COSC/DATA 405/505

Modelling and Simulation

Numerical Optimization

In many areas of data science one has to solve the following problem:

Given a function $f(\cdot)$, which value of x makes $f(x)$ as large or as small as possible?

Knowing how to do minimization is sufficient.

If we want to maximize $f(x)$, we simply change the sign and minimize $-f(x)$.

We call both operations "numerical optimization".

These slides introduce the basics of an important branch of optimization: linear programming.

Linear Programming

We often need to minimize a function subject to constraints.

When the function is linear and the constraints can be expressed as linear equations or inequalities, the problem is called a *linear programming* problem.

Linear Programming

The so-called standard form for the minimization problem in linear programming is

$$\min_{x_1, x_2, \ldots, x_k} C(x) = c_1 x_1 + \cdots c_k x_k$$

subject to the *constraints*

$$a_{11} x_1 + \cdots + a_{1k} x_k \geq b_1$$

$$a_{21} x_1 + \cdots + a_{2k} x_k \geq b_2$$

$$\cdots$$

$$a_{m1} x_1 + \cdots + a_{mk} x_k \geq b_m$$

and the *nonnegativity conditions* $x_1 \geq 0, \ldots, x_k \geq 0$.

Linear Programming

The idea is to find values of the *decision variables* $x_1, x_2, \ldots, x_n$ which minimize the *objective function* $C(x)$, subject to the constraints and nonnegativity conditions.

Linear Programming: Example

A company has developed two procedures for reducing sulfur dioxide and carbon dioxide emissions from its factory.

The first procedure reduces equal amounts of each gas at a per unit cost of $5.

The second procedure reduces the same amount of sulfur dioxide as the first method, but reduces twice as much carbon dioxide gas; the per unit cost of this method is $8.

The company is required to reduce sulfur dioxide emissions by 2 million units and carbon dioxide emissions by 3 million units.

What combination of the two emission procedures will meet this requirement at minimum cost?

Linear Programming: Example

Let x_1 denote the amount of the first procedure to be used, and let x_2 denote the amount of the second procedure to be used.

For convenience, we will let these amounts be expressed in millions of units.

Then the cost (in millions of dollars) can be expressed as

$$C = 5x_1 + 8x_2.$$

Since both methods reduce sulfur dioxide emissions at the same rate, the number of units of sulfur dioxide reduced will then be

$$x_1 + x_2$$

Noting that there is a requirement to reduce the sulfur dioxide amount by 2 million units, we have the constraint

$$x_1 + x_2 \geq 2.$$

The carbon dioxide reduction requirement is 3 million units, and the second method reduces carbon dioxide twice as fast as the first method, so we have the second constraint

$$x_1 + 2x_2 \geq 3.$$

Linear Programming: Example

Finally, we note that x_1 and x_2 must be nonnegative, since we cannot use negative amounts of either procedure.

Thus, we obtain the linear programming problem:

$$\min C = 5x_1 + 8x_2$$

subject to the constraints

$$x_1 + x_2 \geq 2$$

$$x_1 + 2x_2 \geq 3$$

and

$$x_1, x_2 \geq 0.$$

Linear Programming: Example

These relations are graphed in the figure on the next slide.

The region shaded in grey is the *feasible region*; this is the set of all possible (x_1, x_2) combinations which satisfy the constraints.

The unshaded area contains those combinations of values where the constraints are violated.

Linear Programming: Example

A graphical interpretation of the pollution emission linear programming example.

The grey region corresponds to values of x_1 and x_2 which satisfy all of the constraints.

The dashed grey line corresponds to values of x_1 and x_2 which give the minimum cost (13).

This line intersects the feasible region at exactly one point - the optimal solution to the problem $(1, 1)$.

Linear Programming: Example

The gradient of the function $C(x)$ is $(5, 8)$, so this vector gives the direction of most rapid increase for that function.

The level sets or contours of this function are perpendicular to this vector.

One of the level sets is indicated as a dashed line in the earlier figure.

The solution of the minimization problem lies at the intersection of the first contour which intersects the feasible region.

If this happens at a single point, we have a *unique* minimizer.

In this example, this intersection is located at the point $(1, 1)$.

Linear Programming

It can be shown that the only possible minimizers for such linear programming problems must be at the intersections of the constraint boundaries, as in the earlier example.

The points of intersection of the constraints are called *basic solutions*.

If these intersection points lie in the feasible region, they are called *basic feasible solutions*.

If there is at least one basic feasible solution, then one of them will be an *optimal solution*.

In the above example, the point $(1, 1)$ is the optimal solution.

Solving linear programming problems in R

There is more than one linear programming function available in R, but the lp() function in the lpSolve package may be the most stable version currently available.

It is based on the *revised simplex method*; this method tests a number of extreme points of the feasible region to see whether they are optimal.

Solving linear programming problems in R

The `lp()` function has a number of parameters; the following are needed to solve minimization problems like the one in the earlier example.

- `objective.in` **– the vector of coefficients of the objective function.**
- `const.mat` **– a matrix containing the coefficients of the decision variables in the left-hand side of the constraints; each row corresponds to a constraint.**
- `const.dir` **– a character vector indicating the direction of the constraint inequalities; some of the possible entries are** `>=`**,** `==` **and** `<=`**.**
- `const.rhs` **– a vector containing the constants given on the right-hand side of the constraints**

Solving Linear Programming Problems in R

To solve the minimization problem set out in the pollution example, type

```
library(lpSolve)
eg.lp <- lp(objective.in=c(5, 8), const.mat=matrix(c(1, 1, 1, 2),
              nrow=2), const.rhs=c(2, 3), const.dir=c(">=", ">="))
eg.lp

## Success: the objective function is 13
```

```
eg.lp$solution

## [1] 1 1
```

The output tells us that the minimizer is at $x_1 = 1$, $x_2 = 1$, and the minimum value of the objective function is 13.

Maximization and other kinds of constraints

The `lp()` function can handle maximization problems with the use of the `direction="max"` parameter.

Furthermore, the `const.dir` parameter allows for different types of inequalities.

Maximization: Example

We will solve the problem:

$$\max C = 5x_1 + 8x_2$$

subject to the constraints

$$x_1 + x_2 \leq 2$$

$$x_1 + 2x_2 = 3$$

and

$$x_1, x_2 \geq 0.$$

Maximization and other kinds of constraints

In R, this can be coded as

```
eg.lp <- lp(objective.in=c(5, 8),
            const.mat=matrix(c(1, 1, 1, 2), nrow=2),
            const.rhs=c(2, 3),
            const.dir=c("<=", "="), direction="max")
eg.lp$solution

## [1] 1 1
```

The solution is $(1, 1)$, giving a maximum value of 13.

Special situations: Multiple Optima

It sometimes happens that there are multiple solutions for a linear programming problem.

Multiple Optima: Example

A slight modification of the pollution emission example is

$$\min C = 4x_1 + 8x_2$$

subject to the constraints

$$x_1 + x_2 \geq 2$$

$$x_1 + 2x_2 \geq 3$$

and

$$x_1, x_2 \geq 0.$$

This problem has a solution at $(1, 1)$ **as well as at** $(3, 0)$**. All points on the line joining these two points are solutions as well. The figure on the next slide shows this graphically.**

Multiple Optima

A plot of the gradient of the objective function and the constraint boundaries for the pollution example.

The points on the heavy black segment are all optimal for this problem.

The `lp()` function does not alert the user to the existence of multiple minima.

In fact, the output from this function for the modified pollution emission example is the solution $x_1 = 3, x_2 = 0$.

Degeneracy

For a problem with m decision variables, degeneracy arises when more than m constraint boundaries intersect at a single point.

This situation is quite rare, but it has potential to cause difficulties for the simplex method, so it is important to be aware of this condition.

In very rare circumstances, degeneracy can prevent the method from converging to the optimal solution; most of the time, however, there is little to worry about.

Degeneracy: Example

The following problem has a point of degeneracy which is not at the optimum; however, the `lp()` function still finds the optimum without difficulty.

$$\min C = 3x_1 + x_2$$

subject to the constraints

$$x_1 + x_2 \geq 2$$

$$x_1 + 2x_2 \geq 3$$

$$x_1 + 3x_2 \geq 4$$

$$4x_1 + x_2 \geq 4$$

and

$$x_1, x_2 \geq 0.$$

Degeneracy: Example

An illustration of the concept of degeneracy.

A plot of four constraint boundaries, one of which is redundant, leading to degeneracy.
The feasible region is shaded.

Degeneracy

This problem can be solved easily:

```
degen.lp <- lp(objective.in=c(3, 1),
       const.mat=matrix(c(1, 1, 1, 4, 1, 2, 3, 1), nrow=4),
       const.rhs=c(2, 3, 4, 4), const.dir=rep(">=", 4))
degen.lp

## Success: the objective function is 3.333333
```

```
degen.lp$solution

## [1] 0.6666667 1.3333333
```

Infeasibility

Infeasibility is a more common problem.

When the constraints cannot simultaneously be satisfied there is no feasible region.

Then no feasible solution exists.

Infeasibility: Example

$$\min C = 5x_1 + 8x_2$$

subject to the constraints

$$x_1 + x_2 \geq 2$$

$$x_1 + x_2 \leq 1$$

and

$$x_1, x_2 \geq 0.$$

It is obvious that the constraints cannot simultaneously be satisfied.

Infeasibility: Example

Here is the output from the `lp()` function:

```
eg.lp <- lp(objective.in=c(5, 8),
            const.mat=matrix(c(1, 1, 1, 1), nrow=2),
            const.rhs=c(2, 1), const.dir=c(">=", "<="))
eg.lp


## Error: no feasible solution found
```

Unboundedness

In rare instances, the constraints and objective function give rise to an unbounded solution.

A trivial example of unboundedness arises when solving the problem

$$\max C = 5x_1 + 8x_2$$

subject to the constraints

$$x_1 + x_2 \geq 2$$

$$x_1 + 2x_2 \geq 3$$

and

$$x_1, x_2 \geq 0.$$

The feasible region for this problem was plotted earlier.

However, instead of trying to minimize the objective function, we are now maximizing, so we follow the direction of increasing the objective function this time.

Unboundedness

We can make the objective function as large as we wish, by taking x_1 and x_2 arbitrarily large.

Here is what happens when `lp()` is applied to this problem:

```
eg.lp <- lp(objective.in=c(5, 8),
            const.mat=matrix(c(1, 1, 1, 2), nrow=2),
            const.rhs=c(2, 3), const.dir=c(">=", ">="),
            direction="max")
eg.lp

## Error: status 3
```

The condition of unboundedness will most often arise when constraints and/or the objective function have not been formulated correctly.

Unrestricted variables

Sometimes a decision variable is not restricted to be nonnegative. The `lp()` function is not set up to handle this case directly. However, a simple device gets around this difficulty.

If x is unrestricted in sign, then x can be written as $x_1 - x_2$, where $x_1 \geq 0$ and $x_2 \geq 0$.

This means that every unrestricted variable in a linear programming problem can be replaced by the difference of two nonnegative variables.

Unrestricted Variables: Example

We will solve the problem:

$$\min C = x_1 + 10x_2$$

subject to the constraints

$$x_1 + x_2 \geq 2$$

$$x_1 - x_2 \leq 3$$

and

$$x_1 \geq 0.$$

Unrestricted variables

Noting that x_2 is unrestricted in sign, we set $x_2 = x_3 - x_4$ for nonnegative x_3 and x_4. Plugging these new variables into the problem gives

$$\min C = x_1 + 10x_3 - 10x_4$$

subject to the constraints

$$x_1 + x_3 - x_4 \geq 2$$

$$x_1 - x_3 + x_4 \leq 3$$

and

$$x_1 \geq 0, x_3 \geq 0, x_4 \geq 0.$$

Unrestricted Variables: Example

Converting this to R code, we have

```
unres.lp <- lp(objective.in=c(1, 10, -10),
      const.mat=matrix(c(1, 1, 1, -1, -1, 1), nrow=2),
      const.rhs=c(2, 3), const.dir=c(">=", "<="))
unres.lp

## Success: the objective function is -2.5
```

```
unres.lp$solution

## [1] 2.5 0.0 0.5
```

The solution is given by $x_1 = 2.5$ and $x_2 = x_3 - x_4 = -0.5$.