Agricultural Marketing and Price Analysis - Handout 2
Mathematical programming

Rodney Beard

September 25, 2016

In this lecture I will introduce you to linear and non-linear programming and application is agricultural marketing

This handout is your map to the lecture it will cover the key takeaway messages and the most important graphs will be placed in the right margin with a short explanation.

The aim of the lecture is to introduce you to mathematical programming and it's role in agricultural marketing.

Linear programming

Linear programming was invented by George Dantzig and independently by Leonid Kantorovich in the 1940's. It is a powerful tool with a wide range of applications in business and economics, in agricultural economics it has particular found applications in farm management: whole farm planning and in applications to feed mix problems. I has found applications to transportation planning that are important in supply chain management and marketing. There is a story that Geroge Dantzig was presenting linear programming at a seminar and that at the end of the presentation, Harold Hotelling said: 'Mr. Speaker, but we all know the world is not linear", to which John von Neumann replied: "The speaker titled his talk ?linear programming? and carefully stated his axioms. If you have an application that satisfies the axioms, well use it. If it does not, then don?t." (Source: `https://punkrockor.com/2014/04/29/happiness-is-assuming-the-world-is-linear/`. I've seen the story reported in a number of books not sure of the original citation).

Linear programming is concerned with maximizing a linear objective subject to constraints, mathematically we can formulate a linear program as follows :

$$\max \textit{or} \min c_1 x_1 + \ldots c_{1n} x_n$$

subject to

$$\begin{array}{ccc} a_{11}x_1 + & \ldots & +a_{1n}x_m \leq b_1 \\ \vdots & \ddots & \vdots \\ a_{21}x_1 + & \ldots & a_{2n}x_m \leq b_2 \end{array}$$

where $x_1, \ldots, x_n$ are the **decision variables**, i.e. the variables to be chosen.

$c_1, \ldots, c_n$ are cost or benefit parameters, the a_{ij} are technical coefficients, that convert the resources b_i into an output that is reflected in the costs.

There are a number of algorithms for solving linear programming problems, the most well-known of which due to Dantzig is the simplex algorithm `http://mathworld.wolfram.com/SimplexMethod.html`.

We can also write linear programs in matrix form:

$$\min c^T \vec{x}$$

subject to

$$A\vec{x} \leq \vec{b}$$

Using SciPy to Solve Linear Programming Problems

Example $\min 1x_0 + 4x_1$

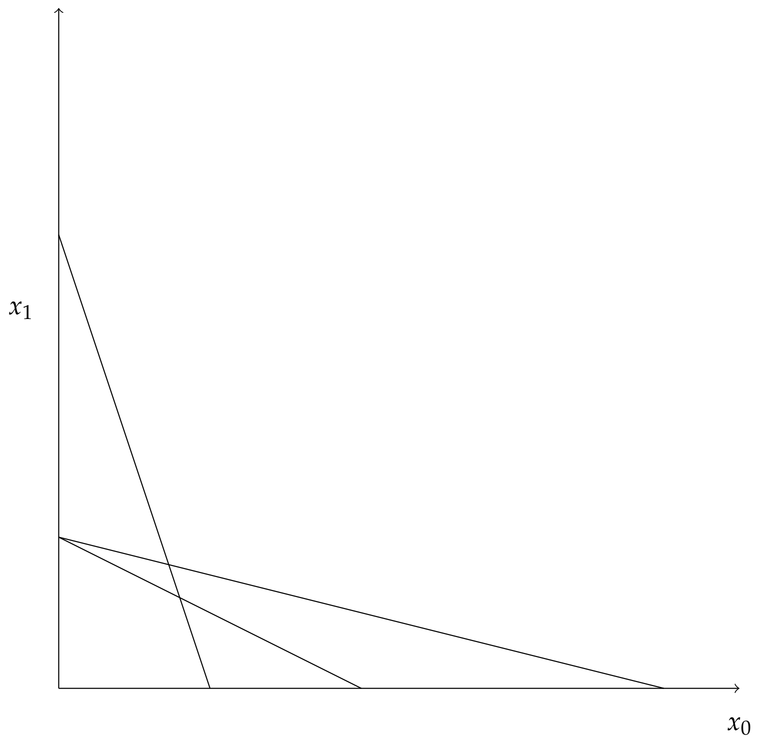
subject to

$3x0 + 1x_1 \leq 6$

$1x0 + 2x_1 \leq 4$

$x_1 \geq -3$

We can plot this example on a graph to find the **graphical solution**:


x_1
x_0

Using SciPy we can solve this as follows:

```
from scipy.optimize import linprog

c = [-1,-4]
A = [[3,1],[1,2]]
b = [6,4]
x0_bounds = (0,None)
x1_bounds = (0, None)
res = linprog(c,A_ub =A,b_ub=b,
bounds=(x0_bounds, x1_bounds),
options={"disp": True})
print(res)
```

This produces the output:

```
fun: -8.0
 message: 'Optimization␣terminated␣successfully.'
     nit: 1
   slack: array([ 4.,  0.])
  status: 0
 success: True
       x: array([ 0.,  2.])
```

from scipy.optimize import linprog calls SciPy's linear programming module linprog

Non-linear Programming

Non-linear programming generalizes linear programming by allowing for either non-linear objective functions or non-linear constraints or both.

Example

The source for this is `http://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fmin_cobyla.html`, cobyla essentially turns a non-linear programming problem into a linear programming problem automatically.

$$\max xy$$

$$x^2 + y^2 \leq 1$$

$$y \geq 0$$

```
from scipy.optimize import fmin_cobyla

def objective(x):
    return x[0]*x[1]

def constr1(x):
    return 1 - (x[0]**2 + x[1]**2)

def constr2(x):
    return x[1]

res = fmin_cobyla(objective, [0.0, 0.1],
[constr1, constr2], rhoend=1e-7)

print(res)
```

This produces the output: $[-0.70710685 \quad 0.70710671]$

Application: Spatial Price Equilibrium Models

We wish to maximize (see lecture slides):

$$NAW_V = \sum_{i=1}^{n}(a_i - 0.5b_iQ_i^D)Q_i^D - \sum_{i=1}^{n}(\alpha_i + 0.5\beta_iQ_i^S)Q_i^S - \sum_{i=1}^{n}\sum_{j=1}^{n}C_{ij}T_{ij}$$

or

$$NAW_H = \sum_{i=1}(a_i - 0.5b_iQ_i^D)Q_i^D - \sum_{i=1}^{n}0.5(\alpha_i + \beta_iQ_i^S)(Q_i^S + \frac{\alpha_i}{\beta_i}) - \sum_{i=1}^{n}\sum_{j=1}^{n}C_{ij}T_{ij}$$

These need to be maximized subject to the import supply and demand restrictions:

$$\sum_{j=1}^{n} T_{ji} \geq Q_i^D$$

$$\sum_{j=1}^{n} T_{ij} \geq Q_i^S$$

The first-order conditions for this are given by the Karush-Kuhn-Tucker conditions (see lecture slides).