CS325 Linear Programming Project

by Eric Rouse

Resources

1. Piazza, especially the post by Matt Schreiber.
2. Solving a regression problem with linear programming.
3. Regression examples.
4. GNU Linear Programming Kit manual

Project Team

Eric Rouse.

Project 3, Problem 1

Linear program as an objective and set of constraints

Objective:

We make $8 for each fresh ham, $14 for each smoked ham and $11 for each ham smoked on the overtime shift. So the ham profit is the sum of 8 times the number of fresh hams (N_{HF}), 14 times the number of smoked hams (N_{HS}) and 11 times each overtime smoked ham (N_{HO}).

Using the same logic for pork bellies (N_{BF}, N_{BS}, N_{BO}) and picnic hams (N_{PF}, N_{PS}, N_{PO}) we can derive the profit objective.

MAXIMIZE Profit = $8 * N_{HF} + 14 * N_{HS} + 11 * N_{HO} + 4 * N_{BF} + 12 * N_{BS} + 7 * N_{BO} + 4 * N_{PF} + 13 * N_{PS} + 9 * N_{PO}$

Constraints:

The number of hams produced in a day is 480, the number of pork bellies is 400 and the number of picnic hams is 230. Also, the maximum number of smoked products during normal operating hours is 420. There is also a maximum number of smoked products that can be produced during overtime production, 250.

Thus we have the following constraints.

1: $N_{HF} + N_{HS} + N_{HO} = 480$

2: $N_{BF} + N_{BS} + N_{BO} = 400$

3: $N_{PF} + N_{PS} + N_{PO} = 230$

4: $N_{HS} + N_{BS} + N_{PS} <= 420$

5: $N_{HO} + N_{BO} + N_{PO} <= 250$

6: $N_{HF}, N_{HS}, N_{HO}, N_{BF}, N_{BS}, N_{BO}, N_{PF}, N_{PS}, N_{PO} <= 0$

Linear program in matrix form

Description

In order to put the program into matrix form we need solutions to the max profits of each subset (ham, pork bellies and picnic hams) as they are not provided. But the maximum profit of each meat subset is trivial to calculate.

For instance, using the ham as an example, there are 480 produced. If we assume they are produced and smoked then 420 are sold at $14 profit and the remaining 60 are sold for $11 profit. The maximum possible profit from ham is then $6,540. Doing the same for pork bellies and picnic hams produces maximums of $4,800 and $2,990 respectively.

Also, to get into "standard form" for linprog we must invert each $N_{XX} <= 0$ to $- N_{XX} >= 0$ by multiplying both sides by -1.

Matrix

$$\begin{bmatrix} 8 & 14 & 11 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 12 & 7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 13 & 9 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} * \begin{bmatrix} N_{HF}, & N_{BF}, & N_{PF} \\ N_{HS}, & N_{BS}, & N_{PS} \\ N_{HO}, & N_{BO}, & N_{PO} \end{bmatrix} = \begin{bmatrix} 6540 \\ 4800 \\ 2990 \\ 420 \\ 250 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Optimal solution to linear program

The optimal solution to the linear program is a profit of $10,910. This seems reasonable because if we sum the three (unattainable) maximums of each meat subset we get $14,330.

Environment/Language/Solver

Environment

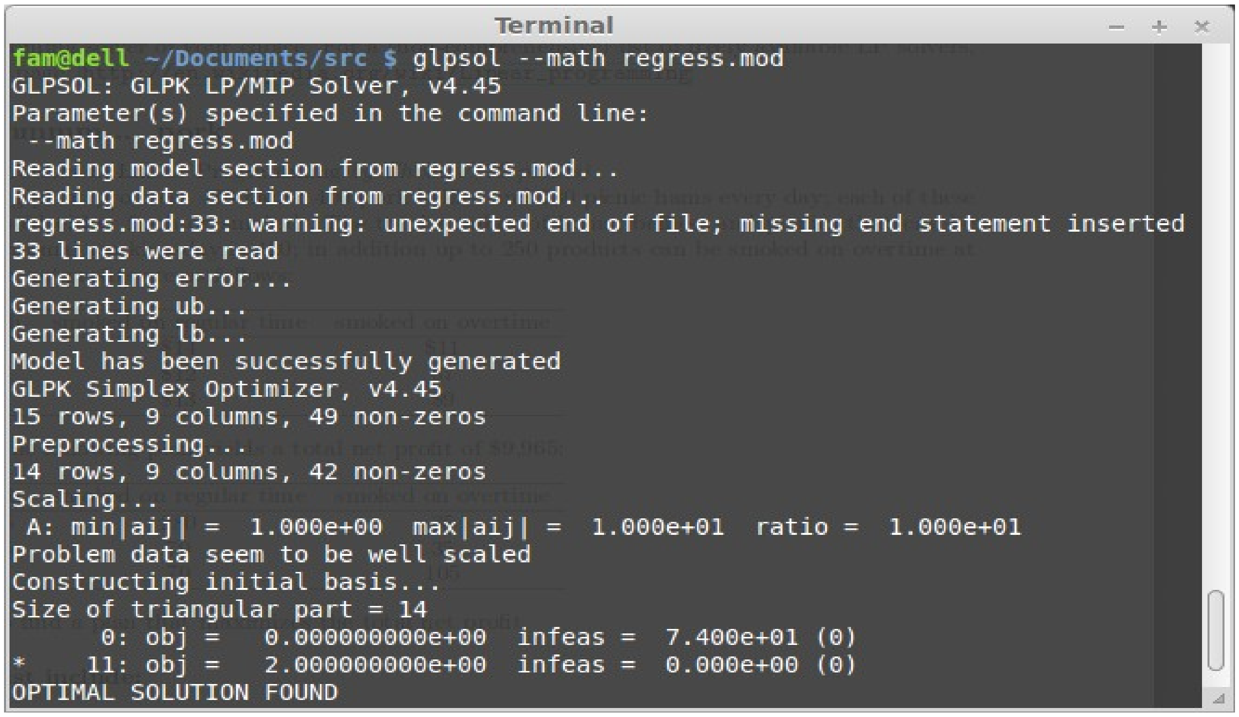
I use Arch Linux with a GNU desktop.

Language

I used GLPK, or GNU linear programming kit. It has a built in solver and a c library. I started using the c library but soon realized that the built in solver was plenty powerful enough.

Solver

The solver provided by GLPK is call glpsol, it is invoked on the command line.



```
fam@dell ~/Documents/src $ glpsol --math regress.mod
GLPSOL: GLPK LP/MIP Solver, v4.45
Parameter(s) specified in the command line:
 --math regress.mod
Reading model section from regress.mod...
Reading data section from regress.mod...
regress.mod:33: warning: unexpected end of file; missing end statement inserted
33 lines were read
Generating error...
Generating ub...
Generating lb...
Model has been successfully generated
GLPK Simplex Optimizer, v4.45
15 rows, 9 columns, 49 non-zeros
Preprocessing...
14 rows, 9 columns, 42 non-zeros
Scaling...
 A: min|aij| =  1.000e+00  max|aij| =  1.000e+01  ratio =  1.000e+01
Problem data seem to be well scaled
Constructing initial basis...
Size of triangular part = 14
      0: obj =   0.000000000e+00  infeas =  7.400e+01 (0)
*    11: obj =   2.000000000e+00  infeas =  0.000e+00 (0)
OPTIMAL SOLUTION FOUND
```

The Code

The glpsol package can read a number of input files. For this problem I used the CPLEX LP format:

```
\* pork.lp *\
Maximize
value: 8 Nhf + 14 Nhs + 11 Nho + 4 Nbf + 12 Nbs + 7 Nbo + 4 Npf
+ 13 Nps + 9 Npo

Subject To
fresh: Nhf + Nhs + Nho = 480
belly: Nbf + Nbs + Nbo = 400
picnc: Npf + Nps + Npo = 230
smoke: Nhs + Nbs + Nps <= 420
otsmk: Nho + Nbo + Npo <= 250

Bounds
Nhf >= 0 \number of hams, fresh
Nhs >= 0 \number of hams, smoked
Nho >= 0 \number of hams, smoked on overtime
Nbf >= 0 \number of pork bellies, fresh
Nbs >= 0 \number of pork bellies, smoked
Nbo >= 0 \number of pork bellies, smoked on overtime
Npf >= 0 \number of picnic hams, fresh
Nps >= 0 \number of picnic hams, smoked
Npo >= 0 \number of picnic hams, smoked on overtime

End
```

It is invoked on the command line using the –lp option: `glpsol -lp pork.lp`. Which produces the following output (highlighting added for clarity):

```
GLPSOL: GLPK LP/MIP Solver, v4.45
Parameter(s) specified in the command line:
 --lp pork.lp
Reading problem data from `pork.lp'...
5 rows, 9 columns, 15 non-zeros
23 lines were read
GLPK Simplex Optimizer, v4.45
5 rows, 9 columns, 15 non-zeros
Preprocessing...
5 rows, 6 columns, 12 non-zeros
Scaling...
 A: min|aij| =  1.000e+00  max|aij| =  1.000e+00  ratio =
1.000e+00
Problem data seem to be well scaled
Constructing initial basis...
Size of triangular part = 5
*     0: obj =   6.360000000e+03  infeas =  0.000e+00 (0)
*     4: obj =   1.091000000e+04  infeas =  0.000e+00 (0)
OPTIMAL SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.1 Mb (54419 bytes)
```

Project 3, Problem 2

Linear program (general) as an objective and set of constraints

Objective:

Since we want to find the values of a, b and c that minimize the function max(|a*x_i + b*y_i - c|) where 1 <= i <= n using linear programming we must first linearize the function. This is easiest done by letting z = max(|a*x_i + b*y_i - c|) for i in range 1 to n. Thus we are now minimizing z.

MINIMIZE z

Constraints:

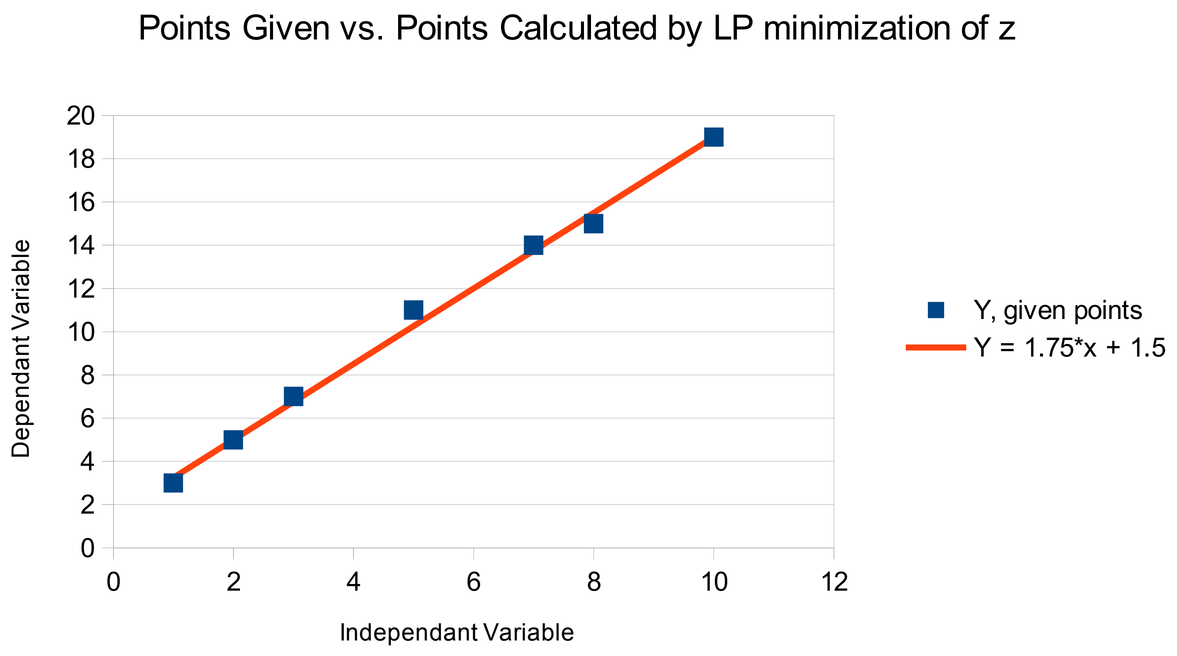
Using the fact that z = |a*x_i + b*y_i - c| and z = |-1*(a*x_i + b*y_i – c|)| are equivalent we can infer that z >= |a*x_i + b*y_i - c| and z <= |-a*x_i - b*y_i + c| for i in range 1 to n. In order to fight the a = b = c = 0 paradox, set b = 1 and adjust only a and c. We now are able to derive our constraints at each node.

node(i): z >= a*x_i + c - y_i **AND** z <= -1*(a*x_i + c - y_i)

Optimal solution to linear program

The optimal solution to the linear program are the values of a =1.75 and b = 1 and c = 1.5 .

Plot of points and solution



The Code

The glpsol package can read a number of input files. For this problem I used the GNU Mathprog format:

```
/*regress.mod*/
param n, integer, >=2;
set N, default{1..n};
param x{N};
param y{N};

var z{N}, >=0;
var a;
var b;
var c;

minimize error: sum{i in N} z[i];

s.t. ub{i in N}: z[i] >= a*x[i] + c - y[i];

s.t. lb{i in N}:  z[i] >= -1 * (a*x[i] + c - y[i]);

solve;

display a,c,error;

data;

param n :=7;

param : N : x y :=
     1   1 3
     2   2 5
     3   3 7
     4   5 11
     5   7 14
     6   8 15
     7   10 19;
```

It is invoked on the command line using the –math option: `glpsol -math regress.mod`.
Which produces the following output (highlighting added for clarity):

```
GLPSOL: GLPK LP/MIP Solver, v4.45
Parameter(s) specified in the command line:
 --math regress.mod
Reading model section from regress.mod...
Reading data section from regress.mod...
regress.mod:33: warning: unexpected end of file; missing end
statement inserted
33 lines were read
Generating error...
Generating ub...
Generating lb...
Model has been successfully generated
GLPK Simplex Optimizer, v4.45
15 rows, 9 columns, 49 non-zeros
Preprocessing...
14 rows, 9 columns, 42 non-zeros
```

```
Scaling...
 A: min|aij| =  1.000e+00  max|aij| =  1.000e+01  ratio =
1.000e+01
Problem data seem to be well scaled
Constructing initial basis...
Size of triangular part = 14
      0: obj =   0.000000000e+00  infeas =  7.400e+01 (0)
*    11: obj =   2.000000000e+00  infeas =  0.000e+00 (0)
OPTIMAL SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.1 Mb (136210 bytes)
Display statement at line 20
a.val = 1.75
c.val = 1.5
error.val = 2
Model has been successfully processed
```