Optimization and Linear Programming

George McNinch

2024-01-29

Section 1

Multivariable optimization with constraints

Formulation

So far we have been looking at various optimization problems, but only of very specific types. Optimization is clearly important and so we'd like to describe some general strategy to help us tackle more problems.

So let's describe what is more-or-less the most general form of an optimization problem:

Consider an $\mathbb{R}$-valued function f defined for $\mathbf{x} \in \mathbb{R}^n$ – thus, $f : \mathbb{R}^n \to \mathbb{R}$. We want to optimize $f(\mathbf{x})$ subject to a system of *constraints* defined by some auxiliary data.

We first consider E constraints defined for $1 \leq i \leq E$ by functions $g_i : \mathbb{R}^n \to \mathbb{R}$ together with values $b_i \in \mathbb{R}$; these constraints have the form

$$(\heartsuit)_i \quad g_i(\mathbf{x}) \leq b_i$$

At the same time, we consider F constraints defined for $1 \leq j \leq F$

Compactly, a *general optimization problem* asks to find the maximum (or minimum) of $f : \mathbb{R}^n \to \mathbb{R}$ for $\mathbf{x} \in \mathbb{R}^n$ subject to constraints

$$g_i(\mathbf{x}) \le b_i \quad 1 \le i \le E$$

$$h_j(\mathbf{x}) = c_j \quad 1 \le j \le F$$

For functions $g_i, h_j : \mathbb{R}^n \to \mathbb{R}$ and scalars b_i, c_j. :::

Remarks

- One might wonder why we don't consider constraints of the form

$$\ell(\mathbf{x}) < d \quad \text{or} \quad \ell(\mathbf{x}) \geq d \quad \text{or} \quad \ell(\mathbf{x}) > d$$

The answer is that the conditions imposed by constraints of these form can be achieved by using (possibly more) constraints of the forms $(\heartsuit)_i$ or $(\clubsuit)_j$.

Remarks

- One might wonder why we don't consider constraints of the form

$$\ell(\mathbf{x}) < d \quad \text{or} \quad \ell(\mathbf{x}) \geq d \quad \text{or} \quad \ell(\mathbf{x}) > d$$

The answer is that the conditions imposed by constraints of these form can be achieved by using (possibly more) constraints of the forms $(\heartsuit)_i$ or $(\clubsuit)_j$.

- Our older examples all (essentially) have this form:

Remarks

- One might wonder why we don't consider constraints of the form

$$\ell(\mathbf{x}) < d \quad \text{or} \quad \ell(\mathbf{x}) \geq d \quad \text{or} \quad \ell(\mathbf{x}) > d$$

The answer is that the conditions imposed by constraints of these form can be achieved by using (possibly more) constraints of the forms $(\heartsuit)_i$ or $(\clubsuit)_j$.

- Our older examples all (essentially) have this form:
  - For example, single-variable optimization amounts to the case $n = 1$ - i.e. f is a function $\mathbb{R} \to \mathbb{R}$. Typically we optimize on an interval – for example we might want to optimize f for x in the closed interval $[a, b]$. Then one of the constraints has the form $x \leq b$ (so this constraint has the form $(\heartsuit)$, g_1 is just the identity function, and $b_1 = b$) and the constraint $a \leq x$ also has the form $(\heartsuit)$, $g_2(x) = -x$, and $b_2 = -a$).

Remarks

- One might wonder why we don't consider constraints of the form
$$\ell(\mathbf{x}) < d \quad \text{or} \quad \ell(\mathbf{x}) \geq d \quad \text{or} \quad \ell(\mathbf{x}) > d$$

The answer is that the conditions imposed by constraints of these form can be achieved by using (possibly more) constraints of the forms $(\heartsuit)_i$ or $(\clubsuit)_j$.

- Our older examples all (essentially) have this form:
    - For example, single-variable optimization amounts to the case $n = 1$ - i.e. f is a function $\mathbb{R} \to \mathbb{R}$. Typically we optimize on an interval – for example we might want to optimize f for x in the closed interval $[a, b]$. Then one of the constraints has the form $x \leq b$ (so this constraint has the form $(\heartsuit)$, g_1 is just the identity function, and $b_1 = b$) and the constraint $a \leq x$ also has the form $(\heartsuit)$, $g_2(x) = -x$, and $b_2 = -a$).

A “linear” example: Carpenter; tables; shelves

This example is an instance of a problem in “linear programming”, as we'll describe below.

> *A carpenter can choose to make either tables or bookshelves. She makes a profit of \$25 per constructed table and \$30 per constructed bookshelf (demand is sufficient that all constructed products will be sold).*
> *It takes 5 hours of labor and 20 board-feet of lumber to make a table and 4 hours of labor and 30 board-feet of lumber to make a bookshelf. If she has access to 120 hours of labor and 690 board-feet of lumber each week, how many tables and how many bookshelves should she make to maximize profit?*

What are the variables?

- $t = \#$ of tables made per week

A "linear" example: Carpenter; tables; shelves

This example is an instance of a problem in "linear programming", as we'll describe below.

> *A carpenter can choose to make either tables or bookshelves. She makes a profit of \$25 per constructed table and \$30 per constructed bookshelf (demand is sufficient that all constructed products will be sold).*
> *It takes 5 hours of labor and 20 board-feet of lumber to make a table and 4 hours of labor and 30 board-feet of lumber to make a bookshelf. If she has access to 120 hours of labor and 690 board-feet of lumber each week, how many tables and how many bookshelves should she make to maximize profit?*

What are the variables?

- $t = \#$ of tables made per week
- $b = \#$ of bookshelves made per week

A "linear" example: Carpenter; tables; shelves

This example is an instance of a problem in "linear programming", as we'll describe below.

> *A carpenter can choose to make either tables or bookshelves. She makes a profit of \$25 per constructed table and \$30 per constructed bookshelf (demand is sufficient that all constructed products will be sold).*
> *It takes 5 hours of labor and 20 board-feet of lumber to make a table and 4 hours of labor and 30 board-feet of lumber to make a bookshelf. If she has access to 120 hours of labor and 690 board-feet of lumber each week, how many tables and how many bookshelves should she make to maximize profit?*

What are the variables?

- t = # of tables made per week
- b = # of bookshelves made per week
- p = profit per week

A “linear” example: Carpenter; tables; shelves

This example is an instance of a problem in “linear programming”, as we’ll describe below.

> *A carpenter can choose to make either tables or bookshelves. She makes a profit of \$25 per constructed table and \$30 per constructed bookshelf (demand is sufficient that all constructed products will be sold).*
> *It takes 5 hours of labor and 20 board-feet of lumber to make a table and 4 hours of labor and 30 board-feet of lumber to make a bookshelf. If she has access to 120 hours of labor and 690 board-feet of lumber each week, how many tables and how many bookshelves should she make to maximize profit?*

What are the variables?

- $t = \#$ of tables made per week
- $b = \#$ of bookshelves made per week
- $p =$ profit per week

A “linear” example: Carpenter; tables; shelves

This example is an instance of a problem in “linear programming”, as we’ll describe below.

> *A carpenter can choose to make either tables or bookshelves. She makes a profit of \$25 per constructed table and \$30 per constructed bookshelf (demand is sufficient that all constructed products will be sold).*
> *It takes 5 hours of labor and 20 board-feet of lumber to make a table and 4 hours of labor and 30 board-feet of lumber to make a bookshelf. If she has access to 120 hours of labor and 690 board-feet of lumber each week, how many tables and how many bookshelves should she make to maximize profit?*

What are the variables?

- $t = \#$ of tables made per week
- $b = \#$ of bookshelves made per week
- $p =$ profit per week

A "linear" example: Carpenter; tables; shelves

This example is an instance of a problem in "linear programming", as we'll describe below.

> *A carpenter can choose to make either tables or bookshelves. She makes a profit of \\$25 per constructed table and \\$30 per constructed bookshelf (demand is sufficient that all constructed products will be sold).*
> *It takes 5 hours of labor and 20 board-feet of lumber to make a table and 4 hours of labor and 30 board-feet of lumber to make a bookshelf. If she has access to 120 hours of labor and 690 board-feet of lumber each week, how many tables and how many bookshelves should she make to maximize profit?*

What are the variables?

- $t = \#$ of tables made per week
- $b = \#$ of bookshelves made per week
- $p =$ profit per week

A “linear” example: Carpenter; tables; shelves

This example is an instance of a problem in “linear programming”, as we’ll describe below.

> *A carpenter can choose to make either tables or bookshelves. She makes a profit of \$25 per constructed table and \$30 per constructed bookshelf (demand is sufficient that all constructed products will be sold).*
> *It takes 5 hours of labor and 20 board-feet of lumber to make a table and 4 hours of labor and 30 board-feet of lumber to make a bookshelf. If she has access to 120 hours of labor and 690 board-feet of lumber each week, how many tables and how many bookshelves should she make to maximize profit?*

What are the variables?

- $t = \#$ of tables made per week
- $b = \#$ of bookshelves made per week
- $p =$ profit per week

A "linear" example: Carpenter; tables; shelves

This example is an instance of a problem in "linear programming", as we'll describe below.

> *A carpenter can choose to make either tables or bookshelves. She makes a profit of \$25 per constructed table and \$30 per constructed bookshelf (demand is sufficient that all constructed products will be sold).*
> *It takes 5 hours of labor and 20 board-feet of lumber to make a table and 4 hours of labor and 30 board-feet of lumber to make a bookshelf. If she has access to 120 hours of labor and 690 board-feet of lumber each week, how many tables and how many bookshelves should she make to maximize profit?*

What are the variables?

- $t = \#$ of tables made per week
- $b = \#$ of bookshelves made per week
- $p =$ profit per week

Linear Programming

The term *linear programming* refers to optimization problems in which the function to be optimized, as well as all of the constraint equations, are *linear* functions of the variables.

The strategy used to find the optimal value in the carepentry example was pretty good! It works well if we only have a few constraints and a few variables. But if we have **many variables and many constraints**, we find **a lot** of vertices in high dimensions.

For example, let's assume given a linear function of 50 variables, and 150 linear constraints (including the conditions that all 50 variables are non-negative). With 50 variables, we expect a point to be specified by exactly 50 linear equations. So we expect a point of intersection of our boundary equations to be determined by selecting 50 of the 150 possible equations. The number of possible ways of choosing 50 items from 150 possible items is the binomial

Linear programming – some preliminaries

Let's take a moment and describe linear programming problems using notation from *linear algebra*. If there are n variables x_i, we write $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n$ for the corresponding "variable vector".

More generally, we denote by $\mathbb{R}^{m\times n}$ the space of $m \times n$ matrices – i.e. matrices with m rows and n columns; thus

$$\mathbb{R}^{m\times n} = \left\{ \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \;\middle|\; a_{ij} \in \mathbb{R} \right\}$$

Now, a linear function $\mathbb{R}^n \to \mathbb{R}$ is given by a $1 \times n$ matrix – i.e. a row vector

Notations

We now pause to fix some *notation:*

Suppose that the inequality constraints are determined by linear functions corresponding to vectors

$$\mathbf{a}_1 = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \end{bmatrix}, \mathbf{a}_2, \cdots, \mathbf{a}_r \in \mathbb{R}^{1\times n}$$

and scalars b_i for $1 \le i \le r$.

The i-th inequality constraint requires that

$$\mathbf{a}_i \cdot \mathbf{x} \le b_i$$

.

Now form the $r \times n$ matrix A whose rows are given by the vectors $\mathbf{a}_i$:

$$A = \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \end{pmatrix}$$

Standard Form

Recapitulating, a **linear programming problem** is determined by the number n of variables, the choice of vectors $\mathbf{c}, \mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_r \in \mathbb{R}^{1\times n}$ and the choice of scalars $b_1, \ldots, b_r$.

The goal is to maximize $\mathbf{c} \cdot \mathbf{x}$ subject to the constraint

$$\mathbf{A} \cdot \mathbf{x} \leq \mathbf{b}$$

where $A = \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_r \end{pmatrix}$ is the $r \times n$ matrix whose rows are the row-vectors $\mathbf{a}_i$ and $\mathbf{b} \in \mathbb{R}^r$ has entries b_i.

We say that the linear programming problem is posed in standard form if it has this form.

- **Remark**: if $\mathbf{a} \in \mathbb{R}^{1\times n}$ and $b \in \mathbb{R}$, an inequality constraint of the form

Why do we impose no equality constraints??

Consider a *linear programming problem* as above, but suppose also that we imposed *equality constraints* determined by vectors

$$\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_s$$

in $\mathbb{R}^{1\times n}$ and the scalar values

$$\gamma_1, \gamma_2, \cdots, \gamma_s$$

In other words, the ith equality constraint requires that

$$(\clubsuit) \quad \mathbf{b}_i \cdot \mathbf{x} = \gamma_i$$

Now form the $s \times n$ matrix B whose rows are the $\mathbf{b}_i$:

$$B = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \end{pmatrix}$$

Why do we impose no equality constraints??

Consider a *linear programming problem* as above, but suppose also that we imposed *equality constraints* determined by vectors

$$\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_s$$

in $\mathbb{R}^{1\times n}$ and the scalar values

$$\gamma_1, \gamma_2, \cdots, \gamma_s$$

In other words, the ith equality constraint requires that

$$(\clubsuit) \quad \mathbf{b}_i \cdot \mathbf{x} = \gamma_i$$

Now form the $s \times n$ matrix B whose rows are the $\mathbf{b}_i$:

$$B = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \end{pmatrix}$$

Example: eliminating equality constraints

A simple example should illustrate the idea just described

Consider the linear program in 2 variables x, y which seeks to minimize the value of the function given by $\mathbf{c} = \begin{bmatrix} c_1 & c_2 \end{bmatrix}$ subject to the constraints:

- $\mathbf{0} \le \mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$

and

for some scalar quantity $\lambda > 0$.

Of course, the equality constraint $\begin{bmatrix} -\lambda & 1 \end{bmatrix} \cdot \mathbf{x} = 1$ just says that $y = \lambda x + 1$ - i.e. the point (x, y) must lie on the line with slope λ and y intercept 1.

Example: eliminating equality constraints

A simple example should illustrate the idea just described

Consider the linear program in 2 variables x, y which seeks to minimize the value of the function given by $\mathbf{c} = \begin{bmatrix} c_1 & c_2 \end{bmatrix}$ subject to the constraints:

- $\mathbf{0} \leq \mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$
- $\begin{bmatrix} 1 & 1 \end{bmatrix} \cdot \mathbf{x} \leq 3$

and

for some scalar quantity $\lambda > 0$.

Of course, the equality constraint $\begin{bmatrix} -\lambda & 1 \end{bmatrix} \cdot \mathbf{x} = 1$ just says that $y = \lambda x + 1$ - i.e. the point (x, y) must lie on the line with slope λ and y intercept 1.

Example: eliminating equality constraints

A simple example should illustrate the idea just described

Consider the linear program in 2 variables x, y which seeks to minimize the value of the function given by $\mathbf{c} = \begin{bmatrix} c_1 & c_2 \end{bmatrix}$ subject to the constraints:

- $\mathbf{0} \leq \mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$
- $\begin{bmatrix} 1 & 1 \end{bmatrix} \cdot \mathbf{x} \leq 3$

and

- $\begin{bmatrix} -\lambda & 1 \end{bmatrix} \cdot \mathbf{x} = 1$

for some scalar quantity $\lambda > 0$.

Of course, the equality constraint $\begin{bmatrix} -\lambda & 1 \end{bmatrix} \cdot \mathbf{x} = 1$ just says that $y = \lambda x + 1$ - i.e. the point (x, y) must lie on the line with slope λ and y intercept 1.

Remarks about linear programming

History

The idea arose during World War II to reduce costs for the military. It was first developed in 1939 by Leonid Kantorovich, a Russian mathematician and economist. In the 1970s, he won the Nobel Prize in Economics for his "contributions to the theory of optimum allocation of resources."
For more information, see this historical discussion. Significant contributions include the simplex method, invented by George Dantzig in the late 1940s.

Applications

Linear programming problems arise naturally in many settings: - minimal staffing needed to complete scheduled tasks - maximizing profit & minimizing costs when considering multiple options - minimizing risk of investment subject to achieving a return - minimizing transport costs

Using scipy to solve linear programs

Before we discuss how

The `scipy` library (more precisely, the `scipy.optimize` library) provides a `python` function which implements various algorithms for solving linear programs.

The `API` interface of this function can be found here:

docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html

Here is a minimalist sketch.

The function call

`linprog(c,A_ub=A,b_ub=v)`

for a 1 dimensional "row vector" c of dimension n minimizes the linear objective function

$$\mathbf{x} \mapsto \mathbf{c} \cdot \mathbf{x}$$

subject to constraint

Some examples

Here is the solution to the *carpenter example* obtained via `scipy`:

Higher dimensional example

Here is an example with more variables:

Maximize the value of the linear function given by

$$\mathbf{c} = \begin{bmatrix} 5 & 4 & 3 \end{bmatrix}$$

and with inequality constraints determined by

$$\begin{bmatrix} 2 & 3 & 1 \end{bmatrix} \cdot \mathbf{x} \leq 5$$
$$\begin{bmatrix} 4 & 1 & 2 \end{bmatrix} \cdot \mathbf{x} \leq 11$$
$$\begin{bmatrix} 3 & 4 & 2 \end{bmatrix} \cdot \mathbf{x} \leq 8$$
$$\mathbf{0} \leq \mathbf{x}$$

where

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Note that the constraint