Optimisation and Operations Research

Lecture 2: Revision

Matthew Roughan
<matthew.roughan@adelaide.edu.au>
http://www.maths.adelaide.edu.au/matthew.roughan/notes/OORII/

School of Mathematical Sciences,
University of Adelaide

July 30, 2019

Section 1

Optimisation Revision

Optimisation

Optimisation is a task all human beings, indeed all living things, do. It is central to any *decision making* task, *i.e.*, in any task involving *choosing* between *alternatives*. Choice is governed by wanting to make the "best" decision: *e.g.*,

- minimise the cost of producing a widget;
- shortest route to Hungry Jacks, or the bar; or
- getting greatest exam mark, given a limited amount of study time.

All involve looking for the best solution to some objective often subject to some constraints.

We can even think of natural processes such as evolution as a form of optimisation, and indeed *genetic algorithms* and *evolutionary computing* deliberately exploit this metaphor to solve other optimisation problems we may wish to solve.

Notation and Conventions

Throughout these notes we will try to use consistent notation.

- lower-case letters, *e.g.*, x, will generally denote scalars
- boldface letters, *e.g.*, $\mathbf{x}$, will denote (column) vectors, *i.e.*,

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

  and we say $\mathbf{x} \in \mathbb{R}^n$
- upper case letters, *e.g.*, A, will denote matrices.

There may be exceptions, but we will try to make them clear.

Optimising a real function

You probably remember that when maximising (or minimising) a real-valued function

$$\max f(x), \quad x \in \mathbb{R},$$

we look for x such that the derivative is zero

$$\frac{df(x)}{dx} = 0.$$

- we can include more than one variable (set partials to zero)
- we can include constraints (through Lagrange multipliers)

but the problems we consider in this course don't fit this pattern.

A Standard Linear Programming Problem

Linear programming:

$$\begin{array}{rrcl} \max & z & = & \mathbf{c}^T\mathbf{x} + z_0 \\ \text{such that} & A\mathbf{x} & \leq & \mathbf{b} \\ \text{and} & \mathbf{x} & \geq & 0 \end{array}$$

Note the use of *program* here. It doesn't mean a computer program[1], it is using the older sense of "a plan, schedule, or procedure." Often the goal of one of our programs is to determine a schedule (its one of the oldest mathematically treated optimisation problems).

[1] Incidentally, the term *programming* was used for optimisation at least in 1948 by Dantzig [WD49, Dan49], well before it was used for computer programming. For a nice, quick history of optimisation see [Leo08].

Requirements

We'll be looking at a different class of problems from $\max f(x), x \in \mathbb{R}$.

- Constraints are a *very* important part
- The objective function is linear, so $f' \neq 0$
- Variables might be restricted to be integers
- The dimension of problem might be very high (1000s)

We will consider new techniques which allow us to address some of these challenges.

What is the background we need to know to get started?

We'll be looking at Linear Programs.

What do we need to know?

- Constraints define a region
  - we should understand the shape of that region, and the affect it has on the solution
  - the constraints are linear inequalities (or equalities), so what sort of space do they define?
- The objective function is also linear

We need to remember some of the linear algebra you did in first year.

Section 2

The Geometry of Linear Programming

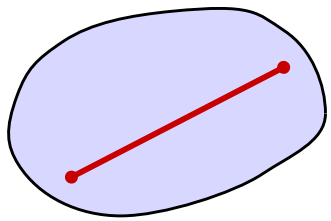
Convex Sets

Definition (A convex set)

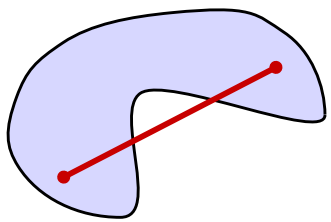
A set $S \subseteq \mathbb{R}^n$ is called a *convex set* if for all $x, y \in S$ and $0 \leq \alpha \leq 1$

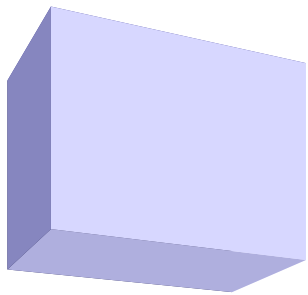
$$x, y \in S \implies \alpha x + (1 - \alpha) y \in S,$$

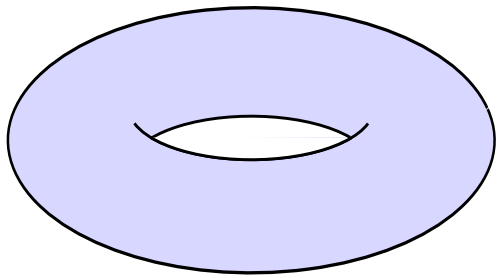
i.e., the line segment joining x and y lies in S.



Convex







Non-convex

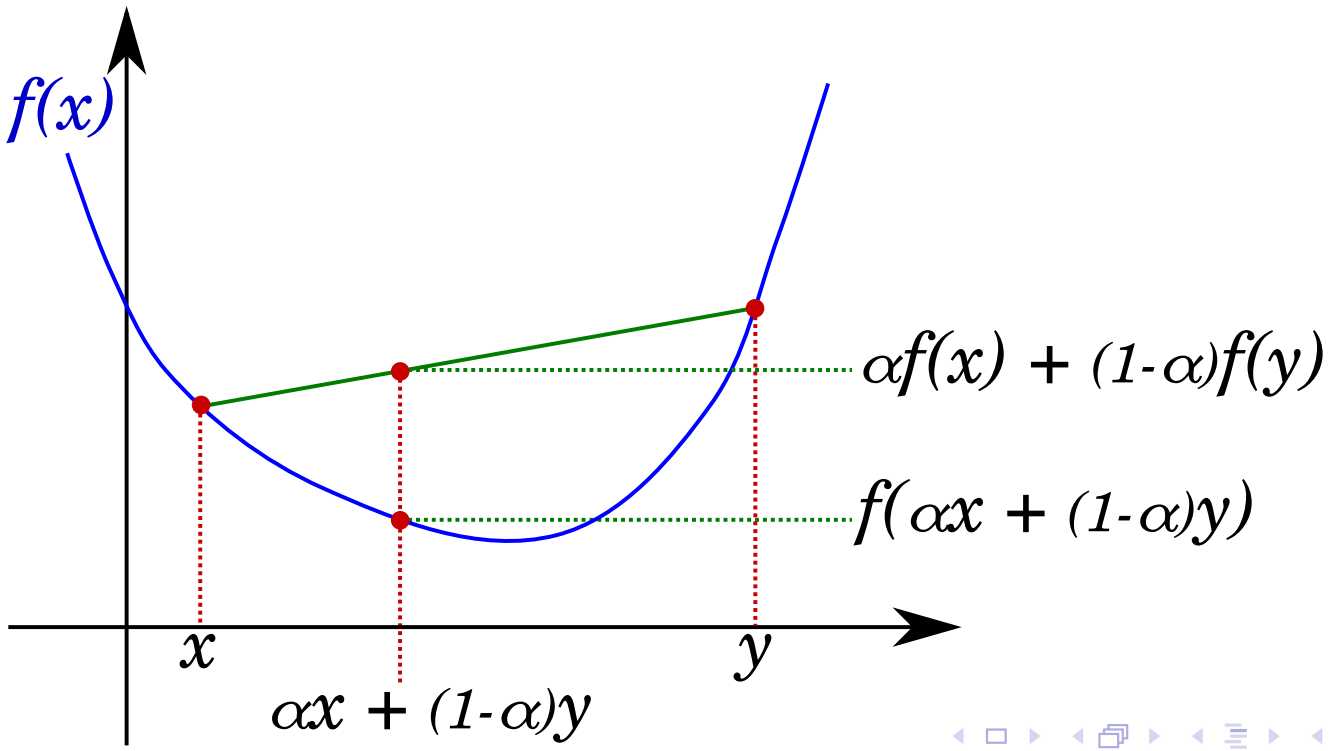
Convex Functions

Definition (A convex function)

A function $f : S \to \mathbb{R}$ is a *convex function* if for all $x, y \in S$ and $0 \leq \alpha \leq 1$

$$\alpha f(x) + (1 - \alpha) f(y) \geq f\big(\alpha x + (1 - \alpha) y\big)$$

i.e., chords don't lie below the function.



Convex Sets

Lemma

Given m convex functions $g_i(\mathbf{x})$, the set Ω defined by

$$\Omega = \left\{ \mathbf{x} \in \mathbb{R}^n \,\middle|\, g_i(\mathbf{x}) \leq 0, \ \text{for } i = 1, 2, \ldots, m \right\},$$

is convex.

Convex Sets

Proof

Start by considering a *single* constraint.

Take the set

$$\Omega_i = \left\{ \mathbf{x} \in \mathbb{R}^n \middle| g_i(\mathbf{x}) \leq 0 \right\}.$$

Take two points in this set $\mathbf{x}$ and $\mathbf{y}$, then consider a point on the chord between them, $\mathbf{z} = \alpha\mathbf{x} + (1-\alpha)\mathbf{y}$, for $\alpha \in [0,1]$. As $g_i(\cdot)$ is convex,

$$g_i(\mathbf{z}) = g_i\big(\alpha\mathbf{x} + (1-\alpha)\mathbf{y}\big) \leq \alpha g_i(\mathbf{x}) + (1-\alpha) g_i(\mathbf{y}),$$

Now $g_i(\mathbf{x}) \leq 0$ and $g_i(\mathbf{y}) \leq 0$ because $\mathbf{x}, \mathbf{y} \in \Omega_i$ and α and $1-\alpha \geq 0$, so

$$g_i\big(\alpha\mathbf{x} + (1-\alpha)\mathbf{y}\big) \leq 0,$$

and hence $\mathbf{z} \in \Omega_i$, *i.e.*, any point on a chord between two point in the set must also be in the set, so Ω_i is convex.

Convex Sets

Proof (continued).

Now we move on to sets with multiple constraints.

$$\Omega = \left\{ \mathbf{x} \in \mathbb{R}^n \,\middle|\, g_i(\mathbf{x}) \leq 0, \text{ for } i = 1, 2, \ldots, m \right\} = \cap_{i=1}^{m} \Omega_i$$

We note two properties of convex sets:

- The empty set and the set $\mathbb{R}^n$ are convex.
- The intersection of two convex sets is convex.

Together these mean (*e.g.*, by induction) that $\Omega = \cap_{i=1}^{m} \Omega_i$ must be convex. □

You are asked to prove the intersection of two convex sets is convex in Assignment 0.

Linear bounds, and convexity

Lemma

The region defined by a set of linear inequalities $A\mathbf{x} \leq \mathbf{b}$ *and* $\mathbf{x} \geq 0$ *is convex.*

Proof.

- It is almost self-evident that a straight line is convex.
  - a chord along the straight line, is just that
  - they aren't *strictly* convex as the chord doesn't lie above the line, but that doesn't matter here
- From the previous lemma, the set of linear constraints $\mathbf{a}_i\mathbf{x} \leq b_i$ and $x_i \geq 0$ defines a convex set.

Why does it matter?

Theorem

If S is a convex set and f is a convex (concave) function, then any local minimum (maximum) of f is a global minimum (maximum).

Proof.

If x^* is a local minimum, then by definition for some small h

$$f(x^* + h) \geq f(x^*)$$

Now choose any other point $x > x^*$, we can draw a chord between x^* and x, and this chord must lie no lower than $f(\cdot)$. For instance, given $h > 0$, at $x^* + h$ the chord will be $\geq f(x^* + h)$. That implies the chord from $(x^*, f(x^*))$ to $(x, f(x))$ has non-negative slope, and hence $f(x) > f(x^*)$, which implies a global minimum.
We can repeat the argument for $x < x^*$. □

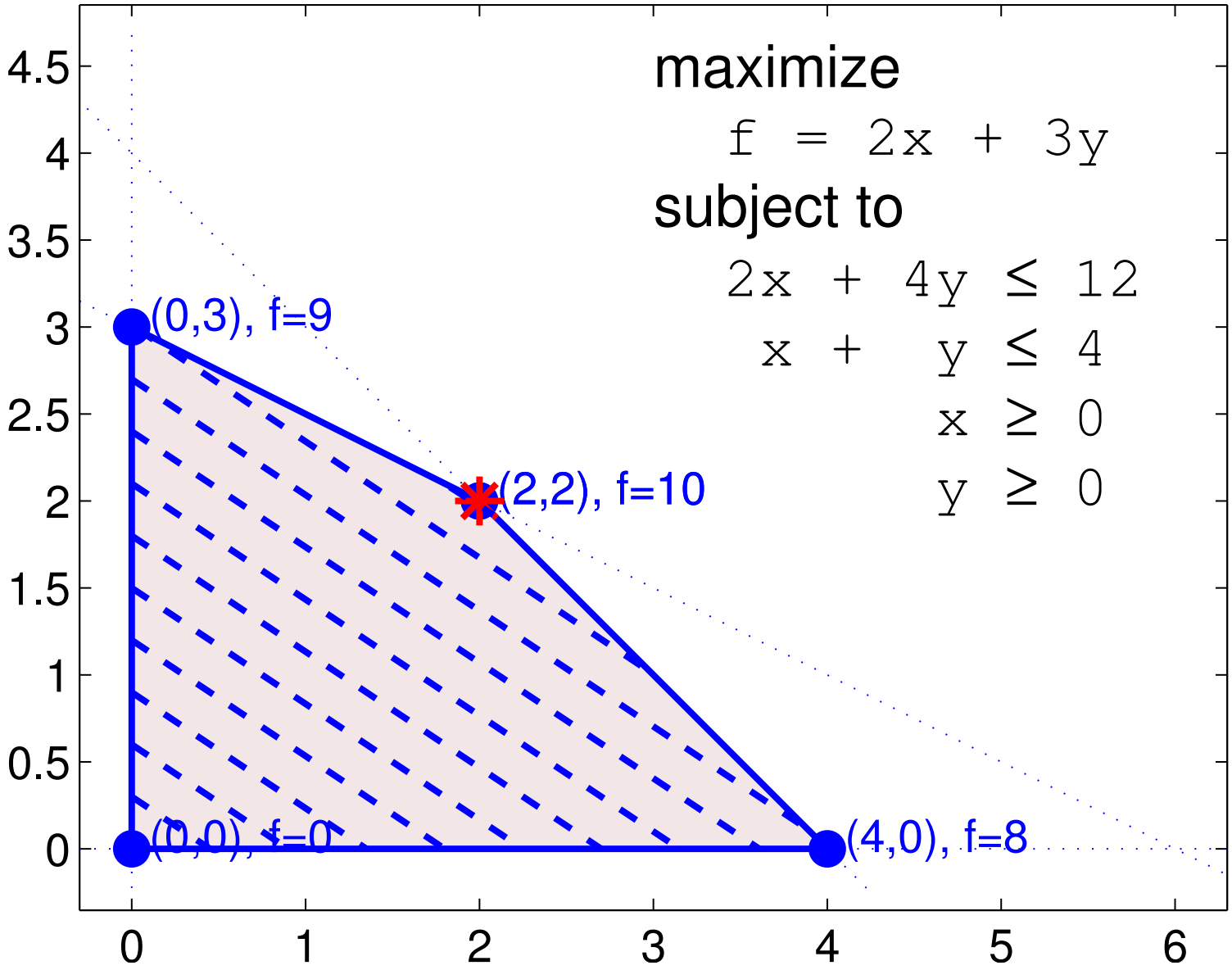
Note that the result doesn't require a differentiable function.

Geometry of linear programming

Obviously, the previous result can be generalised to n dimensions, and we see that

- Linear inequalities define a convex set, and the linear objective function is both convex and concave
- Hence the local minimum of a linear program will also be its global minimum
- The objective function here is linear, so $f' \neq 0$ anywhere, so the local minimum (if it exists) will occur on the boundary
- So the minimum of a linear program (if it exists) is somewhere on the boundary

Illustration



Section 3

Linear Equations

Linear Equations

Given a *system of linear equations* $A\mathbf{x} = \mathbf{b}$ there are three possibilities:

- They have no solutions
- They have exactly one solution
- They have infinitely many solutions

And we know how to test for these cases, and solve when possible.

Naïvely

For a square matrix A we can solve $A\mathbf{x} = \mathbf{b}$ by using the matrix inverse, *e.g.*,

$$\mathbf{x} = A^{-1}\mathbf{b}$$

But this has issues

- in our problem, the matrix might not be square
- even if it were square it might not be invertible (non-singular)
- even it is invertible, this approach is
  - numerically inefficient
  - potentially numerically unstable

So we don't do this

Gauss-Jordan elimination

Carl Friedrich Gauss (17771855)

Though apparently was known to Chinese mathematicians around the birth of Christ.

Gauss-Jordan elimination

- We augment the matrix A to include the RHS, *i.e.*, $M = [A \mid \mathbf{b}]$
- We perform an allowed set of operations $M \to M'$
  1. Interchange two rows
  2. Perform a *pivot* at element (i, j) which combines
     1. Multiply a row by a nonzero number; and,
     2. Add a multiple of one row to another row,

     such that element $M'_{i,j} = 1$ and $M'_{k,j} = 0$ for $k \neq i$

  The allowed operations result in an *equivalent* augmented matrix.
- The goal is to manipulate it into a form where the solution is obvious. For instance, if we could manipulate it into the form

$$[I|\mathbf{b}^*]$$

Then we can read off the solution $\mathbf{x} = \mathbf{b}^*$ because this augmented matrix corresponds to the equations

$$I\mathbf{x} = \mathbf{b}^*$$

Augmented matrix

Definition

Given a set of equations $A\mathbf{x} = \mathbf{b}$ *the*

- *coefficient matrix is* A
- *augmented matrix is* $[A|\mathbf{b}]$

- The augmented matrix contains all the information about the equations
  - there is a 1:1 correspondence between the augmented array and a set of equations
- We work on that to keep all the relevant data together
  - row operations on the augmented matrix convert to a new augmented matrix, corresponding to a set of equations which have the same solutions

Gauss-Jordan elimination example

Equations	Augmented matrix
$x_1 + 3x_2 + x_3 = 9$	1 3 1 9
$x_1 + x_2 - x_3 = 1$	1 1 -1 1
$3x_1 + 11x_2 + 5x_3 = 35$	1 11 5 35
$x_1 + 3x_2 + x_3 = 9$	1 3 1 9
$-2x_2 - 2x_3 = -8$	0 -2 -2 -8
$2x_2 + 2x_3 = 8$	0 2 2 8
$x_1 - 2x_3 = -3$	1 0 -2 -3
$x_2 + x_3 = 4$	0 1 1 4
$0 = 0$	0 0 0 0

We performed 2 pivots

1. `pivot(1,1)` (a pivot on the (1,1) element of the augmented matrix)
2. `pivot(2,2)`

Gauss-Jordan elimination example

In this form

Equations	Augmented array
$x_1 - 2x_3 = -3$	1 0 -2 -3
$x_2 + x_3 = 4$	0 1 1 4
$0 = 0$	0 0 0 0

we can read off the solution. There is a degree of freedom, because the last equation is always true. This gives us a *free* variable: here we take it to be $x_3 = t$, and the solution will be

$$\mathbf{x} = (-3, 4, 0) + t(2, -1, 1)$$

We could choose to set $x_3 = 0$, and get a solution $(-3, 4, 0)$.

Notes

- Formally we perform *elementary row operations*
  - Swap two rows.
  - Multiply a row by a nonzero scalar.
  - Add a scalar multiple of one row to another.
- We aim to put the tableau in *reduced row echelon form*
  - the lower-triangular part of the augmented matrix are all zeros
  - for every non-zero row, the leading coefficient is to the right of the leading coefficient of the row above
  - leading coefficients are 1
- Gauss-Jordan is a numerically unstable procedure in some cases
  - other approaches, *e.g.*, QR decomposition, are often preferred

General Case

For optimisation, we don't want a single solution to the constraints, or the optimisation is trivial.

Take $A\mathbf{x} = \mathbf{b}$, where A is of size $m \times n$

- n variables
- m equations

With *full rank* A, and $n > m$:

- There are ∞ solutions
  - we don't have enough information to select one
- Choose $(n - m)$ variables to be 0. Think of this as either
  - adding $n - m$ additional equations $x_i = 0$
  - reducing the number of variables down to m

  then there will (hopefully) be a unique solution for the other m variables

Terminology

Definition (Basic solution)

A *basic solution* to $A\mathbf{x} = \mathbf{b}$ is a solution with at least $n - m$ zero variables.

Definition (Non-degenerate basic solution)

A basic solution is *non-degenerate* iff exactly $n - m$ variables are zero.

It's *degenerate*[2] if there are more than $n - m$ zeros.

Definition (Basic and non-basic variables)

For a non-degenerate basic solution, the m non-zero variables are called *basic*, and the $n - m$ zero variables are *non-basic* or *free*.

[2] In mathematics, we call a case degenerate when it is qualitatively different from the other solutions, and thus belongs to a different class of (often simpler) solution. Often its features are lost under small perturbations. For example, a line is a degenerate parabola.

Takeaways

- Linear programs are guaranteed to be convex problems, and hence we only need to find a local minimum (maximum) and it will be guaranteed to be a global minimum (maximum)
- Linear equation techniques
  - Gauss-Jordan (pivots, etc)
  - Basic solutions

  will be vital for what we are doing
- If you don't remember this stuff, revise!!!
  - the first assignment has a bunch of revision questions

Further reading I

George B. Dantzig, *Programming of interdependent activities: Ii mathematical model*, Econometrica **17** (1949), no. 3/4, 200–211 (English), This is a revised version of a paper that appeared at the Cleveland Meeting of the Econometric Society on December 27, 1948.

R. De Leone, *The origin of operations research and linear programming*, `http://globopt.dsi.unifi.it/gol/Seminari/DeLeone2008.pdf`, 2008.

Marshall K. Wood and George B. Dantzig, *Programming of interdependent activities: I general discussion*, Econometrica **17** (1949), no. 3/4, 193–199 (English), This is a revised version of a paper that appeared at the Madison Meeting of the Econometric Society on September 9, 1948.