Table of Contents

The following notes use the color scheme:

Definition

Theorem

Fact

Example

Method/step-by-step recipe

Proof

Super-duper important things

Motivation

Convex optimization can solve a broad variety of practical problems *reliably* and *quickly*. These problems include:

- Portfolio optimization
- Device sizing (e.g. of electronic circuits)
- Data fitting
- Rocket landing
- Optimal energy control in hybrid/electric road transport

Recently, there has been a rapid growth in *embedded optimization*. In these applications, optimization must make *real-time choices* and require *little to now human intervention*. This requires proofs that an optimal solution will be found, and that it will be founded in bounded time. This is where convex optimization (and its subsets linear and least-squares optimizations) shine.

Introduction to Optimizations

Definition of an optimization problem

A *mathematical optimization problem* has the form

$$\begin{aligned} &\text{minimize} && f_0(x) \\ &\text{subject to} && f_i(x) \leqslant b_i, \quad i = 1, \ldots, m \end{aligned} \tag{1}$$

where:

- $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ is the *optimization variable*;
- $f_0 : \mathbb{R}^n \to \mathbb{R}$ is the *objective function*;
- $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$ are the *constraint functions*;
- $b_1, \ldots, b_m$ are the limits (bounds) for the constraints.

A vector x^* is *optimal* or is a *solution* if

$$\forall z \in \mathbb{R}^n \text{ satisfying the constraints}, \quad f_0(x^*) \leqslant f_0(z).$$

Classes of optimization problems

Optimization problems are bundled into classes based on the form of their objective (f_0) and constraint (f_i) functions. Here are some below.

Note: in optimization jargon, a "program" is an alias for an "optimization problem".

Linear program

(1) is a **linear program** if $f_0, \ldots, f_m$ are linear functions (i.e. both objective and constraints).

Recall that:
$f(x) : \mathbb{R}^n \to \mathbb{R}$ is linear $\Leftrightarrow \forall x, y \in \mathbb{R}^n,\ \forall \alpha \in \quad f(\alpha x + \beta y) = \alpha f(x) + \beta f(y)$.

Nonlinear program

An optimization problem is nonlinear if it is not linear. Of course, there are many classes of nonlinear programs. A general nonlinear program can be applied to basically any optimization problem, however:

- It may take a very long time to solve;
- It may not find the (global) solution;
- It may be very difficult to solve!

Convex program

...and convex optimization is one of them!

A **convex optimization problem** is one in which $f_0, \ldots, f_m$ are convex:
$f(x) : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if
$\forall x, y \in \mathbb{R}^n, \forall \alpha, \beta \in \mathbb{R}_+ \text{ s.t. } \alpha + \beta = 1 \quad \Rightarrow \quad f(\alpha x + \beta y) \leqslant \alpha f(x) + \beta f(y)$.

Unlike general nonlinear programs, convex (and linear) programs can be solved *efficiently* and *reliably*, making them ideally suited for real world (and even high stakes) applications!

In fact, a convex program is a common parent of two well know, reliable and efficient optimization classes: least squares and linear programs.

Least-Squares Problems

A **least-squares problem** is an optimization problem with no constraints and an objective function which is affine in the optimization variable x. Let $A \in \mathbb{R}^{k \times n}$ (with $k \geqslant n$) and a_i^T be its rows. Then the optimization problem is:

$$\text{minimize} f_0(x) = \|Ax - b\|_2^2 = \sum_{i=1}^{k} (a_i^T x - b_i)^2$$

Least-squares programming is a *mature technology*: least-squares problems can be solved efficiently and reliably. Many software packages exist that do this.

Least-squares properties:

- The analytical solution for $A \in \mathbb{R}^{k \times n}$ tall (i.e. $k > n$) and full rank (i.e. full column rank, $\text{null}(A) = \varnothing$): $x^* = (A^T A)^{-1} A^T b$;
  - NB: good solvers don't do literally this operation, though!
- Compute time $\propto n^2 k$.
  - Where n is "*small*" while k is "*large*". Just remember this: computation time is "small squared times large".

How to know if your problem is a least-squares problem? **Simple, just one question**:

- Q: is the objective function the 2-norm squared of an affine function of x (and you have no constraints)?
  - A: Yes: it's a least-squares problem!
  - A: No: it's not a least squares problem!

Applications to least-squares programming include:

- Regression analysis
- Parameter estimation
- Data fitting methods
- Optimal control

A variation of standard least squares is *weighted least squares* with weights $w_i \geqslant 0\ \forall i$ reflecting the relative concern for term $a_i^T x - b_i$ being large:

$$\text{minimize} \sum_{i=1}^{k} w_i (a_i^T x - b_i)^2$$

Another technique in least squares is *regularization* which adds extra terms for the cost function, e.g.

$$f_0(x) = \sum_{i=1}^{k} (a_i^T x - b_i)^2 + \rho \sum_{i=1}^{k} x_i^2$$

where $f_0(x)$ can still be formulated as a least squares problem. The extra terms penalize large values of x and result in a sensible solution in cases where minimizing the first sum only does not. Parameter $\rho \in [0, 1]$ trades off between meeting the original objective ($\rho = 0$) and keeping x_i^2 small ($\rho = 1$).

Regularization is used in statistical estimation when x is given a prior distribution.

Linear Programming

$$\begin{aligned} &\text{minimize} && c^T x \\ &\text{subject to} && a_i^T x \leqslant b_i, \quad i = 1, \ldots, m \end{aligned}$$

Linear programming is, like least squares, a mature technology. Existing algorithms are (almost) as reliable as least squares.

Some linear programming properties:

- No analytical solution (except for trivial cases)!
- Computation time $\propto n^2 m$ if $m \geqslant n$ (more constraints than optimization variables);
  - Less with structure;
  - NB: this is the same computational time as least-squares!
  - It is good news that it's $n^2 m$ and not $m^2 n$ (since generally there can be very many constraints on a relatively small optimization variable)