star) = f^\star\}$ is nonempty, then we say that (10) admits solution. In this case, $\mathcal{X}^\star$ is called the solution set and $f^\star$ is called the optimal value.

For the convex problem (10), any optimal solution is global, and the solution set $\mathcal{X}^\star$ is also closed and convex. This is different from nonconvex problems where solutions may not be global, meaning that there exists a neighborhood $\mathcal{N}(\mathbf{x}^\star)$ of the solution $\mathbf{x}^\star$ such that $f(\mathbf{x}^\star) \le f(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{N}(\mathbf{x}^\star)$, but this does not hold for $\mathbf{x} \notin \mathcal{N}(\mathbf{x}^\star)$.

2.2.1 Composite convex optimization

A special class of (10) is the composite convex optimization problem, which recently attracts a great attention due to its applications in different fields such as machine learning, statistics, and engineering. This problem can be written as

$$\min_{\mathbf{x}\in\mathbb{R}^p} \left\{f(\mathbf{x}) := g(\mathbf{x}) + h(\mathbf{x})\right\}, \tag{11}$$

where $g, h \in \Gamma_0(\mathbb{R}^p)$. In data science, image processing, and machine learning, g is often called a loss/data fidelity/empirical risk term, while h is referred to as a regularizer or a penalty term. A well-known example of this is the following penalized LASSO problem:

$$\min_{\mathbf{x}\in\mathbb{R}^p} \left\{\tfrac{1}{2}\left\|\mathbf{A}\mathbf{x} - \mathbf{b}\right\|_2^2 + \lambda\|\mathbf{x}\|_1\right\},$$

using in statistical learning and compressive sensing, where $\lambda > 0$ is a penalty parameter. We are more interested in the class of h with a "simple" structure as we will discuss in the sequel.

2.2.2 Simple constrained convex optimization

We separate this class of problems since we will design special optimization schemes to solve this class of problems. We consider the following convex optimization problem:

$$\min_{\mathbf{x}\in\mathbb{R}^p} \left\{g(\mathbf{x}) \mid \mathbf{x} \in \mathcal{C}\right\}, \tag{12}$$

where $g \in \Gamma_0(\mathbb{R}^p)$ is convex, and $\mathcal{C}$ is a nonempty, closed, and convex set in $\mathbb{R}^p$. We focus on the case that $\mathcal{C}$ is simple if we can solve one of the following problems efficiently (e.g., in a closed form, or with a low-order polynomial time algorithm):

$$\min_{\mathbf{x}\in\mathcal{C}}\langle\mathbf{a}, \mathbf{x}\rangle \qquad \text{or} \qquad \min_{\mathbf{x}\in\mathcal{X}}\|\mathbf{x} - \mathbf{a}\|^2,$$

where $\mathbf{a} \in \mathbb{R}^p$ is a given vector. The first problem is called a linear minimization oracle, while the second one is just the projection of $\mathbf{a}$ onto $\mathcal{C}$. For instance, when $\mathcal{X}$ is a box, ball, cone, or a simplex, solving these problems can be done in a closed form or by a simple polynomial time algorithm. By choosing $h(\cdot) := \delta_{\mathcal{X}}(\cdot)$, we can convert (12) into the composite convex minimization form (11) and vice versa.

2.2.3 Convex-concave saddle-point problem

By using its Fenchel conjugate g^* of a convex function g, we can write any convex function as $g(\mathbf{x}) = \sup_{\mathbf{y}}\{\langle\mathbf{x}, \mathbf{y}\rangle - g^*(\mathbf{y})\}$. Hence, for a general convex function, we can assume that it has the form

$$g(\mathbf{x}) = \max_{\mathbf{y}\in\mathbb{R}^n}\left\{\langle\mathbf{A}\mathbf{x}, \mathbf{y}\rangle - \varphi(\mathbf{y})\right\}, \tag{13}$$

where φ from $\mathbb{R}^n \to \mathbb{R}\cup\{+\infty\}$ is a given proper, closed, and convex function, and $\mathbf{A} \in \mathbb{R}^{n\times p}$. The function g defined by (13) is called a max-type convex function. For instance, we can write $\|\mathbf{x}\|_1 = \max_{\mathbf{y}}\{\langle \mathbf{x}, \mathbf{y}\rangle \mid \|\mathbf{y}\|_\infty \le 1\} = \max_{\mathbf{y}}\left\{\langle \mathbf{x}, \mathbf{y}\rangle - \delta_{\{\mathbf{y}\mid\|\mathbf{y}\|_\infty\le 1\}}(\mathbf{y})\right\}$, which is in the form (13) with $\mathbf{A} = \mathbb{I}$ and $\varphi(\mathbf{y}) := \delta_{\{\mathbf{y}\mid\|\mathbf{y}\|_\infty\le 1\}}(\mathbf{y})$.

Now, let say, we consider the composite minimization problem (11) with g defined by (13). Then, this problem can be written into the following min-max saddle point form:

$$\min_{\mathbf{x}}\{h(\mathbf{x}) + g(\mathbf{x})\} \qquad \Leftrightarrow \qquad \min_{\mathbf{x}\in\mathbb{R}^p}\max_{\mathbf{y}\in\mathbb{R}^n}\{F(\mathbf{x},\mathbf{y}) := h(\mathbf{x}) + \langle \mathbf{A}\mathbf{x}, \mathbf{y}\rangle - \varphi(\mathbf{y})\}. \tag{14}$$

Clearly, the objective function $F(\cdot,\cdot)$ is convex w.r.t. the first argument $\mathbf{x}$, while it is concave w.r.t. the second argument $\mathbf{y}$. The convex-concave saddle point problem is an interesting topic in convex optimization. We will focus on this problem in the next lecture.

2.2.4 Constrained convex optimization

The general form of a constrained convex optimization problem is defined as

$$\begin{cases} \min\limits_{\mathbf{x}\in\mathbb{R}^p} & f_0(\mathbf{x}) \\ \text{s.t.} & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & f_i(\mathbf{x}) \le 0, i = 1,\cdots,m_I, \\ & \mathbf{x}\in\mathcal{X}, \end{cases} \tag{15}$$

where $\mathbf{A} \in \mathbb{R}^{m_E\times p}$ is a given matrix; $\mathbf{b}\in\mathbb{R}^{m_E}$; $f_i \in \Gamma_0(\mathbb{R}^p)$ for $i = 0,\cdots,m_I$; and $\mathcal{X}$ is nonempty, closed, and convex set in $\mathbb{R}^p$. Clearly, we can skip one of these constraints to obtain a simpler representation of a convex problem. However, for the time being, we just leave it as it is. We also emphasize that, by means of indicator functions, one can convert a constrained convex problem into an unconstrained form (10). In order to convert the unconstrained convex problem (10) into a constrained one (15), we can introduce slack variables. For instance, we can write (10) into

$$\begin{cases} \min\limits_{\mathbf{x}\in\mathbb{R}^p, t\in\mathbb{R}} & t \\ \text{s.t.} & f(\mathbf{x}) - t \le 0. \end{cases}$$

Hence, there is no clear border between constrained and unconstrained formulations in terms of mathematical representation. This reformulation can be used to study some mathematical properties of the problem. However, when we design numerical methods for solving it, this reformulation may not always be useful.

2.2.5 Conic programming

One of the most important class of convex optimization is conic programming, which includes linear programming, convex quadratic programming, second-order cone programming, and semidefinite programming. The general form of this problem can be written into the following canonical form:

$$\begin{cases} \min\limits_{\mathbf{x}} & \langle \mathbf{c}, \mathbf{x}\rangle \\ \text{s.t.} & \mathcal{A}(\mathbf{x}) = \mathbf{b}, \\ & \mathbf{x}\in\mathcal{K}, \end{cases} \tag{16}$$

Here, $\mathbf{x}$ is an optimization variable, which can be a vector in $\mathbb{R}^p$ or a symmetric matrix in $\mathcal{S}^p$ (the set of all symmetric matrix of the size $p\times p$); $\mathbf{c}$ is a given vector or matrix of the same size as $\mathbf{x}$; $\mathcal{A}$ is a linear operator; $\mathbf{b}$ is a given vector in $\mathbb{R}^m$, and $\mathcal{K}$ is a convex cone. We emphasize three kinds of convex cone:

- The orthant cone: $\mathcal{K} = \mathbb{R}^p_+ := \{\mathbf{x}\in\mathbb{R}^p \mid \mathbf{x}_i \ge 0,\ i = 1,\cdots,p\}$. Then, (16) becomes a linear program (LP).
- The second-order cone: $\mathcal{K} = \mathcal{L}^p_+ := \left\{\mathbf{x}\in\mathbb{R}^p \mid \sqrt{\sum_{i=1}^{p-1}\mathbf{x}_i^2} \le \mathbf{x}_p\right\}$. This cone is also called a Lorentz cone or an ice-cream cone. Then, (16) becomes a second-order cone program (SOCP).

- The semidefinite cone: $\mathcal{K} = \mathcal{S}^p_+ := \big\{\mathbf{x} \in \mathbb{R}^{p\times p} \mid \mathbf{x} = \mathbf{x}^\top, \mathbf{x} \succeq 0\big\}$. Here $\mathbf{x} \succeq 0$ means that $\mathbf{x}$ is positive semidefinite. Then, (16) becomes a semidefinite program (SDP).

In (16) we do not only consider a single cone constraint $\mathbf{x} \in \mathcal{K}$, but also we consider $\mathbf{x} \in \mathcal{K}_1 \otimes \cdots \otimes \mathcal{K}_N$ the Cartesian product of N cones $\mathcal{K}_i$, and $\mathcal{K}_i$ is either $\mathbb{R}^p_+$, $\mathcal{L}^p_+$ or $\mathcal{S}^p_+$ for all $i = 1, \cdots, N$. When they are mixed, then we obtain a mixed conic program. Roughly speaking, a conic program is an optimization problem of optimizing a linear function over the intersection of an affine space and a nonempty, closed, and convex cone.

We note that, we have the following inclusions

$$\text{LP} \subset \text{SOCP} \subset \text{SDP}.$$

Hence, any LP can be seen as a special SOCP, and any SOCP can be considered as a special SDP.
To see that, we consider the fact that $x_i \geq 0$ can be formulated as $\mathbf{x}_i \in \mathcal{L}^1_+$. Hence, LP is a special case of SOCP. Alternatively, we can write a second-order cone constraint $(\mathbf{x}, t) \in \mathcal{L}^{p+1}_+$ as $t - \frac{\mathbf{x}^\top\mathbf{x}}{t} \geq 0$, which, by Schur's complement formula, is equivalent to

$$\begin{bmatrix} t & \mathbf{x}^\top \\ \mathbf{x} & t\mathbb{I} \end{bmatrix} \succeq 0.$$

Hence, any SOCP can be considered as a special SDP.
As an example, we consider a convex quadratic program:

$$\min_{\mathbf{x}\in\mathbb{R}^p} \left\{ \tfrac{1}{2}\langle \mathbf{Q}\mathbf{x}, \mathbf{x}\rangle + \mathbf{q}^\top\mathbf{x} \;\mid\; \mathbf{A}\mathbf{x} = \mathbf{b},\; \mathbf{x} \geq 0 \right\}, \tag{17}$$

where $\mathbf{Q}$ is a symmetric and positive semidefinite matrix in $\mathbb{R}^{p\times p}$. This convex quadratic program can be converted into an SOCP problem. Hence, (17) can also be considered as a special instance of SOCP. Indeed, let $LL^\top = \mathbf{Q}$ is the Cholesky decomposition of $\mathbf{Q}$. Then, the objective $f(\mathbf{x}) = (1/2)\langle \mathbf{Q}\mathbf{x}, \mathbf{x}\rangle + \mathbf{q}^\top\mathbf{x}$ can be written as $f = t + \mathbf{q}^\top\mathbf{x}$ with additional constraints $\mathbf{y} = L^\top\mathbf{x}$ and $\|\mathbf{y}\|_2^2 \leq 2t$. While the objective is linear in $\mathbf{x}$ and t, we have one linear constraint $L^\top\mathbf{x} - \mathbf{y} = 0$. The constraint $\|\mathbf{y}\|_2^2 \leq 2t$ can be written as $\|\mathbf{x}\|_2^2 + t^2 + 1 \leq (t+1)^2$ or $\sqrt{\sum_{i=1}^p \mathbf{x}_i^2 + t^2 + 1^2} \leq t + 1$, which is a second-order cone. Hence, we can convert a convex QP into an SOCP with additional variables t and $\mathbf{y}$; and some linear constraints. If $\mathbf{Q}$ has zero eigenvalues, then we can use incomplete Cholesky decomposition to factorize it.

2.3 Disciplined convex programming

Several convex optimization problems can be converted into a conic program (LP, SOCP, or SDP) using convex analysis rules. For examples, here are some common rules:

- The intersection of convex sets is convex.
- The direct sum of two convex sets is convex.
- The point-wise supremum of many convex functions is convex.
- The nonnegative linear combination $\alpha f + \beta g$ of two convex functions f and g is convex, where $\alpha, \beta \geq 0$.
- If $f(\mathbf{x}) = \sup_{\mathbf{y}\in\mathcal{Y}} \varphi(\mathbf{x}, \mathbf{y})$ is convex if $\varphi(\cdot, \mathbf{y})$ is convex for any $\mathbf{y} \in \mathcal{Y}$.

By using these rules, we can transform several convex programs into conic programs by introducing slack variables and additional constraints. This technique is called *disciplined convex progamming*, which was used by M. Grant and S. Boyd (2006) [11], and is implemented in CVX and YALMIP among other modeling software packages.

Let us consider a couple of examples as follows:

1. We consider the following basis pursuit denoising problem (see below):

$$\min_{\mathbf{x}\in\mathbb{R}^p} \left\{ \tfrac{1}{2}\left\|\mathbf{y} - \mathbf{A}\mathbf{x}\right\|_2^2 + \lambda \left\|\mathbf{x}\right\|_1 \right\},$$

This problem can be reformulated as a convex quadratic program. Then, as we have seen, it can be converted into an SOCP. Indeed, by introducing slack variables, we can write this problem equivalently to

$$\begin{cases} \min\limits_{\mathbf{x}\in\mathbb{R}^p,\mathbf{t}\in\mathbb{R}^p} & \frac{1}{2}\mathbf{x}^\top(\mathbf{A}^\top\mathbf{A})\mathbf{x} - (\mathbf{A}^\top\mathbf{y})^\top\mathbf{x} + \lambda\mathbf{1}^\top\mathbf{t} \\ \text{s.t.} & -\mathbf{t}_i \le \mathbf{x}_i \le \mathbf{t}_i,\ i = 1,\cdots,p. \end{cases}$$

Then, we can easily convert this into a standard convex QP with the joint variable $\mathbf{z} := (\mathbf{x}^\top, \mathbf{t}^\top)^\top \in \mathbb{R}^{2p}$.

2. We consider the problem of minimizing the maximum eigenvalue of a symmetric matrix $\mathbf{X} \in \mathcal{S}^p$:

$$\lambda^\star_{\max} = \min_{\mathbf{X}\in\mathcal{S}^p_+} \lambda_{\max}(\mathbf{X}).$$

Since $\lambda_{\max}(\mathbf{X}) = \sup_{\|\mathbf{u}\|_2\le 1} \mathbf{u}^\top\mathbf{X}\mathbf{u}$, which is convex, we use the fact that $\lambda_{\max}\mathbb{I} \succeq \mathbf{X}$ to write it as

$$\lambda^\star_{\max} = \min_{\lambda\ge 0} \left\{\lambda \mid \lambda\mathbb{I} - \mathbf{X} \succeq 0\right\}.$$

This problem is an SDP but not in standard form. However, we can easily convert it into a standard SDP. We have seen several modeling software packages such as GAMP, AMPL, CVX and YALMIP that allow us to convert a nonstandard convex optimization problem into a standard one, which can be solved by available solvers. CVX or YALMIP by itself is not a solver. They simply convert a given convex problem into a standard conic program and pass it to available solvers such as SeDuMi, SDPT3, SDPA or CPLEX to solve the resulting problem and convert the solution back to the solution of the original problem.

Curse of dimensionality: Several existing optimization solvers rely on interior-point (IP) methods. Unfortunately, IP solvers only work well on small and medium size problems, and become inefficient in large-scale settings. The main reason is that it has high per-iteration complexity due to the operations on matrices (matrix-matrix multiplication and matrix decomposition). In addition, they often require to directly input matrices instead of the corresponding linear operators. For instance, if this operator is a fast Fourier transform (FFT) of the form $y = Fx$, then forming the matrix F directly will lead to a huge memory requirement in high dimensional space. But, if we just compute directly the matrix-vector operator $y = Fx$ using an FFT algorithm, then it only requires at most $p\log(p)$ operations, which is much faster than performing matrix-vector multiplications, where p is the dimension of x.

We notice that IP solvers are often designed to solve the standard conic problems of the form:

$$\min_{\mathbf{x}}\langle\mathbf{c},\mathbf{x}\rangle \quad \text{s.t.} \quad \mathcal{A}(\mathbf{x}) = \mathbf{b}, \mathbf{x}\in\mathcal{K}, \tag{18}$$

where $\mathcal{A}$ is a linear operator, and $\mathcal{K}$ is the Cartesian product of standard cones (orthant, second-order, or SDP cones). Given a convex problem, we need to convert it into this standard form (18) using *disciplined convex programming* techniques mentioned above. We often introduce slack variables and add auxiliary constraints. In many cases, the number of variables and constraints are increasing significantly. Hence, the resulting standard conic problem (18) becomes large, and is inefficient to solve by IP solvers. We consider the following two examples:

Example 1. Consider the following convex problem

$$\min_{\mathbf{x}\in\mathbb{R}^p} \|\mathbf{x}\|_1 \quad \text{s.t.} \quad \mathbf{B}\mathbf{x} = \mathbf{b}.$$

In this case, we have to introduce p slack variables t_i for $i = 1,\cdots,p$, and rewrite this problem equivalently to

$$\min_{t,\mathbf{x}\in\mathbb{R}^p} \sum_{i=1}^p t_i \quad \text{s.t.} \quad \mathbf{B}\mathbf{x} = \mathbf{b}, \quad |\mathbf{x}_i| \le t_i, \quad 1 \le i \le p.$$

This problem is in the standard form (18) with $\mathbf{c} = [\mathbf{0}^\top; \mathbf{1}^\top]^\top$, $\mathbf{A} := [\mathbf{B}, \mathbf{0}]$, and $\mathcal{K} = \prod_{i=1}^p \mathcal{L}_2$, where $\mathcal{L}_2$ is a Lorentz (or second-order) cone of dimension 2 (i.e., $\mathcal{L}_2 := \left\{\mathbf{u}\in\mathbb{R}^2 \mid |\mathbf{u}_1| \le \mathbf{u}_2\right\}$). The dimension of problem goes from p to $2p$, and the number of constraints are from m to $m + p$.

Example 2. We consider a semidefinite programming of the form

$$\min_{\mathbf{X}\in\mathcal{S}^p_+}\Big\{\mathrm{trace}\left(\mathbf{CX}\right)+\lambda\left\|\mathrm{vec}\left(\mathbf{X}\right)\right\|_1 \quad \text{s.t.} \;\; \mathrm{trace}\left(\mathbf{X}\right)=1, \;\; \mathbf{X}\geq -c\mathbf{E}\Big\},$$

where $\mathbf{C}\in\mathcal{S}^p$ is a given matrix, $\lambda>0$ and $c>0$ are given constants, and $\mathbf{E}$ is a matrix of all ones.

To convert this problem into the standard SDP form (18), we need to introduce two additional slack matrix variables $\mathbf{U}$ and $\mathbf{V}$ in $\mathcal{S}^p$. Then we can write this problem into

$$\min_{\mathbf{X},\mathbf{U},\mathbf{V}\in\mathcal{S}^p}\left\{\mathrm{trace}\left(\mathbf{CX}\right)+\lambda\sum_{i,j=1}^{p}\mathbf{U}_{ij} \;\;|\;\; \mathrm{trace}\left(\mathbf{X}\right)=1, \mathbf{X}-\mathbf{V}=-c\mathbf{E}, \;[\mathbf{X},\mathbf{U},\mathbf{V}]\in\mathcal{K}\right\},$$

where $\mathcal{K}:=\mathcal{S}^p_+\otimes\prod_{i,j=1}^{p}\mathcal{L}_2\otimes\mathbb{R}^{p^2}_+$ with $\mathcal{L}_2:=\left\{\mathbf{u}\in\mathbb{R}^2\mid|\mathbf{u}_1|\leq\mathbf{u}_2\right\}$. This problem has $n:=3p(p+1)/2$ variables, and $m:=p(p+1)/2+1$ linear constraints.

For example, with $p=100$, we have 15150 variables in the new problem (100 SDP variables, 5050 SOC variables, and 5050 linear nonnegative variables), and 5051 linear constraints.

3 Convex optimization in applications

Convex optimization has various applications in different fields, especially in signal and image processing, statistics, machine learning, computer-vision, control, operations research, economics and finance. In this section, we select some prominent applications in recent years to demonstrate the power of convex optimization.

We mention here some problems where we obtain a convex optimization formulation directly from modeling stage. Certainly, the list of applications is getting longer and longer, which is impossible to have an exhaustive category. We just pick some representative examples which can easily be found in the literature.

3.1 Linear programming in applications

We have mentioned linear programming (LP) previously. This problem can be written as:

$$\min_{\mathbf{x}\in\mathbb{R}^p}\Big\{\mathbf{c}^\top\mathbf{x}\mid\mathbf{Ax}=\mathbf{b},\;\mathbf{x}\geq 0\Big\}, \tag{19}$$

where $\mathbf{c}\in\mathbb{R}^p$, $\mathbf{A}\in\mathbb{R}^{m\times p}$ and $\mathbf{b}\in\mathbb{R}^n$ are given, which are called the data of this linear program. This formulation is often referred to as the *primal canonical form* of LPs. Direct applications of linear programming include, but not limited to: production planning, transportation, diet problems, blending problems, portfolio, scheduling, and telecommunication, etc. See [1, 7, 18, 29] for more compelling examples. This topic is rather classical and has been insensitively covered in our STOR 614. Hence, we do not focus on it in this course.

3.2 Linear regression and least squares problem

We are given a set of data points $\mathcal{D}:=\left\{(\mathbf{x}^{(i)},\mathbf{y}_i)\mid i=1,\ldots,m\right\}$, where $\mathbf{x}^{(i)}$ is a feature vector in $\mathbb{R}^p$ and $\mathbf{y}_i$ is an output/observation drawing from a linear regression model:

$$\mathbf{y}=\beta_0+\sum_{j=1}^{p}\beta_j\mathbf{x}_j+\varepsilon,$$

where ε is an $\mathcal{N}(0,1)$ standard Gaussian noise. That is the distribution density of ε_i is $p(\mathbf{z}_i)=\frac{1}{\sqrt{2\pi}}e^{-\mathbf{z}_i^2/2}$. Using the maximum likelihood principle, we can obtain the following problem

$$\min_{\beta_0,\cdots,\beta_p\in\mathbb{R}}\sum_{i=1}^{m}\left(\mathbf{y}_i-\beta_0-\sum_{j=1}^{p}\beta_j\mathbf{x}_j^{(i)}\right)^2.$$

Let us define $\beta = (\beta_0, \beta_1, \cdots, \beta_p)^\top$, $\mathbf{y} := (\mathbf{y}_1, \cdots, \mathbf{y}_m)^\top$, and $\mathbf{X} := \begin{bmatrix} 1 & \mathbf{x}_1^{(1)} & \cdots & \mathbf{x}_1^{(m)} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & \mathbf{x}_p^{(1)} & \cdots & \mathbf{x}_p^{(m)} \end{bmatrix}$, then you can write the above problem as

$$\min_{\beta \in \mathbb{R}^{p+1}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}^\top \beta\|_2^2. \tag{20}$$

This problem is called a linear least squares problem. It is a special case of unconstrained convex quadratic programming. This problem can be solved by simple linear algebra routines such as QR or SVD factorizations.

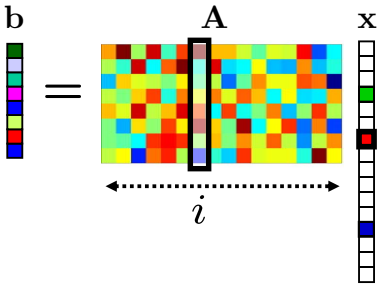
3.3 Basis pursuit and LASSO

The basis pursuit problem can be described by the following formulation:

$$\min_{\mathbf{x} \in \mathbb{R}^p} \Big\{ \|\mathbf{x}\|_1 \mid \mathbf{A}\mathbf{x} = \mathbf{b} \Big\}, \tag{BP}$$

Here, $\mathbf{A} \in \mathbb{R}^{m \times p}$ and $\mathbf{b} \in \mathbb{R}^m$ are given. We often assume that $p \gg m$.

This problem has several applications in different fields. For instance, in signal recovery, we can assume that the unknown signal vector $\mathbf{x}$ lives in a high-dimensional space of p, which is only observed after transforming it into the output vector $\mathbf{b}$, where $\mathbf{b}$ has much smaller dimension m via a linear transformation $\mathbf{A}$. In order to recover $\mathbf{x}$ from the observation $\mathbf{b}$, we need to solve a linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$. However, since $p \gg m$, this system is underdetermined and has infinite solutions. Hence, it is unclear to pick the right solution that we want. Since we expect the signal $\mathbf{x}$ to be sparse in a given domain in order to save storage capacity (we only store the nonzero elements), we design an objective to capture this requirement. The sparsity is represented by minimizing $\|\mathbf{x}\|_0 = \text{card}(\{\mathbf{x}_i \neq 0\})$ (the number of nonzero elements of $\mathbf{x}$), which is very hard to solve. We instead use a convex relaxation of $\|\mathbf{x}\|_0$ as $\|\mathbf{x}\|_1$, and obtain the new problem (BP) above. This formulation is a central problem in compressive sensing and sparse signal recovery.



It is obvious that (BP) is convex but nonsmooth. Solving this problem is often challenging in high dimensional settings. In addition, this problem does not reflect the noise model when we do not have $\mathbf{b} = \mathbf{A}\mathbf{x}$ but rather $\mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{e}$, for some small noise vector $\mathbf{e}$. Therefore, there are several derivations from (BP) as follows:
Penalized LASSO or basis pursuit denoising (BPDN): This problem can be considered as a quadratic penalty formulation of (BP):

$$\min_{\mathbf{x} \in \mathbb{R}^p} \left\{ \frac{1}{2}\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_1 \right\}, \tag{21}$$

where $\lambda > 0$ is a penalty parameter. This problem turns out to be a composite convex minimization (11), but it is nonsmooth due to the ℓ_1-norm term.

Another variant of (BP) is the following basis pursuit denoising formulation that can capture noise:

$$\min_{\mathbf{x} \in \mathbb{R}^p} \Big\{ \|\mathbf{x}\|_1 \mid \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 \leq \sigma \Big\}, \tag{BPDN}$$

where $\sigma > 0$ is a noise level. This problem is a constrained convex problem, and nonsmooth as well.

LASSO: Alternative to these formulations, we can also minimize the noise while keeping the sparsity level at a given value $\gamma > 0$. We can write it as

$$\min_{\mathbf{x}\in\mathbb{R}^n}\left\{\frac{1}{2}\left\|\mathbf{A}\mathbf{x}-\mathbf{b}\right\|_2^2 \mid \left\|\mathbf{x}\right\|_1 \le \gamma\right\}. \tag{LASSO}$$

This problem is indeed an extension of the least squares formulation (20), which is called LASSO (least absolute shrinkage and selection operator). It has been widely used in statistics for variable/feature selection [9, 26]. Other extension of these models such as (21) consists of group LASSO, elastic-net models, and fused LASSO models, which we do not cover them here.

3.4 Compressive sensing, image recovery/reconstruction

Suppose that we want to recover an unknown signal or image $\mathbf{x}$ in a high-dimensional space $\mathbb{R}^p$ from a few observation/measurements $\mathbf{b}$ in a possibly low-dimensional space $\mathbb{R}^m$ ($p \gg m$) that measures through a linear measurement operator $\mathbf{A}$ as $\mathbf{b} = \mathbf{A}\mathbf{x} + \mathbf{e}$ for some noise vector $\mathbf{e}$. While we aim at cleaning the effect of noise by minimizing the data fidelity term $\|\mathbf{b}-\mathbf{A}\mathbf{x}\|_2^2$, we also want to preserve desired properties of the recovered signal or image. In this case, we need to add a regularizer $R(\mathbf{x})$ to promote desired structures of our recovered signal $\mathbf{x}$. We can formulate this problem into the following composite convex minimization of the form (11):

$$\min_{\mathbf{x}\in\mathbb{R}^p}\left\{\frac{1}{2}\left\|\mathbf{A}\mathbf{x}-\mathbf{b}\right\|_2^2 + \lambda R(\mathbf{x})\right\}, \tag{22}$$

where $\lambda > 0$ is a regularization parameter. Clearly, when $R(\mathbf{x}) = \|\mathbf{x}\|_1$, this problem turns to be (21) (the penalized LASSO formulation). However, in image processing, we obtain choose $R(\mathbf{x})$ is a total variation norm (TV-norm). This norm can be defined as

$$\|\mathbf{x}\|_{\mathrm{TV}} := \sum_{i=1}^{p_1}\sum_{j=1}^{p_2}|\mathbf{x}_{i,j+1}-\mathbf{x}_{i,j}| + |\mathbf{x}_{i+1,j}-\mathbf{x}_{i,j}| \quad\text{or}\quad \|\mathbf{x}\|_{\mathrm{TV}} := \sum_{i=1}^{p_1}\sum_{j=1}^{p_2}\sqrt{(\mathbf{x}_{i,j+1}-\mathbf{x}_{i,j})^2 + (\mathbf{x}_{i+1,j}-\mathbf{x}_{i,j})^2},$$

for a given image $\mathbf{x} \in \mathbb{R}^{p_1\times p_2}$. The first one is called "anisotropic", while the second is "isotropic". Using this norm, and with an appropriate linear operator $\mathbf{A}$, we can recast many problems in image processing to this form, such as image denoising, image deblurring, and image inpainting.

In general, we can replace the least squares objective by some general objective such as

$$g(\mathbf{x}) := \sum_{i=1}^{m}\ell_i(\mathbf{a}_i^\top\mathbf{x} - \mu_i; \mathbf{y}_i),$$

where ℓ_i is a nonlinear loss function, such as the absolute $\|\cdot\|_1$, logistic, or hinge loss function.

3.5 Poisson image reconstruction

Consider the low-light imaging problem in signal processing [13], where the imaging data is collected by counting photons hitting a detector over the time. In this setting, we wish to accurately reconstruct an image in low-light, which leads to noisy measurements due to a low photon count level. We can express our observation model using the Poisson distribution as:

$$\mathbb{P}(\mathbf{y}|\mathcal{A}(\mathbf{x})) = \prod_{i=1}^{m}\frac{(\mathbf{a}_i^\top\mathbf{x})^{y_i}}{y_i!}e^{-\mathbf{a}_i^\top\mathbf{x}},$$

where $\mathbf{x}$ is the true image, $\mathcal{A}$ is a linear operator that projects the scene onto the set of observations, $\mathbf{a}_i$ is the i-th row of $\mathcal{A}$, and $\mathbf{y} \in \mathbb{Z}_+^m$ is a vector of observed photon counts.

Via a log-likelihood formulation, we stumble upon a composite minimization problem:

$$\mathbf{x}^* := \arg\min_{\mathbf{x}\in\mathbb{R}^n} \Big\{ \underbrace{\sum_{i=1}^m \mathbf{a}_i^\top\mathbf{x} - \sum_{i=1}^m y_i \log(\mathbf{a}_i^\top\mathbf{x})}_{=:f(\mathbf{x})} + g(\mathbf{x}) \Big\}, \tag{23}$$

where $f(\mathbf{x})$ is self-concordant (but not standard) (see [20] for concrete definitions). In the above formulation, the typical image priors $g(\mathbf{x})$ include the ℓ_1-norm for sparsity in a known basis, total variation semi-norm of the image, and the positivity of the image pixels. While the formulation (23) seems specific to imaging, it is also common in sparse regression with unknown noise variance [24], heteroschedastic LASSO [6], barrier approximations of, e.g., the Dantzig selector [4] and quantum tomography [2] as well.

One state-of-the-art solver for (23) is called SPIRAL-TAP [13], which biases the logarithmic term (i.e., $\log(\mathbf{a}_i^\top\mathbf{x} + \varepsilon) \to \log(\mathbf{a}_i^\top\mathbf{x})$, where $\varepsilon \ll 1$) and then applies non-monotone composite gradient descent algorithms with a Barzilai-Borwein step-size as well as other line-search strategies. Unfortunately, this approach does not have a theoretical convergence guarantee for this problem in general setting.

3.6 Low-rank matrix optimization

Approximating a high dimensional unknown quantity by a low-dimensional approximation in some sense is a fundamental idea to solve many high-dimensional problems in model reduction and multivariate data analysis nowadays. If we consider the underlying unknown as a matrix, then the rank of this matrix represents some sort of dimensionality of the underlying information space. Therefore, it is important to approximate this matrix by a low-rank matrix, which can be convenient to store, transfer, and operate. In the opposite case, some unknown matrix may live in a high-dimensional space, but is is in fact a low-rank matrix. In this case, we also want to reconstruct a low-rank representation of this matrix.

As an example, if we stack frames of a video into a matrix where each column of this matrix is a vectorization of a video frame, then it is likely that the obtained matrix is low-rank or at least can be approximated by a low-rank matrix. The reason is that objects in this video often change slowly or even unchange, which leads to similar frames. Hence, we can stack them into a matrix, we obtain similar columns which represent the low-rankness of this matrix. To this end, a low-rank matrix optimization problem is often reformulated as follows:

$$\min_{\mathbf{X}\in\mathbb{R}^{p_1\times p_2}} \left\{ \frac{1}{2}\left\|\mathbf{b} - \mathcal{A}(\mathbf{X})\right\|_2^2 + \lambda \mathrm{rank}(\mathbf{X}) \right\}, \tag{24}$$

where $\mathcal{A}$ is a linear measurement operator from $\mathbb{R}^{p_1\times p_2}$ to $\mathbb{R}^m$, $\mathbf{b}\in\mathbb{R}^m$ is an observed vector, $\lambda > 0$ is a regularization parameter, and $\mathrm{rank}\,(\mathbf{X})$ is the rank of $\mathbf{X}$.

Unfortunately, this problem is nonconvex and highly nonsmooth. We relax to obtain a convex relaxation:

$$\min_{\mathbf{X}\in\mathbb{R}^{p_1\times p_2}} \left\{ \frac{1}{2}\left\|\mathbf{b} - \mathcal{A}(\mathbf{X})\right\|_2^2 + \lambda \left\|\mathbf{X}\right\|_* \right\}, \tag{25}$$

where $\|\mathbf{X}\|_*$ is the nuclear norm of matrix $\mathbf{X}$ (i.e., $\|\mathbf{X}\|_* := \sum_{i=1}^{\min\{p_1,p_2\}} \sigma_i(\mathbf{X})$, the sum of all singular values $\sigma_i(\mathbf{X})$ of $\mathbf{X}$). Here, $\|\cdot\|_*$ is a convex envelope of $\mathrm{rank}\,(\cdot)$ and it is convex (see the next lecture). This problem is again a composite convex minimization (11). Instead of penalizing the rank term, we can also put it into constraint as a rank constraint of the form $\mathrm{rank}(\mathbf{X}) \le \mu$, for a given value μ.

Another application of (25) is called matrix completion. In this application, we are given a collection of observed entries $\mathbf{M}_\Omega$ of a $(p_2 \times p_2)$-matrix $\mathbf{M}$ obtained from some activities such as movie rating, or website ranking, where $\mathbf{M}_\Omega := \{\mathbf{M}_{ij} \mid (i,j)\in\Omega\}$ with $\Omega \subset \{1,\cdots,p_1\}\times\{1,\cdots,p_2\}$ being the subset of indices. Our goal is to recover $\mathbf{M}$ (approximately) using this observation set. Clearly, since $|\Omega| \ll p_1 \times p_2$, for general case, there

are infinite number of matrices $\mathbf{M}$ such that $\mathcal{P}_\Omega(\mathbf{M}) = \mathbf{M}_\Omega$, where $\mathcal{P}_\Omega$ is the projection of a given matrix on the subset Ω.

$$\text{(Observation)} \quad \mathbf{M}_\Omega = \begin{bmatrix} 3 & ? & 1 & 5 & 4 \\ 4 & 3 & ? & 4 & 3 \\ 2 & ? & 3 & ? & 5 \\ 1 & 2 & ? & 5 & 4 \\ 3 & ? & 4 & 4 & 3 \end{bmatrix} \quad \Rightarrow \quad \mathbf{X}^\star := \begin{bmatrix} 3 & 4 & 1 & 5 & 4 \\ 4 & 3 & 2 & 4 & 3 \\ 2 & 1 & 3 & 5 & 5 \\ 1 & 2 & 2 & 5 & 4 \\ 3 & 2 & 4 & 4 & 3 \end{bmatrix} \quad \text{(a constructed solution)}.$$

Hence, we have to impose some structure properties on our matrix, typically the low-rankness of $\mathbf{M}$. This assumption somehow makes sense. For example, we consider a movie rating problem, where it is very often that the good movie will be rated at a high rank with different users, while bad movies probably get a low rank from many users. Hence, the matrix presenting the rank of all movies w.r.t. users probably becomes low rank. In such a situation, we can formulate this problem into the following convex program:

$$\min_{\mathbf{X}\in\mathbb{R}^{p_1\times p_2}} \Big\{ \|\mathbf{X}\|_* \mid \mathcal{P}_\Omega(\mathbf{X}) = \mathbf{M}_\Omega \Big\}, \tag{26}$$

where the constraint $\mathcal{P}_\Omega(\mathbf{X}) = \mathbf{M}_\Omega$ means that $\mathbf{X}_{ij} = \mathbf{M}_{ij}$ for all $(i,j) \in \Omega$; and $\|\mathbf{X}\|_*$ is the nuclear norm of $\mathbf{X}$ that approximates the rank of $\mathbf{X}$. When we have noise, this constraint can be relaxed to $\|\mathcal{P}_\Omega(\mathbf{X}) - \mathbf{M}_\Omega\|_F \le \sigma$, where $\sigma > 0$ is a given noise level. Problem (26) looks very similar to the basis pursuit problem (BP). Hence, methods for solving (BP) can be adapted to solve (26). In addition, we can also consider different variants of (26) such as the basis pursuit denoising or the LASSO form.

Alternatively, if we consider the observed matrix $\mathbf{M}$ as the sum of a low-rank matrix $\mathbf{L}$ and a sparse matrix $\mathbf{S}$ such that $\mathbf{M} = \mathbf{L} + \mathbf{S}$. In applications such as background subtraction (foreground detection) the low-rank part represents the background which often unchanges, while the sparse part captures moving objects which often sparse. In this case, we want to solve the following problem:

$$\min_{\mathbf{S},\mathbf{L}\in\mathbb{R}^{p_1\times p_2}} \Big\{ \|\mathbf{S}\|_1 + \lambda \|\mathbf{L}\|_* \mid \mathbf{L} + \mathbf{S} = \mathbf{M} \Big\}, \tag{27}$$

Here, $\|\mathbf{X}\|_1 = \sum_{i=1}^{p_1}\sum_{j=1}^{p_2} |X_{ij}|$. This problem is also referred to as a **robust principal component analysis** (RPCA) [5]. In the case of noise, we can replace $\mathbf{L}+\mathbf{S} = \mathbf{M}$ by $\|\mathbf{L}+\mathbf{S}-\mathbf{M}\|_F \le \sigma$ with a given noise level $\sigma > 0$.

3.7 Sparse inverse covariance estimation in graphical models

Let us consider a Gaussian Markov random field of p nodes/variables from a dataset $\{Z_1, \cdots, Z_p\}$, where Z_j is a p-dimensional random vector with Gaussian distribution $\mathcal{N}(\mu, \Sigma)$. We consider the inverse covariance matrix Σ^{-1} of these variables, where $\Sigma^{-1}_{ij} = 0$ if Z_i is uncorrelated to Z_j or there is no edge between nodes i and node j, and $\Sigma^{-1}_{ij} > 0$ otherwise. Traditionally, we can estimate the covariance matrix Σ using n observations $\left\{Z_i^{(k)}\right\}_{k=1}^n$ generated from Z_i $(i = 1, \cdots, p)$ such as $\hat{\Sigma}_{ij} = \frac{1}{n-1}\sum_{k=1}^n (Z_i^{(k)} - \bar{Z}_i)(Z_j^{(k)} - \bar{Z}_j)$, where $\bar{Z}_i := \frac{1}{n}\sum_{k=1}^n Z_i^{(k)}$ is sample mean of Z_i. Then, we can compute an approximation of the inverse covariance Σ^{-1} by $\hat{\Sigma}^{-1}$.

Let us denote $\mathbf{Z} := (Z_1, \cdots, Z_p)$ to be a joint random vector. Then, the joint distribution of $\mathbf{Z}$ is

$$p(\mathbf{z}; \mu, \Sigma) = p(z_1, \cdots, z_p; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^p}\sqrt{\det(\Sigma)}} \exp\left(-\frac{1}{2}(\mathbf{z}-\mu)^\top \Sigma^{-1}(\mathbf{z}-\mu)\right).$$

Taking logarithm both sides and noting that $\det(\Sigma) = \frac{1}{\det(\Sigma^{-1})}$, and $\mathbf{a}^\top\mathbf{H}\mathbf{a} = \text{trace}\left((\mathbf{a}\mathbf{a}^\top)\mathbf{H}\right)$, we obtain

$$\log(p(\mathbf{z}; \mu, \Sigma)) = \frac{1}{2}\log(\det(\Sigma^{-1})) - \frac{1}{2}(\mathbf{z}-\mu)^\top\Sigma^{-1}(\mathbf{z}-\mu) - \frac{p}{2}\log(2\pi)$$

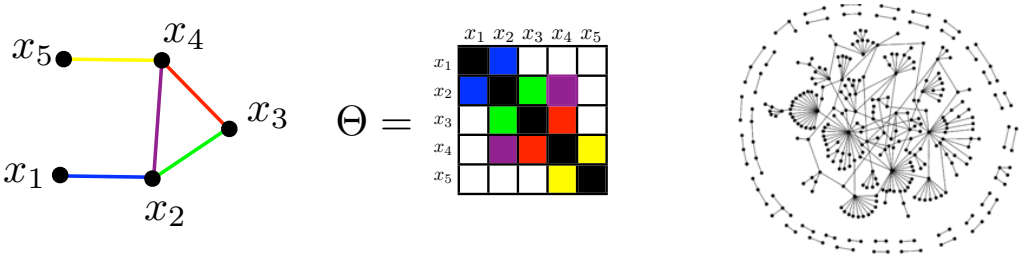
Now, if we use n sample points $\{\mathbf{Z}^{(1)}, \cdots, \mathbf{Z}^{(n)}\}$, then we have $\bar{\mathbf{Z}} := \frac{1}{n}\sum_{k=1}^{n}\mathbf{Z}^{(k)}$ is the sample mean, and the sample covariance is $\hat{\Sigma} := \frac{1}{n-1}\sum_{k=1}^{n}(\mathbf{Z}^{(k)} - \bar{\mathbf{Z}})(\mathbf{Z}^{(k)} - \bar{\mathbf{Z}})^\top$. By the maximum-likelihood principle, we have

$$\sum_{k=1}^{n}\log(p(\mathbf{Z}^{(k)})) = \frac{n}{2}\log(\det(\Sigma^{-1})) - \frac{1}{2}\mathrm{trace}\left(\sum_{k=1}^{n}((\mathbf{Z}^{(k)} - \mu)(\mathbf{Z}^{(k)} - \mu)^\top)\Sigma^{-1}\right) - \frac{np}{2}\log(2\pi)$$

Partly solve for μ from the maximum-likelihood principle we obtain a solution $\bar{\mathbf{Z}} = \frac{1}{n}\sum_{k=1}^{n}\mathbf{Z}^{(k)}$ for μ. If we define $\bar{\Sigma} := \frac{1}{n}\sum_{k=1}^{n}(\mathbf{Z}^{(k)} - \bar{\mathbf{Z}})(\mathbf{Z}^{(k)} - \bar{\mathbf{Z}})^\top)$, and $\mathbf{X} := \Sigma^{-1} \in \mathcal{S}^p_{++}$. The maximum-likelihood problem can be written equivalently to the following convex problem:

$$\min_{\mathbf{X}\in\mathcal{S}^p_{++}}\left\{-\log\det\mathbf{X} + \mathrm{trace}\left(\bar{\Sigma}\mathbf{X}\right)\right\}. \tag{28}$$

Unfortunately, when $p \ll n$, $\bar{\Sigma} \approx \hat{\Sigma}$ is not invertible. Hence, we are not able to estimate $\bar{\Sigma}^{-1}$ using (28).



We can think of a pseudo-inverse, but this one is often unstable and is not positive definite. Hence, it is better to formulate the problem of estimating an inverse covariance Σ^{-1} into an optimization problem, where we want to impose the sparsity by adding an ℓ_1-regularizer. In this case, this problem can be formulated as

$$\min_{\mathbf{X}\in\mathcal{S}^p_{++}}\left\{-\log\det\mathbf{X} + \mathrm{trace}\left(\bar{\Sigma}\mathbf{X}\right) + \lambda\left\|\mathrm{vec}\left(\mathbf{X}\right)\right\|_1\right\},$$

where $\lambda > 0$ is a regularization parameter to tune the desired sparsity level, and $\mathrm{vec}(\cdot)$ is the vectorization operator of matrices. We note that the objective function $f(\mathbf{X}) := -\log\det\mathbf{X} + \mathrm{trace}\left(\bar{\Sigma}\mathbf{X}\right)$ is differentiable, but its gradient $\nabla f(\mathbf{X})$ is not Lipschitz continuous. Hence, existing first-order methods from the literature do not have a theoretical guarantee when solving this problem. Fortunately, f is self-concordant (see [27]), which can be efficiently solved by a special second-order method.

3.8 Empirical risk minimization

The empirical risk minimization problem can be obtained from a sample averaging approach in stochastic optimization, which can be described as follows. Let $f(\mathbf{x}, \xi)$ be a bifunction representing risk or loss defined on $\mathbb{R}^n \times \Omega$, where ξ is a random variable taking values in Ω computed from a given statistical learning model (e.g., $f(\mathbf{x}, \xi) = \ell(\varphi(\mathbf{x}; \mathbf{w}), y)$). Clearly, for each fixed $\mathbf{x}$, $f(\mathbf{x}, \cdot)$ is also a random variable. For a given realization $\xi \in \Omega$, we often assume that $f(\cdot, \xi)$ is convex. Since ξ is random, we want to solve the problem on the expected value $\mathbb{E}_\xi\left[f(\mathbf{x}, \xi)\right]$ as

$$\min_{\mathbf{x}\in\mathbb{R}^p}\left\{\bar{f}(\mathbf{x}) := \mathbb{E}_\xi\left[f(\mathbf{x}, \xi)\right]\right\}.$$

Unfortunately, evaluating the expected value $\mathbb{E}_\xi\left[f(\mathbf{x}, \xi)\right]$ is often impractical in many applications. We instead replace this expected function by its sample mean. This leads us to the following convex empirical risk minimization problem:

$$\min_{\mathbf{x}\in\mathbb{R}^p}\left\{\bar{f}(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n}f(\mathbf{x}, \xi^{(i)})\right\}, \tag{29}$$

where $\left\{\xi^{(1)}, \cdots, \xi^{(n)}\right\}$ are n realizations of the random vector ξ. In order to simplify the notation, we often define $f_i(\mathbf{x}) := f(\mathbf{x}, \xi^{(i)})$ and write this problem as

$$\min_{\mathbf{x}\in\mathbb{R}^p}\left\{\bar{f}(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x})\right\}.$$

The latter problem is not only an empirical risk minimization problem, but it is also convenient to formulate other problems such as distributed optimization, optimization over networks, optimization over graphs, and consensus optimization problems.

Another way to derive this problem is from a supervised machine learning viewpoint. Here, we want to learn a hypothesis $\mathbf{x} : U \to Y$ (as a function) from a collection of observations $\left\{(\mathbf{u}^{(i)}, \mathbf{y}_i)\right\}_{i=1}^n$ i.i.d. samples drawn from a joint distribution $P(\mathbf{u}, \mathbf{y})$ of $\mathbf{u}$ and $\mathbf{y} = \mathbf{x}(\mathbf{u})$. We consider a loss function $\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$ that measure the error between $\hat{\mathbf{y}}$ and its true outcome $\mathbf{y}$. The total loss (or risk) associated with $\mathbf{x}$ is defined as the expected value of the loss, which is $f(\mathbf{x}) := \mathbb{E}\left[\mathcal{L}(\mathbf{x}(\mathbf{u}), \mathbf{y})\right]$. Similar to the above argument, we instead replace this expectation by its sample averaging quantity to obtain

$$\min_{\mathbf{x}\in\mathbb{R}^p}\left\{\bar{f}(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(\mathbf{x}(\mathbf{u}^{(i)}), \mathbf{y}^{(i)})\right\}. \tag{30}$$

Recently, (30) is often studied combining with a regularization term in order to obtain a desired structure in its solutions. Several examples of loss functions such as logistic regression, ridge regression, hinge loss, and Poisson models can be formulated into (30).

3.9 Logistic regression

Let X be an input variable and Y be a response. We consider the conditional probability $\mathbb{P}(Y|X)$ of the response Y given the input X. In binary classification problems, the response Y takes the values in $\{0, 1\}$. Hence, the probability $\mathbb{P}(Y|X)$ is modeled using a logistic function as

$$\mathbb{P}(Y = 1|X = \mathbf{x}) = \frac{1}{1 + e^{-(\beta_0 + \beta^\top\mathbf{x})}}, \quad \text{and} \quad \mathbb{P}(Y = 0|X = \mathbf{x}) = 1 - \frac{1}{1 + e^{-(\beta_0 + \beta^\top\mathbf{x})}},$$

where $\mathbf{x}$ is a realization of X, β_0 is an intercept, and β is a vector of model parameters.

By the maximum log-likelihood principle, we can solve for $\beta = (\beta_0, \cdots, \beta_p)^\top$ from the following optimization problem:

$$\max_{\beta\in\mathbb{R}^{p+1}} \log\left(\prod_{i:\mathbf{y}_i=1}\left(\frac{1}{1 + e^{-(\beta_0 + \sum_{i=1}^p \beta_i\mathbf{x}_i)}}\right)\prod_{i:\mathbf{y}_i=0}\left(1 - \frac{1}{1 + e^{-(\beta_0 + \sum_{i=1}^p \beta_i\mathbf{x}_i)}}\right)\right).$$

This problem can be reformulated equivalently to

$$\min_{\beta\in\mathbb{R}^{p+1}}\left\{\sum_{i=1}^{n}\log\left(1 + e^{(\beta_0 + \sum_{i=1}^p \beta_i\mathbf{x}_i)}\right) - \sum_{i=1}^{n} y_i\left(\beta_0 + \sum_{j=1}^{p}\beta_j\mathbf{x}_j\right)\right\}.$$

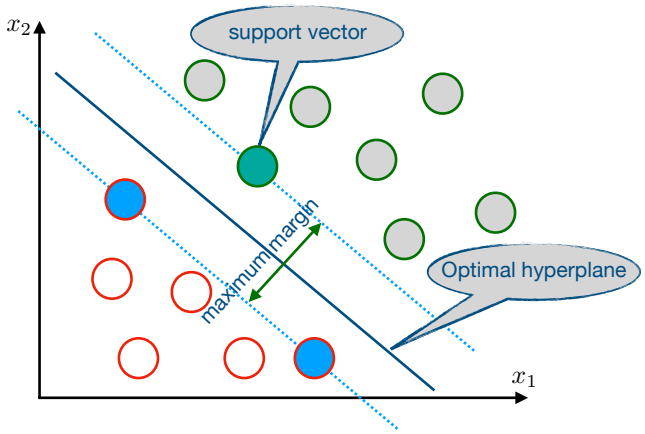
Such a problem is certainly convex and smooth, which can be solved efficiently by gradient-type methods as can be seen in the sequel. In practice, one can usually add a regularization term $\lambda R(\beta)$ to characterize certain properties of solutions or to prevent overfitting. Hence, this problem often leads to

$$\min_{\beta\in\mathbb{R}^{p+1}}\left\{\sum_{i=1}^{n}\log\left(1 + e^{(\beta_0 + \sum_{i=1}^p \beta_i\mathbf{x}_i)}\right) - \sum_{i=1}^{n} y_i\left(\beta_0 + \sum_{j=1}^{p}\beta_j\mathbf{x}_j\right) + \lambda R(\beta)\right\},$$

where $R(\cdot)$ is often a convex function and can be nonsmooth, and $\lambda > 0$ is a regularization parameter to tune.

3.10 Support vector machine

We are given a set of data points $\{(\mathbf{x}^{(i)}, y_i)\}_{i=1}^n$, where $\mathbf{x} \in \mathbb{R}^p$ and the label $y_i \in \{-1, 1\}$. Assume that we draw $\{\mathbf{x}^{(i)}\}$ in the $\mathbb{R}^p$ space, and want to find a hyperplane $\mathcal{H} : \mathbf{w}^\top \mathbf{x} + b = 0$ that separates all the points associated with a label $y_i = +1$ into one half-plane and the others into the other half-plane, where $\mathbf{w} \in \mathbb{R}^p$ is its normal vector (later, we call it a support vector), and $b \in \mathbb{R}$ is an intercept. However, there are points which may stay on this hyperplane, and we cannot classify them correctly. In this case, we want to build a hyperplane that



has maximum margin to the points in this dataset. Without loss of generality, we suppose that there are two hyperplanes $\mathbf{w}^\top \mathbf{x} + b = 1$ and $\mathbf{w}^\top \mathbf{x} + b = -1$ that are parallel to $\mathcal{H}$ and such they separate this data set into two half-planes. That is

$$\begin{cases} \mathbf{w}^\top \mathbf{x}^{(i)} + b \geq 1 & \text{if } y_i = 1, \\ \mathbf{w}^\top \mathbf{x}^{(i)} + b \leq -1 & \text{otherwise.} \end{cases} \quad \Leftrightarrow \quad \mathbf{y}_i(\mathbf{w}^\top \mathbf{x}^{(i)} + b) \geq 1, \quad \forall i = 1, \cdots, n.$$

The distance between the two latter hyperplanes is given by $\frac{2}{\|\mathbf{w}\|}$ (why?), and we want to maximize it. This problem turns out to be minimizing $\|\mathbf{w}\|_2$, or equivalently, minimizing $\|\mathbf{w}\|_2^2$. Hence, we can formulate this support vector machine problem as a special quadratic program:

$$\min_{\mathbf{w} \in \mathbb{R}^p} \left\{ \frac{1}{2} \|\mathbf{w}\|_2^2 \mid \mathbf{y}_i(\mathbf{w}^\top \mathbf{x}^{(i)} + b) \geq 1, \quad \forall i = 1, \cdots, n \right\} \tag{31}$$

In reality, we do not have perfect classification of this data set, we accept some misclassification of a few points. We can relax this problem into the following with some penalty constant $C > 0$:

$$\min_{\mathbf{w} \in \mathbb{R}^p, \eta \in \mathbb{R}^n_+} \left\{ \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^n \eta_i \mid \mathbf{y}_i(\mathbf{w}^\top \mathbf{x}^{(i)} + b) \geq 1 - \eta_i, \quad \forall i = 1, \cdots, n \right\}. \tag{32}$$

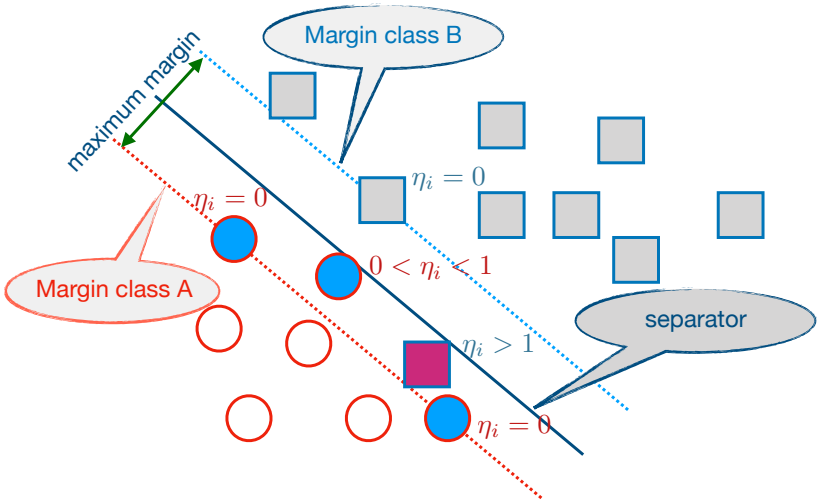
This problem is again a quadratic program with the joint variable $(\mathbf{w}, \eta) \in \mathbb{R}^{p+n}$. If the number of data points n is large, (31) is a strongly convex QP with many constraints, while (32) is only convex QP problem in a high-dimensional space and with many constraints.

4 Representative nonconvex optimization models

The class of nonconvex optimization problems is very large, even in the continuous case. In this course, we only focus on certain subclasses of nonconvex optimization problems that are widely used in practice.

4.1 biconvex optimization

Given two subsets $\mathcal{X}$ and $\mathcal{Y}$ in $\mathbb{R}^p$ and $\mathbb{R}^q$, respectively, we consider a bi-function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ on $\mathcal{X} \times \mathcal{Y}$ is defined as a mapping from $(\mathbf{x}, \mathbf{y}) \mapsto f(\mathbf{x}, \mathbf{y})$ with two groups of variables $\mathbf{x}$ and $\mathbf{y}$. We say that $f(\cdot, \cdot)$ is biconvex



on $\mathcal{X} \times \mathcal{Y}$ if for any fixed $\mathbf{x} \in \mathcal{X}$, $f(\mathbf{x}, \cdot)$ is convex, and for any fixed $\mathbf{y} \in \mathcal{Y}$, $f(\cdot, \mathbf{y})$ is convex [10]. Examples of biconvex functions include $f(\mathbf{x}, \mathbf{y}) := \mathbf{x}^\top \mathbf{y}$ from $\mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ as a bilinear function, or $f(\mathbf{X}, \mathbf{Y}) := \|\mathbf{X}\mathbf{Y}^\top - \mathbf{B}\|_F^2$ from $\mathbb{R}^{p\times r} \times \mathbb{R}^{q\times r} \to \mathbb{R}$ is a biconvex quadratic function. We note that, we can also define similar concepts such as biconvex sets, bi-linear, bi-affine, and bi-concave functions and their corresponding optimization problems, see [10] for more details.

Optimization problems involving biconvex functions can be stated into the following form:

$$\begin{cases} \min\limits_{\mathbf{x}\in\mathbb{R}^p, \mathbf{y}\in\mathbb{R}^q} & f_0(\mathbf{x}, \mathbf{y}) \\ \text{subject to} & f_i(\mathbf{x}, \mathbf{y}) \le 0, \ i = 1, \cdots, m, \\ & (\mathbf{x}, \mathbf{y}) \in \Omega, \end{cases} \tag{33}$$

where f_i for $i = 0, \cdots, m$ are biconvex, and Ω is a nonempty, closed, and convex set. biconvex optimization has many applications in different fields such as multi-facility location, medical image registration, control theory, and model reduction in multivariate data analysis [10].

As a concrete example of the biconvex optimization model (33), we consider the following well-known nonnegative factorization problem:

$$\min_{\mathbf{L}\in\mathbb{R}^{p\times r}, \mathbf{R}\in\mathbb{R}^{q\times r}} \left\{ \tfrac{1}{2}\|\mathbf{L}\mathbf{R}^\top - \mathbf{M}\|_F^2 \mid \mathbf{L} \ge 0, \mathbf{R} \ge 0 \right\}, \tag{34}$$

Here, $\mathbf{L} \ge 0$ means that $\mathbf{L}_{ij} \ge 0$ for $i = 1, \cdots p$ and $j = 1, \cdots, r$. If $r \ll p, q$, then this problem becomes a low-rank nonnegative matrix factorization, where we approximate $\mathbf{M}$ by a low-rank matrix $\mathbf{L}\mathbf{R}^\top$. In particular, this problem has many application in multivariate and data analysis such as astronomy, computer vision, document clustering, chemometrics, audio signal processing, recommender systems, and bioinformatics [16, 30].

As another example is the following bilinear program:

$$\min_{\mathbf{X}\in\mathbb{R}^{p\times q}} \left\{ \text{trace}\left(\mathbf{X}^\top \mathbf{C}\mathbf{X}\right) \ \mid \ \text{trace}\left(\mathbf{X}^\top \mathbf{A}_i \mathbf{X}\right) = b_i, \ i \in \mathcal{E}, \ \text{trace}\left(\mathbf{X}^\top \mathbf{B}_j \mathbf{X}\right) \le c_j, \ j \in \mathcal{I} \right\}, \tag{35}$$

where $\mathbf{C}$, $\mathbf{A}_i$, and $\mathbf{B}_j$ are given matrices, and b and c are given vectors with appropriate dimensions. This problem often appears in economics, engineering, and computer vision applications.

4.2 D.C. (or convex-concave) programming

A d.c. (difference-of-two-convex-functions) function is a function of the form $f(\mathbf{x}) := g(\mathbf{x}) - h(\mathbf{x})$, where $h, g : \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ are two proper, closed, and convex functions. The class of d.c. functions is rather large, which

covers all twice continuously differentiable functions [28]. For a d.c. function $f(\mathbf{x}) = g(\mathbf{x}) - h(\mathbf{x})$, we call (g, h) is its d.c. representation. Any d.c. function has infinite number of d.c. representations since we can always write it as $f(\mathbf{x}) = g(\mathbf{x}) - h(\mathbf{x}) = (g(\mathbf{x}) + m(\mathbf{x})) - (h(\mathbf{x}) + m(\mathbf{x}))$ for any convex function m. Although a d.c. function has infinite number of d.c. representation, finding one d.c. representation of a general function f is still a challenging problem. Note that d.c. functions are also known as convex-concave functions in the literature [23].

Let us provide some examples of d.c. functions as follows:

1. Any convex or concave function is also d.c. Indeed, if f is convex, we can write $f(\mathbf{x}) = f(\mathbf{x}) - 0(\mathbf{x})$, and if f is concave, we can write $f(\mathbf{x}) = 0(\mathbf{x}) - f(\mathbf{x})$, where $0(\mathbf{x})$ is the zero function.
2. *Bilinear functions:* A bilinear function $f(\mathbf{x}, \mathbf{y}) = 4\mathbf{x}^\top\mathbf{y}$ is d.c.. Indeed, we can write $4\mathbf{x}^\top\mathbf{y} = \|\mathbf{x} + \mathbf{y}\|^2 - \|\mathbf{x} - \mathbf{y}\|^2$ as one d.c. representation.

D.C. programs are optimization problems involving d.c. functions, and are stated as follows:

$$\left\{\begin{array}{ll} \min\limits_{\mathbf{x}\in\mathbb{R}^p} & f_0(\mathbf{x}) \\ \text{subject to} & f_i(\mathbf{x}) \le 0,\ i = 1, \cdots, m, \\ & \mathbf{x} \in \mathcal{X}, \end{array}\right. \tag{36}$$

where $f_i : \mathbb{R}^p \to \mathbb{R}$ $(i = 0, \cdots, m)$ are d.c. functions, and $\mathcal{X}$ is a nonempty, closed, and convex set. Again D.C. programming is also referred to as convex-concave optimization in the literature [23]. Theory, numerical methods, and applications of d.c. functions and D.C. programming can be found, e.g., in [8, 25].

Let us provide some examples of D.C. programming as follows. Our first example is a boolean least-squares problem of the form [23]:

$$\min_{\mathbf{x}\in\mathbb{R}^p}\Big\{\tfrac{1}{2}\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 \mid \mathbf{x}_i^2 = 1,\ i = 1, \cdots, p\Big\}, \tag{37}$$

where $\mathbf{A} \in \mathbb{R}^{n\times p}$ and $\mathbf{b} \in \mathbb{R}^n$ are given. Here, the objective function $f_0(\mathbf{x}) = \frac{1}{2}\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2$ is convex. The constraint $\mathbf{x}_i^2 = 1$ can be written as $\mathbf{x}_i^2 \le 1$ and $-\mathbf{x}_i^2 \le -1$ as d.c. constraints.

The second example is the following sparse principal component analysis model:

$$\max_{\mathbf{x}\in\mathbb{R}^p}\Big\{\mathbf{x}^\top\mathbf{S}\mathbf{x} \mid \|\mathbf{x}\|_2 = 1,\ \ \|\mathbf{x}\|_1 \le \mu\Big\}, \tag{38}$$

where $\mathbf{S} \in \mathbb{R}^{p\times p}$ is a symmetric matrix, and $\mu \ge 0$ is a given threshold to control the sparsity level. The last constraint $\|\mathbf{x}\|_1 \le \mu$ is the convexification of the cardinality constraint $\|\mathbf{x}\|_0 \le \mu$, which makes the problem to be tractable. Clearly, the function $f_0(\mathbf{x}) = \mathbf{x}^\top\mathbf{S}\mathbf{x}$ is a d.c. function since $f_0(\mathbf{x}) = \mathbf{x}^\top\mathbf{S}_+\mathbf{x} - \mathbf{x}^\top\mathbf{S}_-\mathbf{x}$, where $\mathbf{S}_+$ and $\mathbf{S}_-$ are the positive and negative decompositions of $\mathbf{S}$, respectively, i.e., $\mathbf{S}_+ := \mathbf{V}[\Sigma]_+\mathbf{V}^\top$ and $\mathbf{S}_- := \mathbf{V}[-\Sigma]_+]\mathbf{V}^\top$, where $\mathbf{S} = \mathbf{V}\Sigma\mathbf{V}^\top$ is the eigen-decomposition of $\mathbf{S}$. The constraint $\|\mathbf{x}\|_2 = 1$ can be written as $\|\mathbf{x}\|_2 \le 1$ and $-\|\mathbf{x}\|_2 \le -1$ as d.c. constraints.

4.3 Composite optimization

Our next example is the following composite non-convex optimization model:

$$f^\star := \min_{\mathbf{x}\in\mathbb{R}^p}\left\{f(\mathbf{x}) := \varphi(F(\mathbf{x})) \mid \mathbf{x} \in \mathcal{X}\right\}, \tag{39}$$

where $\varphi : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is proper, closed, and convex function, $F : \mathbb{R}^p \to \mathbb{R}^n$ is a smooth function, but not necessarily linear, and $\mathcal{X}$ is a nonempty, closed, and convex set in $\mathbb{R}^p$. This model looks very simple but it covers a wide range of applications in parameter estimation, model predictive control and matrix approximation.

One typical example of (39) is the following parameter estimation problem:

$$\min_{\mathbf{x}\in\mathbb{R}^p}\sum_{j=1}^{n}\left(F_j(\mathbf{W}_j; \mathbf{x}) - \hat{\mathbf{y}}_j\right)^2, \tag{40}$$

where F_i is a mapping from the input $\mathbf{W}_j$ modeled with a parameter $\mathbf{x}$ to $\mathbf{y}_j = F_j(\mathbf{W}_j;\mathbf{x})$, and $\hat{\mathbf{y}}_j$ is an observed output for $j = 1,\cdots,n$. Here, we are more interested in the nonlinear model F_j than a linear one as in least-squares model. Problem (40) is also known as non-linear least-squares problem, and can be cast into (39).

As another concrete example, the following matrix factorization problem can also be cast into (39):

$$\min_{\mathbf{L}\in\mathbb{R}^{p\times r},\mathbf{R}\in\mathbb{R}^{q\times r}}\left\{\tfrac{1}{2}\|\mathbf{L}\mathbf{R}^\top - \mathbf{M}\|_F^2 \mid \mathbf{L}\in\mathcal{X},\mathbf{R}\in\mathcal{Y}\right\}, \tag{41}$$

where $\mathbf{M}$ is a given $p\times q$ input matrix, and $\mathcal{X}$ and $\mathcal{Y}$ are given convex sets. Clearly, by setting $\varphi(\cdot) := \frac{1}{2}\|\cdot\|_F^2$ and $F(\mathbf{L},\mathbf{R}) := \mathbf{L}\mathbf{R}^\top - \mathbf{M}$, we can cast this problem into (39).

4.4 Nonlinear optimization in deep neural networks

Our last example is an optimization model in deep neural networks. Neural networks are often designed to perform a complicated task in machine learning as well as many other areas. Mathematically, we can consider a neural network as a "black box" model to approximate a complex function of multiple inputs and outputs. We believe this model works due to a well-known theorem called "universal approximation theorem", which roughly states that any continuous function on a compact set can be approximated by a neural network with only one hidden layer. Neural networks have a long history starting around 1940s with electronic brains, and become extremely active since 2010s under a new name: deep neural networks.

A common deep neural networks architecture consists of three components: an input layer with one or many input nodes, an output layer with one or many output nodes, and some or many hidden layers (can be no hidden layer). Input nodes are connected to the hidden layers with direct links to propagate the inputs into the network, and each hidden layer also connects to its next hidden layers by forward links. The output layer is connected to the hidden layers to obtain the outputs [15]. Figure 2 illustrates a multilayers feedforward deep neural network architecture.

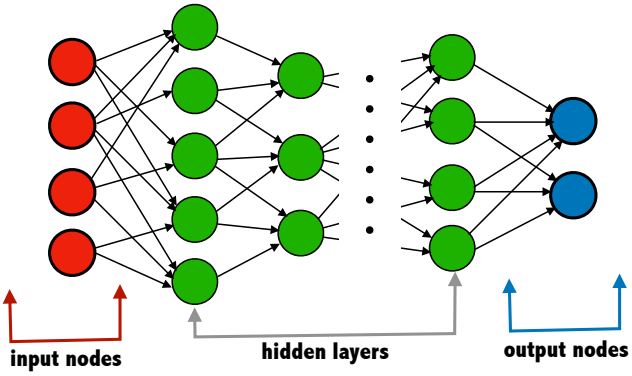


Figure 2: An illustration of a feedforward deep neural network.

A mathematical formulation of the output at each node j of the layer l ($l = 1,\cdots,L$) is given by the following expression:

$$\mathbf{x}_j^l = \sigma(\mathbf{w}_j^l\mathbf{x}^{l-1} + \mu_j^l), \quad l = 1,\cdots,L, \tag{42}$$

where $\mathbf{w}_j^l$ is a vector in $\mathbb{R}^{|\mathbf{x}^{l-1}|}$, which is referred to as a weight vector of the layer l with respect to the output node j, and μ_j^l is an intercept of node j (also called bias), σ is a given nonlinear function, called an activation function, and L is the number of hidden layers. If we denote by $\mathbf{W}^l$ a matrix formed from $\mathbf{w}_j^l$ for $j = 1,\cdots,|\mathbf{x}^l|$, then we can write the output at the l-th layer as

$$\mathbf{x}^l = \sigma(\mathbf{W}^l\mathbf{x}^{l-1} + \mu^l), \quad l = 1,\cdots,L.$$

Here, μ^l is the vector of biases. Clearly, x_j^0 is the j-th input node, which can be denoted by x_j, and x_j^L is the j-th output node. Common activation functions include

- ReLU function (rectified linear function): $\sigma(\tau) = \max\{0, \tau\}$.
- Sigmoid function: $\sigma(\tau) = \frac{1}{1+e^{-\tau}}$
- Bernoulli: A random function that outputs 1 with probability $\frac{1}{1+e^{-\tau}}$, and 0 otherwise.

The parameters of a deep neural network is a collection of weight matrices W^l and intercept vectors μ^l at all layers $l = 1, \cdots, L$. That is

$$\mathbf{w} := [\mathbf{W}^1, \mu^1, \mathbf{W}^2, \mu^2, \cdots, \mathbf{W}^L, \mu^L] \in \mathbb{R}^p,$$

where $p = |\mathbf{x}^0| \times |\mathbf{x}^1| + |\mathbf{x}^1| + \cdots + |\mathbf{x}^{L-1}| \times |\mathbf{x}^L| + |\mathbf{x}^L|$. If we consider the presentation of the output $\mathbf{x}^L$ as a function h of the input $\mathbf{x}$ depending on the parameter $\mathbf{w}$ defined above, then, we can write it as

$$\mathbf{x}^L = h(\mathbf{x}; \mathbf{w}).$$

Given a set of train examples $\left\{(\bar{\mathbf{x}}^{(k)}, \mathbf{y}^{(k)})\right\}_{k=1}^N$, where $\mathbf{y}^{(k)}$ represents the label of input data $\bar{\mathbf{x}}^{(k)}$ for $k = 1, \cdots, N$, we consider the following optimization model:

$$\min_{\mathbf{w}\in\mathbb{R}^p} \frac{1}{N} \sum_{k=1}^N \ell\left(h(\bar{\mathbf{x}}^{(k)}; \mathbf{w}), \mathbf{y}^{(k)}\right), \tag{43}$$

where ℓ is a given loss function that measures the mismatch between the output $\mathbf{x}^L$ and the label vector $\mathbf{y}$. Due to the nonlinearity of the activation functions σ and the large number of layers and weights, problem (43) is often highly nonlinear and very challenging for optimization methods. Several loss functions ℓ can be used. The most common one is the cross-entropy loss function, which is defined as $\ell(u, v) = -\sum_\tau u(\tau)\log(v(\tau))$, where u is the true distribution, and v is an estimate.

Deep neural networks have been extensively studied in recent years, and become extremely active with many remarkable applications. However, in order to obtain a successful deep neural network application, the following components are fundamental:

1. Good neural network architectures and clever parameter constraints
2. Massive and well-labeled datasets
3. Advanced computation infrastructure, and sharing stable and reliable computational toolboxes.
4. Good training algorithms (i.e., efficient optimization algorithms)

Deep neural networks work well in several imaging applications. One of successful applications is image classification/captioning/segmentation using massive datasets such as ImageNet http://www.image-net.org or OpenImage https://github.com/openimages/dataset. Other applications include speech and pattern recognition, recommendation systems, natural language processing, drug discovery and toxicology, and mobile advertising. Together with applications, new architectures (e.g., AlexNet, ResNet, DenseNet), computational toolboxes (e.g., TensorFlow, Theano, Caffe, Keras, and PyTorch), supporting theory, methodology as well as new models for deep learning are really well developed in the last few years.

References

[1] E.J. Anderson and P. Nash. *Linear Programming in Infinite-Dimensional Spaces*. Wiley, 1987.

[2] K. Banaszek, G. M. DAriano, M. G. A. Paris, and M. F. Sacchi. Maximum-likelihood estimation of the density matrix. *Phys. Rev. A.*, 61(010304):010304–1 – 010304–4, 1999.

[3] Dimitris Bertsimas and Melvyn Sim. Robust discrete optimization and network flows. *Math. Program.*, 98(1-3):49–71, 2003.

[4] E. Candes and T. Tao. The Dantzig selector: Statistical estimation when p is much larger than n. *Annals of Statistics*, 35(6):2313–2351, 2007.

[5] E.J. Candés, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 58(3):1–37, 2011.

[6] A. S. Dalalyan, M. Hebiri, K. Meziani, and J. Salmon. Learning heteroscedastic models by convex programming under group sparsity. *Proc. of the International conference on Machine Learning*, pages 1–8, 2013.

[7] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.

[8] Tao Pham Dinh and Hoai An Le Thi. Recent advances in dc programming and dca. In *Transactions on Computational Intelligence XIII*, pages 1–37. Springer, 2014.

[9] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer-Verlag, New York, 2001.

[10] J. Gorski, F. Pfeuffer, and K. Klamroth. Biconvex sets and optimization with biconvex functions: a survey and extensions. *Mathematical methods of operations research*, 66(3):373–407, 2007.

[11] M. Grant, S. Boyd, and Y. Ye. Disciplined convex programming. In L. Liberti and N. Maculan, editors, *Global Optimization: From Theory to Implementation*, Nonconvex Optimization and its Applications, pages 155–210. Springer, 2006.

[12] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 2012.

[13] Z.T. Harmany, R.F. Marcia, and R. M. Willett. This is SPIRAL-TAP: Sparse Poisson Intensity Reconstruction Algorithms - Theory and Practice. *IEEE Trans. Image Process.*, 21(3):1084–1096, 2012.

[14] D. Klatte and B. Kummer. *Nonsmooth Equations in Optimization: Regularity, Calculus, Methods and Applications*. Kluwer Academic Publishers, Dordrecht, 2002.

[15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[16] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[17] Jon Lee. *A first course in combinatorial optimization*, volume 36. Cambridge University Press, 2004.

[18] D. G. Luenberger and Y. Ye. *Linear and Nonlinear Programming*. Springer, 2007.

[19] George L Nemhauser and Laurence A Wolsey. Integer programming and combinatorial optimization. *Wiley, Chichester. GL Nemhauser, MWP Savelsbergh, GS Sigismondi (1992). Constraint Classification for Mixed Integer Programming Formulations. COAL Bulletin*, 20:8–12, 1988.

[20] Y. Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87 of *Applied Optimization*. Kluwer Academic Publishers, 2004.

[21] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1982.

[22] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[23] X. Shen, S. Diamond, Y. Gu, and S. Boyd. Disciplined convex-concave programming. In *IEEE 55th Conference on Decision and Control (CDC)*, pages 1009–1014. IEEE, 2016.

[24] N. Städler, P. Bülmann, and S. Van de Geer. l_1-penalization for mixture regression models. *Tech. Report.*, pages 1–35, 2012.

[25] Pham Dinh Tao et al. The dc (difference of convex functions) programming and dca revisited with dc models of real world nonconvex optimization problems. *Annals of operations research*, 133(1-4):23–46, 2005.

[26] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[27] Q. Tran-Dinh, A. Kyrillidis, and V. Cevher. Composite self-concordant minimization. *J. Mach. Learn. Res.*, 15:374–416, 2015.

[28] Hoang Tuy. *Convex analysis and global optimization*. Springer, 1998.

[29] R.J. Vanderbei. *Linear Programming: Foundations and Extensions*. Springer, 2015.

[30] Y. Wang and Y.-J. Zhang. Nonnegative matrix factorization: A comprehensive review. *IEEE Transactions on Knowledge and Data Engineering*, 25(6):1336–1353, 2013.