Decision Making with Business Analytics - Assignment 2

Mike Weltevrede (ANR: 756479, SNR: 1257560)
Daniel Kolenbrander (ANR: 949231, SNR: 1273593)

October 12, 2019

1 Linear Programming with Normal Form

At the top of the MATLAB file `dmba_assignment_2.M`, you can fill out the matrices for your optimisation problem. The following options are given, as used in the function `convert_to_standard` (described in Section 2, but already applied from the start):

- `f`: Coefficient or cost vector.
- `Aleq`: Matrix corresponding to the left-hand-side of less than or equal linear inequality constraints.
- `bleq`: Vector corresponding to the right-hand-side of less than or equal linear inequality constraints.
- `Ageq`: Matrix corresponding to the left-hand-side of greater than or equal linear inequality constraints.
- `bgeq`: Vector corresponding to the right-hand-side of greater than or equal linear inequality constraints.
- `Aeq`: Matrix corresponding to the left-hand-side of linear equality constraints.
- `beq`: Vector corresponding to the right-hand-side of linear equality constraints.
- `lb`: Vector of lower bounds.
- `ub`: Vector of upper bounds. Currently, the code assumes that all variables are unbounded from above.

`Aleq`, `Ageq`, and `Aeq` are $M \times N$ matrices, where M is the number of respective (in)equalities, and N is the number of variables (length of `f`), `bleq`, `bgeq`, and `beq` are M-dimensional vectors, and `lb` and `ub` are N-dimensional vectors (currently setting `ub` to be empty).

1.1 Optimal Basis

The `linprog` function in `MATLAB` solves the linear programming problem and returns the optimal solution. Thereafter, we identify the columns of the basic and nonbasic variables in our matrix `Aeq`, called B and N respectively. Here B represents the optimal basis matrix and is printed in the `Command Window`.

In this exercise, we will use the example of page 255 from the book[1], figure 17 (LINDO Output for HAL). For this example, we get the following output for B and N.

$$B = \begin{bmatrix} 3 & 2 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad N = \begin{bmatrix} 2 & 4 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

[1] Winston, W. L. (2004). Operations research applications and algorithms. Belmont (Calif.): Duxbury Press.

1.2 Lower and upper bounds of the b vector

The right hand side of the optimal table constraints is represented by $B^{-1}b$. When computing the range of the lower and upper bounds of the vector b such that the optimal basis is not changed, we use the fact that we can write the epsilon change in b as $B^{-1}b + B^{-1}\varepsilon$. Since we do not want the optimal basis to change we have B^{-1}b+$B^{-1}\varepsilon \geq 0$. Let $\delta = (-B^{-1}b$ divided element wise by $B^{-1})$. This leads to:

$$\delta_i \text{ is a(n) } \begin{cases} \text{lower bound if } (B^{-1})_i \geq 0 \\ \text{upper bound if } (B^{-1})_i < 0 \end{cases}$$

From this, the bounds are found and are displayed in a table. For our example, the output is given in Table A.1.

1.3 Lower and upper bounds of the cost vector

When determining the bounds of the basic variables of the cost vector, we use the fact that we can rewrite $\bar{C}_j$, representing the (new) value of c in the cost vector, as $\bar{C}_j = c_{BV}B^{-1}a_j - c_j$, where c_{BV} is the cost vector of the basic variables in our LP, a_j represents the j^{th} column of the constraint and c_j represents the current value of the j^{th} column in the cost vector. We rewrite c_{BV} as $c_{BV} + \varepsilon$, where ε represents the change in the coefficient of the basic variables of the cost vector. In order to remain in the same basis, $\bar{C}_j$ has to be greater or equal than 0. Therefore, we know that $c_{BV}B^{-1}a_j - c_j + \varepsilon B^{-1}a_j$ has to be greater or equal than 0. Let $\gamma = (\bar{C}_j$ divided element wise by $B^{-1}a_j)$. This leads to:

$$\gamma_i \text{ is a(n) } \begin{cases} \text{lower bound if } (B^{-1}a_j)_i \geq 0 \\ \text{upper bound if } (B^{-1}a_j)_i < 0 \end{cases}$$

From this, the bounds of the basic variables are found and are displayed in a table.

The bounds of the nonbasic variables in the cost vector are found by rewriting c_j in $\bar{C}_j$ as $c_j + \varepsilon$, where ε represents the change in the coefficient of the nonbasic variables in the cost function. In order to remain in the same basis $\bar{C}_j$ has to remain greater or equal than 0. After rewriting we know that the upper bounds of the nonbasic variables in the cost function are given by $c_{BV}B^{-1}a_j - c_j$. The lower bounds are always $-\infty$. From this, the bounds of the nonbasic variables are found and displayed together with the bounds of the basic variables in a table. For our example, the output is given in Table A.2.

2 Linear Programming with General Form

To extend the previously written M-file to work for the general form of the maximization problem, we have written the function `convert_to_standard`. Here, one can enter several matrices, as specified in Section 1. In essence what happens is that one big matrix is created that adds slack variables to the problem. For less/greater than or equal constraints, this means that their sign is positive and negative, respectively. For equality constraints, no slack variables need to be added. As such, we end up with the following block matrix A:

$$A = \begin{bmatrix} \texttt{Aleq} & I & O \\ \texttt{Ageq} & O & -I \\ \texttt{Aeq} & O & O \end{bmatrix}$$

where I is the identity matrix and O is the matrix of zeros. Note that this matrix represents a linear programming problem with equality constraints, so it should be passed into the `linprog` function as `Aeq`.

We also need to adapt the lower and upper bounds and the cost vector `f` accordingly. We add zero cost for all slack variables to `f` and set the lower bounds to be 0. Slack variables are unbounded from above, by definition.

A Tables

Lower Bound	Current Value	Upper Bound
-2400	0	300
650	800	900
800	1000	∞
800	900	∞
800	900	∞
1600	4000	4300

Table A.1: Bounds for the b vector such that the optimal basis is not changed.

Variable	Lower Bound	Current Value	Upper Bound
1	$-\infty$	600	800
2	975	1000	1200
3	$666\frac{2}{3}$	800	∞
4	$-\infty$	1300	$1333\frac{1}{3}$
5	$-333\frac{1}{3}$	-20	∞

Table A.2: Bounds for the c vector such that the optimal basis is not changed.