Linear and Nonlinear Optimization

Lectures Notes - Fall '22

Samuel F. Potter

Last updated: September 8, 2022

Chapter 1

Review topics

In this chapter, I'll collect some topics which you should know for this course. I will add to this list as we progress through the course. For each topic, I will try to collect some links you can use to review. If you have a particularly good reference, please send it to me. Books, websites, videos, whatever—all are fine.

1.1 Single variable and multivariable calculus

- Derivatives (including how to take derivatives of common functions)
- Integrals
- Partial derivatives
- The gradient
- The Jacobian
- The Hessian
- The dot product
- Limits
- Sequences
- Taylor expansions in a single variable
- Taylor expansions in multiple variables
- "Big-O" notation
- Big-O remainder for a Taylor expansion
- Integral form of the remainder for a Taylor expansion

1.2 Linear algebra

- Matrix notation
- The matrix transpose
- The matrix inverse
- How to convert a linear system written as a set of equations to "matrix form"
- How to write a dot product in matrix notation
- How to compute a matrix vector product

1.3 Python programming

- The basics: syntax, how to call Python from the command-line, how to edit and debug your code, etc.
- How to install packages and use them in a Python script

Chapter 2

9/1/22

2.1 Course introduction

In this chapter, we want to briefly outline some of the basic ideas that you'll encounter in this course. The course title is "Linear and Nonlinear Optimization"—so, what optimization? One answer is that it's a combination of modeling, math, algorithms, numerical implementation, applications... the list goes on. If you haven't encountered it before, the most important of these to understand initially is modeling.

Well, what is modeling? It's an important part of the scientific enterprise, in which phenomena in the real world are reduced to mathematical objects, which are then validated against experiments or used to make predictions in the real world (possibly for business, engineering, government planning... again, the list goes on).

Modeling involves the interplay of the following pieces:

- **The application or problem:** the source of the phenomenon which is to be "captured" by the model.
- **The model:** the mathematical object to which that phenomenon has been reduced.
- **Analysis:** mathematical analysis of the model. What are its properties? How does it work? What are the solutions like? Are the simple instances of this model for which solutions can be derived, either directly or asymptotically?
- **Numerical methods:** algorithms for computing solutions to the model. Often, the model will be complicated, and it will not be possible to extract an analytical or asymptotic solution.

Example 2.1.1. Let's say we want to find the steady state temperature in a room, provided that we know (or can guess) the temperature or the thermal properties of each surface in the room. This is an application or problem that we could attempt to model. One such model is Laplace's equation, which is a partial differential equation. If we analyze the model, we can learn some things about how a characteristic solution might behave—for example,

solutions of Laplace's equation obey a maximum principle, which tells us something about the possible steady state temperature distributions. Furthermore, a course on PDEs might tell us how to compute the solution of Laplace's equation for some simple geometries and boundary conditions, such as for an idealized 1D model of a rod—but for a building, we will need to use a numerical method such as the finite element method (FEM).

This is a simple example from mechanical engineering with some practical use. In optimization, we will follow the same process, but our goal will be *to construct and solve models which describe a problem in which the goal is to minimize or maximize something.* This is extremely open-ended—in fact, Example 2.1.1 can be posed as an optimization problem if we consider the variational form of Laplace's equation!

Example 2.1.2. Say we're interested in driving from Chicago to Los Angeles as quickly as possible. A simple model for this would be to make a graph consisting of the interstates connecting Chicago and Los Angeles, with edges being weighted by the speed limit of each interstate. In this graph, a *path* connecting Chicago to Los Angeles is a sequence of edges in the graph starting at Chicago and leading to Los Angeles. The *cost* of the path is the sum of the weights on the edges. We could then attempt to find the *shortest path* (the minimum cost path) from Chicago to Los Angeles. If we think about this for a bit (i.e., analyze), we'll realize the edge weights being nonnegative means that it is even possible to find a shortest path. Later in these notes, we might see that Dijkstra's algorithm is an efficient algorithm for finding a shortest path.

This is an example of a particular kind of optimization problem called a *combinatorial* or *discrete optimization problem.* We will learn bits and pieces about combinatorial optimization problems in passing in this case, but they will not be our main focus. Instead, we will focus on *continuous optimization problems.*

Example 2.1.3. Imagine we have a *convex* polygon P in $\mathbb{R}^2$, where a *convex set* is a set where, if we consider any pair of points inside, the closed interval connecting them is fully contained in the set. Now: what is the largest area ellipsoidal ball E which fits inside this polygon?

How could we model this problem? Something like:

$$\begin{aligned} \text{maximize} \quad & \mathsf{Area}(E) \\ \text{subject to} \quad & E \subseteq P. \end{aligned} \tag{2.1}$$

This is an example of a continuous optimization problem. The variable we're optimizing over, E, is a continuous variable, since E can be parametrized by a vector—perhaps, without loss of generality, we might just write:

$$E = \left\{ (x, y) \in \mathbb{R}^2 : \frac{x^2}{a^2} + \frac{y^2}{b^2} \leq 1 \right\}. \tag{2.2}$$

Hence, there should be a formula for Area in terms of the pair (a, b). However, it will turn out that solving this problem requires some techniques which we won't come to until later in the course. (If you're interested, this ellipse goes by the name of the Löwner-John ellipse.)

In general, we can see that a continuous optimization problem will take the form:

$$\begin{aligned} \text{minimize} \quad & F(X) \\ \text{subject to} \quad & X \in \mathcal{X}, \end{aligned} \tag{2.3}$$

where F is the *cost function*, and $\mathcal{X}$ is the *domain of the optimization problem* (or *constraint set*). Note that since:

$$\min_{X \in \mathcal{X}} F(X) = -\max_{X \in \mathcal{X}} -F(X), \tag{2.4}$$

so we will usually just talk about minimization problems when we want to discuss optimization problems in the abstract.

Note that Equation (2.3) is very abstract, and there are many ways of making it concrete. We have seen two examples a optimization problems which are encapsulated by Equation (2.3) in Examples 2.1.2 and 2.1.3. We will focus on three major families of optimization problems in this course:

1. *Linear programs*
2. *Unconstrained nonlinear programs*
3. *Constrained nonlinear programs*

Usually, the latter two are just referred to as "unconstrained optimization problems" and "constrained optimization problems", respectively. The difference between these three families resides in how we specify the cost function F and the domain X.

Note that the fact that these are all classes of continuous optimization problems means that we will assume that $X \subseteq \mathbb{R}^n$—that is, we optimize a scalar field $F : X \to \mathbb{R}$ over a subset of the finite-dimensional real vector space $\mathbb{R}^n$ (we may further qualify this by saying that we consider only *finite-dimensional optimization problems*, since it is also possible to consider infinite-dimensional ones).

Another consideration is whether these optimization problems are in fact "computable". The assumption that X is a subset of $\mathbb{R}^n$ is not that restrictive! That are some horrendous subsets of $\mathbb{R}^n$ which would be impossible to efficiently describe on a computer (for instance, consider optimizing a function over a fractal—maybe interesting to try!). To that end, we assume that X can be described by some finite list of *equality* and *inequality constraints*:

$$X = \left\{ x \in \mathbb{R}^n : g_i(x) = 0 \text{ for } i \in I \text{ and } h_i(x) \leq 0 \text{ for } j \in J \right\}, \tag{2.5}$$

where $I, J \subseteq \mathbb{Z}$ are index sets such that $|I| < \infty$ and $|J| < \infty$.

Along the way, we will also encounter some examples of combinatorial optimization problems, as we mentioned, as well as *dynamic programming*. An important theme throughout all of this will be *convexity*—not only of sets, but also of functions.

2.2 What is linear programming?

A linear program (or *LP*) is an instance of Equation (2.3) where $F : X \to \mathbb{R}$ is a *linear function* and each of the constraint functions g_i and h_j are linear, as well. Recall that a linear function f mapping from $\mathbb{R}^n$ to $\mathbb{R}$ is very simple. Let $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$. Then we can write it as:

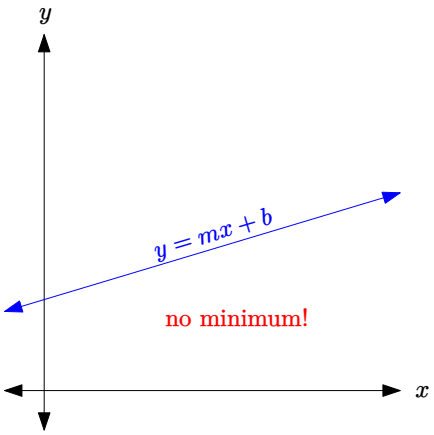
$$f(x) = a^\top x + b = \sum_{i=1}^{n} a_i x_i + b, \tag{2.6}$$

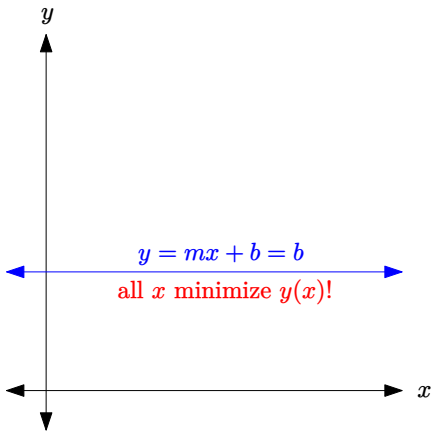
where "$^\top$" denotes the matrix transpose. Hence, a linear program will have the general form:

$$\begin{array}{lll} \text{minimize} & a^\top x & \\ \text{subject to} & c_i^\top x + d_i = 0, & i \in I, \\ & e_j^\top x + f_j \leq 0, & j \in J. \end{array} \tag{2.7}$$

Note that we have left off the "b" in the cost function, since it does not affect the location of the minimizer!

Let's consider a few simple examples of LPs where $n = 1$ (i.e., $x \; in \mathbb{R}$—just a scalar optimization variable). In this case, the cost function takes a simple and familiar form: $y = F(x) = mx + b$. Here, m is the slope and b is the intercept of a line. There are two cases to check: $m \neq 0$ and $m = 0$:

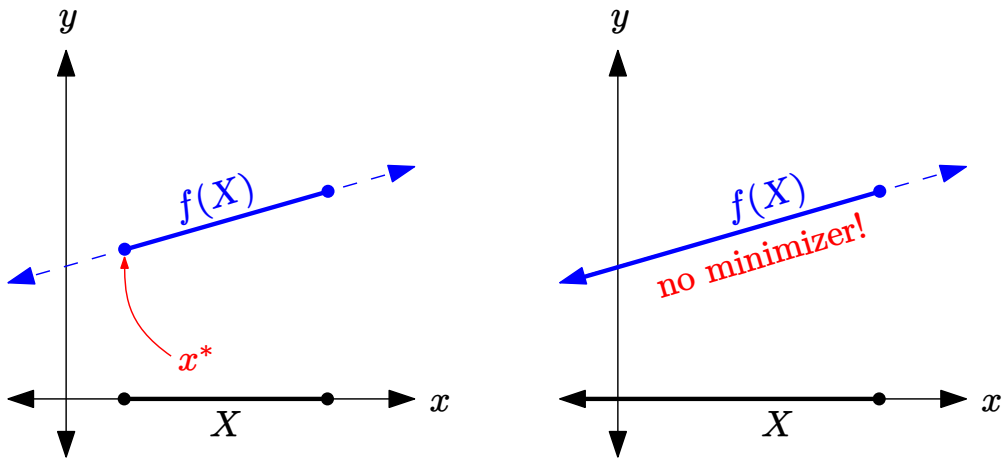




If $m \neq 0$, it is clear that $y = mx + b$ is unbounded below, and there is no minimizer for finite x. On the other hand, if $m = 0$, then since $y(x) = b$ is just the constant function, every point x minimizes $y(x)$.

Problem 2.2.1. Explain why this is actually the same behavior we should expect for an unconstrained LP in any number of dimensions (i.e., where $X = \mathbb{R}^n$, $n \geq 1$).

Now let's consider a constrained LP in 1D. Again, there are two basic important cases to check:



Here, we write the *optimum* (the point which attains the minimum or maximum of an optimization problem) of the minimization problem as:

$$x^* = \arg\min_{x \in X} F(x). \tag{2.8}$$

This is standard notation for the "argmin" or *minimizing argument* of a minimization problem. At this point, based on Problem 2.2.1, we should ask what behavior to expect in higher dimensions. In general, the geometry of high-dimensional spaces is quite complicated and unintuitive. We are not yet in a good position to predict what will happen if $n > 1$.

Note: More to come!

Chapter 3

9/6/22

Chapter 4

9/8/22