I. Introduction to Convex Optimization

Introduction to Optimization

Optimization problems are ubiquitous in science and engineering.

Optimization problems arise any time we have a collection of elements and wish to select the "best" one (according to some criterion). The process of casting a real world problem as being one of mathematical optimization consists of three main components

1. a set of variables, often called **decision variables**, that we have control over;
2. an **objective function** that maps the decision variables to some quality that we want to maximize (goodness of fit, profit, etc.) or some cost that we want to minimize (error, loss, etc.); and
3. a **constraint set** that dictates restrictions on the decision variables imposed by physical limitations, budgets on resources, design requirements, etc.

In its most general form, we can express such an optimization problem mathematically as

$$\underset{\boldsymbol{x}}{\text{minimize}} \;\; f(\boldsymbol{x}) \;\; \text{subject to} \;\; \boldsymbol{x} \in \mathcal{X}, \tag{1}$$

where $f : \mathcal{X} \to \mathbb{R}$ is our objective function and $\mathcal{X}$ is our constraint set.

In order to solve this optimization problem, we must find an $\widehat{\boldsymbol{x}} \in \mathcal{X}$ such that

$$f(\widehat{\boldsymbol{x}}) \leq f(\boldsymbol{x}) \text{ for all } \boldsymbol{x} \in \mathcal{X}. \tag{2}$$

We call an $\widehat{\boldsymbol{x}}$ satisfying (2) a **minimizer** of f in $\mathcal{X}$, and a **solution** to the optimization problem (1).

Georgia Tech ECE 6270 notes by M. A. Davenport, M. B. Egerstedt, and J. Romberg. Last updated 22:02, January 19, 2021

By convention, we will focus only on *minimization* problems, noting that $\widehat{\boldsymbol{x}}$ *maximizes* f in $\mathcal{X}$ if and only if $\widehat{\boldsymbol{x}}$ minimizes $-f$ in $\mathcal{X}$ — thus any maximization problem can be easily turned into an equivalent minimization problem.

There are a number of fundamental questions that arise when considering an optimization problem of the form (1):

1. **Existence.** Does a solution to (1) even exist? It could be that f is not bounded from below, or that $\mathcal{X}$ has been defined in such a way as to be empty. How can we guarantee the existence of a solution?
2. **Uniqueness.** Note that an $\widehat{\boldsymbol{x}}$ satisfying (2) need not be unique. Only when the inequality is strict can we conclude that there is a unique (strict) minimizer. When can we conclude that there is a unique solution?
3. **Verification.** Given a candidate solution $\widehat{\boldsymbol{x}}$, is there a simple condition we can check to determine if it is a/the solution to to (1)?
4. **Solution.** Can we find a closed-form expression for a/the solution to (1)? Can we provide an efficient algorithm for computing a/the solution to (1)?

Throughout this course we will devote significant attention to all of these questions, primarily in the context of **convex** problems.

Convex Optimization

The great watershed in optimization is not between linearity and non-linearity, but convexity and non-convexity.

— R. Tyrrell Rockafellar

Solving optimization problems is in general very difficult. In this class, we will develop a framework for analyzing and solving **convex** programs.[1] To state precisely what we mean by this, recall that a set $\mathcal{C}$ is convex if

$$\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C} \quad \Rightarrow \quad (1-\theta)\boldsymbol{x} + \theta \boldsymbol{y} \in \mathcal{C}$$

for all $\theta \in [0,1]$. A function f is convex if

$$f(\theta \boldsymbol{x} + (1-\theta)\boldsymbol{y}) \ \leq \ \theta f(\boldsymbol{x}) + (1-\theta) f(\boldsymbol{y})$$

for all $\boldsymbol{x}, \boldsymbol{y}$ and for all $\theta \in [0,1]$. (If either of these notions are new to you, don't worry. We will have much more on this later!) With these definitions in hand, a convex program simply corresponds to one where

1. The constraint set $\mathcal{X}$ is a convex subset of a real vector space (in this class we will focus exclusively on $\mathcal{X} \subseteq \mathbb{R}^N$).
2. The objective function $f : \mathcal{X} \to \mathbb{R}$ is a convex function.

Typically, we will rely on $\mathcal{X}$ to be specified by a series of constraint functionals:

$$\boldsymbol{x} \in \mathcal{X} \quad \Leftrightarrow \quad g_m(\boldsymbol{x}) \leq b_m \ \text{ for } m = 1, \ldots, M.$$

[1] Throughout this course will use the terminology "optimization/convex *program*" interchangeably with "optimization/convex problem."

Georgia Tech ECE 6270 notes by M. A. Davenport, M. B. Egerstedt, and J. Romberg. Last updated 22:02, January 19, 2021

In this case, an equivalent way to characterize a convex program is for each of the g_m to be convex functions.

What does convexity tell us? Two important things:

- Local minimizers are also global minimizers. So we can check if a certain point is optimal by looking in a small neighborhood and seeing if there is a direction to move that decreases f.
- First-order necessary conditions for optimality turn out to be sufficient. For example, when the problem is unconstrained and smooth, this means we can find an optimal point by finding $\widehat{\boldsymbol{x}}$ such that $\nabla f(\widehat{\boldsymbol{x}}) = \mathbf{0}$.

The upshot of these two things is that if $f(\boldsymbol{x})$ and its derivative (as well as the $g_m(\boldsymbol{x})$ and their derivatives in the case of a constrained problem)[2] are easy to compute, then relatively simple algorithms (e.g., gradient descent) are provably effective at performing the optimization.

The material in this course has three major components. The first is the mathematical foundations of convex optimization. We will see that talking about the solution to convex programs requires a beautiful combination of algebraic and geometric ideas.

The second component is algorithms for solving convex programs. We will talk about general purpose algorithms (and their associated computational guarantees), but we will also look at algorithms that are specialized to certain classes of problems, and even certain applications. Rather than focus exclusively on the "latest and greatest", we will try to understand the key ideas that are combined in different ways in many solvers.

[2] And as we will see, much of what we do can be naturally extended to non-smooth functions which do not have any derivatives.

Finally, we will talk a lot about *modeling*. That is, how convex optimization appears in signal processing, control systems, machine learning, statistical inference, etc. We will give many examples of mapping a word problem into an optimization program. These examples will be interleaved with the discussion of the first two components, and there are several examples which we may return to several times.

Convexity and Efficiency

Before going any further, there are two natural questions that you might have that we ought to explicitly address.

Can all convex programs be solved efficiently?

Unfortunately, no. There are many examples of even seemingly innocuous convex programs which are NP-hard. One way this can happen is if the objective function f and/or its derivative themselves are hard to compute. For example, consider the $(\infty, 1)$ norm:

$$f(\boldsymbol{X}) = \|\boldsymbol{X}\|_{\infty,1} = \max_{\|\boldsymbol{v}\|_\infty \leq 1} \|\boldsymbol{X}\boldsymbol{v}\|_1.$$

This is a valid matrix norm, and we will see later that all valid norms are convex. But it is known that computing f is NP-hard (see [Roh00]), as is approximating it to a fixed accuracy. Thus, optimization problems involving this quantity (as either the objective function or in the constraints) are bound to be difficult, despite being convex.

Are there any non-convex programs that can be solved efficiently?

Of course there are. Here is one for which you already know the answer:

$$\underset{\boldsymbol{x} \in \mathbb{R}^N}{\text{maximize}} \ \boldsymbol{x}^{\mathrm{T}} \boldsymbol{A} \boldsymbol{x} \ \text{ subject to } \ \|\boldsymbol{x}\|_2 = 1,$$

where $\boldsymbol{A}$ is an arbitrary $N \times N$ symmetric matrix. This is the *maximization* of an indefinite quadratic form (not necessarily convex or concave) over a nonconvex set. But we know that the optimal value of this program is the largest eigenvalue, and the optimizer is the corresponding eigenvector, and there are well-known practical algorithms for computing these.

When there is a solution to a nonconvex program, it often times relies on nice coincidences in the structure of the problem — perturbing the problem just a little bit can disturb these coincidences. Consider another nonconvex program that we know how to solve:

$$\underset{\boldsymbol{X}}{\text{minimize}} \ \sum_{i,j=1}^{N} (X_{i,j} - A_{i,j})^2 \ \text{ subject to } \ \text{rank}(\boldsymbol{X}) \leq R.$$

That is, we want the best rank-R approximation (in the least-squares sense) to the $N \times N$ matrix $\boldsymbol{A}$. The functional we are optimizing above is convex, but the rank constraint definitely is not. Nevertheless, we can compute the answer efficiently using the SVD of $\boldsymbol{A}$:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}} = \sum_{n=1}^{N} \sigma_n \boldsymbol{u}_n \boldsymbol{v}_n^{\mathrm{T}}, \quad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0.$$

The program above is solved simply by truncating this sum to its first R terms:

$$\widehat{\boldsymbol{X}} = \sum_{n=1}^{R} \sigma_n \boldsymbol{u}_n \boldsymbol{v}_n^{\mathrm{T}}.$$

But now suppose that instead of the matrix $\boldsymbol{A}$, we are given a subset of its entries indexed by $\mathcal{I}$. We now want to find the matrix that is most consistent over this subset while also having rank at most R:

$$\underset{\boldsymbol{X}}{\text{minimize}} \ \sum_{(i,j)\in\mathcal{I}} (X_{i,j} - A_{i,j})^2 \ \text{ subject to } \ \text{rank}(\boldsymbol{X}) \leq R.$$

Despite its similarity to the first problem above, this "matrix completion" problem is NP-hard in general.

Convex programs tend to be more robust to variations of this type. Things like adding subspace constraints, restricting variables to be

positive, and considering functionals of linear transforms of $\boldsymbol{x}$ all preserve the essential convex structure.

Note, however, that nonconvex problems can often still be solved in many cases. For instance, consider the "matrix completion" problem described above. Despite being NP-hard *in general*, there are important special cases where we can still solve this problem efficiently. One common trick for dealing with non-convex problems that works here and that we will see later in this course is *convex relaxation*. This approach replaces a non-convex constraint (e.g., the rank constraint above) with a (cleverly chosen) convex surrogate. In some cases (e.g., for a restricted class of matrices $\boldsymbol{A}$ above) one can show that the convex relaxation will have the same solution as the original non-convex problem.

Another approach to non-convex optimization, and one that is popular both in solving the matrix completion problem above as well as in training neural networks, is to simply ignore this non-convexity and to apply standard algorithms like gradient descent that, while derived with convex problems in mind, do not explicitly require convexity to be applied. While we lose the kinds of theoretical guarantees that we will derive for the convex case, these can still be effective tools in practice.

Next time we will continue our introduction to convex optimization by introducing a few of the very well-known classes of convex optimization programs and giving some example applications.

References

[Roh00] J. Rohn. Computing the norm $\|A\|_{\infty,1}$ is NP-Hard. *Linear and Multilinear Algebra*, 47:195–204, 2000.