Diet Problem Optimization

Riley Kenyon

February 13, 2020

1 Problem Statement

The diet problem is an optimization problem where quantities of nutrients are suggested by the FDA and are thereby followed to remain healthy. A sample list of food is provided with all of the included nutrients per serving. The cost function in this problem is determined by the price per serving of each variety of food. The objective in this optimization is to minimize the cost needed to satisfy the nutrient requirements. In addition to the minimization, there is a maximum quantity of nutrients that cannot be exceeded. After determining the minimum solution, another variation of the problem will be optimized to maximize the amount of protein in the diet while still meeting all other nutrient conditions.

2 Formation of Optimal Solution

The nutrients of interest are described in table (1). There are thresholds on the lower and upper end of the values. A linear programming solver, such as Matlab's $LinProg()$, will be used to yield the optimal solution. In the formulation of the optimal solution that minimizes the cost ($), there are four items that are essential to understanding the solution: $\mathbf{A}, \mathbf{c}, \mathbf{b}, \mathbf{x}$. This matrix (A) and vectors (c,b,x) represent different sections of the problem but are used to fully define the constraints for the solver. Matrix A is the description of the system, which is the various foods that could be used to obtain nutrients. Each food group has a different amount of calories, cholesterol, protein, etc. The matrix is constructed as the augmentation of column vectors belonging to each food group. The rows in the matrix

Nutrient	**Unit**	**Min**	**Max**
Calories	cal	2000	2250
Cholesterol	mg	0	300
Total Fat	g	0	65
Sodium	mg	0	2400
Carbohydrates	g	0	300
Dietary Fiber	g	25	100
Protein	g	50	100
Vitamin A	IU	5000	50000
Vitamin C	IU	50	20000
Calcium	mg	800	1600
Iron	mg	10	30

Table 1: The daily values as mandated by the FDA. The diet must remain within the minimum and maximum bounds.

thereby correspond to the nutrients. A representation of the matrix is shown in equation (1), where f_i corresponds to the information of a given food group. The system can also be represented in equation (2), where v_i are the values associated with a single nutrient across all food groups. The number of servings per food group will serve as x, the column vector of size n. This vector is used in conjunction with the c vector to create the cost function. The c vector is a column vector representation of cost per serving. When multiplied with the number of servings, the result is overall cost ($). This is the quantity that the solver will attempt to minimize. Lastly and most importantly is the constraints put on the overall number of nutrients, represented by the vector b (a column vector of length m).

$$A = \begin{pmatrix} f_1 & f_2 & \cdots & f_n \end{pmatrix} \tag{1}$$

$$A = \begin{pmatrix} v_1 & v_2 & \dots & v_n \end{pmatrix}^T \tag{2}$$

Recall that Ax is the overall quantity of nutrients for the given selection of servings per food group. At this point the formulation of the overarching system can be constructed. There are maximum requirements represented by b_{max}, such that the system multiplied with the number of servings, Ax, must be less than that quantity. Mathematically, this is represented in equation 3.

$$Ax \leq b_{max} \tag{3}$$

$$Ax \geq b_{min} \tag{4}$$

$$-Ax \leq -b_{min} \tag{5}$$

On the other end, there are minimum requirements represented by b_{min}. The same conditions apply, such that the overall nutrients must exceed that minimum value. The representation is shown in equation 4. However, in order to utilize the solver the constraints must be in an inequality form, specifically less than. To remedy the difference, the inequality is multiplied through by a negative to flip the direction of the inequality. The resultant equation is shown by equation 5. Lastly, the solver will need bounds on the number of servings. Logically, there can not be negative amounts of servings, so x will be constrained to positive quantities. Using a similar trick to what was described above, the identity matrix will be used with a negative representation to yield equation 6.

$$-Ix \leq 0 \tag{6}$$

Using the three separate matrices, they will be augmented to form a single matrix $\hat{A}$ that is used with the solver. The same applies for the constraint matrix $\hat{b}$. The representation of the two are found is equation 7 and 8 respectively.

$$\hat{A} = \begin{bmatrix} A \\ -A \\ -I \end{bmatrix} \tag{7}$$

$$\hat{b} = \begin{bmatrix} b_{max} \\ b_{min} \\ \vec{0} \end{bmatrix} \tag{8}$$

2.1 Optimal Results

Now that the problem is completely defined and ready to be used with the solver, the process is a single line of code: $linProg(c, \hat{A}, \hat{b})$. The solver performance an optimization on the premise of the following relation:

$$\begin{aligned} &\min c^T x \\ &\text{s.t. } Ax \leq b \end{aligned} \tag{9}$$

The results of which are shown in table 2. The overall cost accrued in the solution is \$0.956 consisting of mostly vegetables and popcorn.

Food	**Quantity**	**Cost ($)**
Frozen Broccoli	0.235	0.0165
Potatoes, Baked	3.545	0.213
Skim Milk	2.168	0.282
Peanut Butter	3.601	0.252
Popcorn, Air-Popped	4.823	0.193
Total Cost:		0.956

Table 2: Optimum solution for minimizing cost and satisfying bounds of nutrient requirements.

Food	**Quantity**	**Cost ($)**
Frozen Broccoli	0.252	0.0403
Carrots,Raw	0.036	0.0025
Skim Milk	3.949	0.5130
Poached Egg	1.336	0.1069
Peanut Butter	2.754	0.1928
Popcorn,Air-Popped	10.390	0.4156
Total Cost:		1.271

Table 3: Optimum solution for minimizing cost, satisfying bounds of nutrient requirements, and maximizing protein.

2.2 Protein Variation

Although most cost effective, it is more beneficial to create additional constraints to adapt to personal preferences. One such variation is to maximize the amount of protein in the solution. To do this, the problem will remain mostly the same. All cost functions and system descriptions are equivalent to the previous setup. However, in the constraint section of the formulation, the minimum value in protein is replaced by the maximum value. Thereby in the formulation of equations 5 and 3, the intersection of the requirements of protein imposes a solution where the selected servings contain protein equal to the maximum daily value. Rerunning $linProg(c, \hat{A}, \hat{b})$ results in a different solution by adding in poached eggs to the mix. The solution to this setup is shown in table 3.

3 Conclusion

The diet problem is a unique optimization problem that can be solved using linear programming. In the formulation of the setup, manipulations are required to be made to utilize a solver that assumes an inequality. The cost function in this example is to minimize cost ($), however it could be number of servings or by food preference using different weightings to shift serving sizes one direction or another. Conditions can also be placed on the constraints of the values to obtain specific solutions meeting those constraints. Quantities such as x are required for both the cost function and constraints for the system. After determining what the vector contains, the matrix A can be massaged to yield the form required to create the inequality. In general, it can be thought of as a multitude of row vectors, or column vectors. The solution using the food groups in this example consisted mainly of vegetables and items with high content of a specific nutrient. A protein emphasis shifted the solution to contain another group, poached eggs, which is high in protein.

A Code

```
%% Homework #4 - Diet Problem
% The diet problem seeks to satisfy the cost function of required
% nutrients, the constraint is the minimum number of nutrients mandated by
% the FDA, and the variables containing various nutrient values are
% described in the excel spreadsheet. The prolem will be solved using
% linear programming and the linprog(c,A,b) function.
close all; clear all; clc;

% Import matrices
load lpmatrices.mat
load names.mat

% Get to form: Ax < b_max, Ax > b_min, x > 0
A_hat = [A; -A; -eye(size(A,2))];
b_hat = [b(:,2);-b(:,1);zeros(size(A,2),1)];

% Linear Programming
x = linprog(c,A_hat,b_hat);
```

```
cost = c*x;
names(x~=0)

% Adjust to personal preference: Maximize protein
b(7,1) = b(7,2);    % set minimum value to upper bound
b_hat = [b(:,2);-b(:,1);zeros(size(A,2),1)];

x_new = linprog(c,A_hat,b_hat);
cost_new = c*x_new;
names(x_new~=0)
```