Linear programming basics (draft)

M. Hassell

January 16, 2017

1 Introduction

This document is a place for me to collect and organize information regarding linear programming and related optimization problems. My goal is to learn the basics of the theory and algorithms enough to eventually implement some of the basic tools. I'd ultimately like to progress to problems like integer and mixed integer programming, nonlinear programming, and the like. Remark: this is a working document and will be updated with new and improved information as I learn more. I'll include a bibliography at the end for my sources. I may or may not cite my sources inline, since some of this material will be from Wikipedia (okay, a lot of it will be from Wikipedia), but then some of it will come about as I start to understand the problem from various angles, and some of it will come from various books and internet sources. Anything included verbatim that isn't "common knowledge" will be cited. Also, if anyone reading this finds any errors or typos, a message would be much appreciated.

2 Linear programming

A linear programming problem is an optimization problem of the form

$$\max_{\mathbf{x}} f(\mathbf{x}) := \mathbf{c}^T\mathbf{x} \tag{1a}$$

$$\text{subject to} \quad \mathbf{A}\mathbf{x} \le \mathbf{b}, \quad \mathbf{x} \ge 0 \tag{1b}$$

The inequality constraints in (1b) can be understood component-wise, i.e. the first linear inequality constraint would read

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \le b_1, \quad x_1 \ge 0.$$

At the moment, I will not be very strict with whether or not we seek to maximize or minimize the objective function. The key points can be changed between the context of minimization and maximization by simply negative the objective function.

A plethora of problems can be modeled as linear programming problems. Here are a few for flavor:

(TBD)

To solve a linear programming problem, we can apply the Simplex Algorithm. To apply the simplex algorithm, we can convert the LP problem to into augmented (also known as slack) form. We introduce non-negative slack variables to replace inequality constraints with equality constraints and positivity constraints. We can rewrite problem (1a)-(1b) as

$$\begin{pmatrix} 1 & -\mathbf{c}^T & \mathbf{0} \\ 0 & \mathbf{A} & \mathbf{I} \end{pmatrix} \begin{pmatrix} z \\ \mathbf{x} \\ \mathbf{s} \end{pmatrix} = \begin{pmatrix} 0 \\ \mathbf{b} \end{pmatrix}, \tag{2}$$

with $\mathbf{x} \geq 0$, $\mathbf{s} \geq 0$ [3]. Geometrically, this transition from inequality constraints to equality constraints pushes the region of interest from the interior and boundary of the polytope to only it's surface.

To move towards the Simplex Method, we must first discuss the notion of basic feasible solutions. If we can write our matrix as $\mathbf{A} = [\mathbf{B}\,\mathbf{N}]$ where $\mathbf{B}$ is an invertible $m \times m$ matrix, and $\mathbf{N}$ is an $m \times (n-m)$ matrix. The variables associated with $\mathbf{B}$ will be called *basic* variables, and the variables associated with $\mathbf{N}$ will be called the *nonbasic variables.* The solution $\mathbf{x} = \begin{bmatrix} \mathbf{x}_B \\ \mathbf{x}_N \end{bmatrix}$ where

$$\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b}$$

and

$$\mathbf{x}_N = \mathbf{0}$$

is called a basic solution of the system [2]. Moreover, if $\mathbf{x}_B > 0$, then we say that it is a basic feasible solution. This corresponds to a an extreme point (vertex) of the simplex under consideration. We will refer to the matrix $\mathbf{B}$ as a basic matrix, or more simply, as a basis (in the vector space sense). Note that in general there can be several ways to construct a basic solution. As long as we can extract a matrix of full column rank from the matrix $\mathbf{A}$, we can find a basic solution.

It can be shown (I'll add the proof later) that for a linear program in standard form, if the objective function has a maximum value on the feasible region, then it has this value on at least one of the extreme points. In addition, we can assume that the rank of the matrix $\mathbf{A}$ is equal to the number of rows of $\mathbf{A}$. If not, then either $\mathbf{A}$ has redundant equations that can be dropped, or the system is inconsistent and has no solutions.

At first glance, it seems we are done. If our optimization problem has a solution (it is neither inconsistent nor unbounded), then all we need to do is enumerate the vertices of our simplex, evaluate the objective function, and keep the smallest instance. A back-of-the-envelope calculation yields that the number of vertices of a simplex can grow as quickly as

$$\begin{pmatrix} n \\ m \end{pmatrix} = \frac{n!}{m!(n-m)!}$$

which becomes intractable for even modest m and n. To overcome the combinatorial growth of the number of possible solutions, we turn to the Simplex Method of Dantzig. This begins by generating an initial feasible solution, and then moving along the edges of the simplex to adjacent vertices corresponding to descent (or ascent) directions of the objective function. In this way we can stay on the boundary of the simplex and move from extreme point to extreme point and continually push the value of the objective function in the desired direction (up for maximization problems and down for minimization problems). There are a few computational challenges that can arise, such as stalling and cycling, but I won't discuss them just yet.

We can now proceed to solve a true LP problem, pulled directly from [4]. We consider the following:

$$\max z := x_1 + 2x_2 - x_3$$

subject to

$$\begin{aligned} 2x_1 + x_2 + x_3 &\leq 14, \\ 4x_1 + 2x_2 + 3x_3 &\leq 28, \\ 2x_1 + 5x_2 + 5x_3 &\leq 30, \end{aligned}$$

and $x_1 \geq 0$, $x_2 \geq 0$, $x_3 \geq 0$.

To apply the simplex method, we first convert the problem to augmented/slack form. It is as follows:

$$\max z$$

subject to

$$\begin{aligned} 2x_1 + x_2 + x_3 + s_1 &= 14, \\ 4x_1 + 2x_2 + 3x_3 + s_2 &= 28, \\ 2x_1 + 5x_2 + 5x_3 + s_3 &= 30, \end{aligned}$$

and $x_i,\ s_i \geq 0$ for $i = 1,\ 2,\ 3$.

We then form the simplex tableu, which has the following form

$$\begin{array}{cccccc|c} 2 & 1 & 1 & 1 & 0 & 0 & 14 \\ 4 & 2 & 3 & 0 & 1 & 0 & 28 \\ 2 & 5 & 5 & 0 & 0 & 1 & 30 \\ \hline -1 & -2 & 1 & 0 & 0 & 0 & 0. \end{array} \tag{3}$$

Before we proceed further, let's pick apart some of the structure in this object. The first three columns contain the original coefficients from the linear inequality constraints. Essentially, we've copied $\mathbf{A}$ into this position. The next three columns are the identity matrix $\mathbf{I}_3$, corresponding to the slack variables s_i, $i = 1, 2, 3$. The rightmost column contains the constraints from the $\mathbf{b}$ vector. The bottom row contains in the first three columns the negative of the gradient of the objective function z. The last row is called the indicator row. What we do next feels somewhat like a combination of Gaussian elimination and gradient descent. We start with the column containing the most negative coefficient in the last row. In this case, the most negative coefficient is -2 in the second column. This is referred to as the pivot column. We form the quotients of elements in the last column with the elements in the corresponding row of the pivot column. In this case, the pivot column is $[1\ 2\ 5]^T$, so we form the quotients 14/1, 28/2, and 30/5, akin to the Matlab operation [14 28 30]./[1 2 5]. From this list, we pick the smallest non-negative quotient. This corresponds to the maximum distance that we can move the variable under consideration without leaving the feasible region. In this instance, the smallest non-negative quotient is $30/5 = 6$ in the last row. We then divide the entire pivot row by 5, and arrive at the simplex tableau

$$\begin{array}{cccccc|c} 2 & 1 & 1 & 1 & 0 & 0 & 14 \\ 4 & 2 & 3 & 0 & 1 & 0 & 28 \\ 2/5 & 1 & 1 & 0 & 0 & 1/5 & 6 \\ \hline -1 & -2 & 1 & 0 & 0 & 0 & 0. \end{array} \tag{4}$$

We now proceed as we would in Gaussian elimination and use elementary row operations to zero out the entries in the first two rows of the simplex tableu. Upon performing elementary row operations (EROs) to clear the second column out, the simplex tableu has the following entries

$$\begin{array}{cccccc|c} 8/5 & 0 & 0 & 1 & 0 & -1/5 & 8 \\ 16/5 & 0 & 1 & 0 & 1 & -2/5 & 16 \\ 2/5 & 1 & 1 & 0 & 0 & 1/5 & 6 \\ \hline -1/5 & 0 & 3 & 0 & 0 & 2/5 & 12. \end{array} \tag{5}$$

We now look at our indicator row again, and see that the most negative entry is $-1/5$. Our pivot column will be the first column. We form the quotients between the pivot column and the last column to find a tie: $8/(8/5) = 5$, $16/(16/5) = 5$, and $6/(2/5) = 15$. We can choose either the first or second row without a problem. For this exercise, we take the middle row. Row reducing as before gives us the new tableu

$$\begin{array}{cccccc|c} 0 & 0 & -1/2 & 1 & -1/2 & 0 & 0 \\ 1 & 0 & 5/16 & 0 & 5/16 & -1/8 & 5 \\ 0 & 1 & 7/8 & 0 & -1/8 & 1/4 & 4 \\ \hline 0 & 0 & 49/16 & 0 & 1/16 & 3/8 & 13. \end{array} \tag{6}$$

Our indicator row now has all positive entries, so there are no more ascent directions, so we are done. We then read off the solution as follows:

$$\begin{array}{cccccc|c} x_1 & x_2 & x_3 & s_1 & s_2 & s_3 & \\ 0 & 0 & -1/2 & 1 & -1/2 & 0 & 0 \\ 1 & 0 & 5/16 & 0 & 5/16 & -1/8 & 5 \\ 0 & 1 & 7/8 & 0 & -1/8 & 1/4 & 4 \\ \hline 0 & 0 & 49/16 & 0 & 1/16 & 3/8 & 13. \end{array} \tag{7}$$

The columns that have been reduced to the identity matrix (marked in red) correspond to the variables we have effectively solved for. We have $x_1 = 5$, $x_2 = 4$, and $s_1 = 0$. We can pull the last quantity, s_1, back into the original variables by solving to find that $x_3 = 0$. The value of the objective function is given in the bottom right corner. In this case, it has the maximum value $z = 13$. The remaining variables (corresponding to non-identity columns) are all zero.

3 Solving LP problems in Python

For now, we'll jump into using SciPy to solve some linear programs. The LP solvers are in the module `scipy.optimize.linprog`. The problem is as follows:

$$\begin{array}{ll} \text{minimize} & \mathbf{c}^T\mathbf{x} \\ \text{subject to} & \mathbf{A}_{ub}\mathbf{x} \leq \mathbf{b}_{ub} \\ & \mathbf{A}_{eq}\mathbf{x} = \mathbf{b}_{eq}. \end{array}$$

The example problem from the documentation is quoted below [1].

$$\begin{array}{c} \text{minimize} \quad -x_0 + 4x_1 \\ \text{subject to} \begin{pmatrix} -3 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \leq \begin{pmatrix} 6 \\ 4 \end{pmatrix} \\ x_0 \in \mathbb{R}, x_1 \geq -3. \end{array}$$

This is input to SciPy as follows:

```
from scipy.optimize import linprog

c = [-1,4]
A=[[-3,1],[1,2]]
b =[6,4]
x0_bounds = (None, None)
x1_bounds = (-3, None)

res = linprog(c, A_ub=A, b_ub=b,
        bounds=(x0_bounds, x1_bounds), options={''disp":True})

print(res)
```

The method works and returns the minimum value of the objective function, as well as the slack variables and the values of $\mathbf{x}$ that minimize the objective function. There are some other interesting cases to test: when we have an empty or degenerate feasible region, when the feasible region is (partially) unbounded but the objective decreases in the bounded direction (there is therefore a minimum), the feasible region is unbounded and the objective function decreases in the direction of unbounded-ness, the feasible region is bounded and there is a minimizing segment (there isn't a unique minimizer), and finally the region is unbounded in one direction, but the objective attains a minimum along a ray (there is a ray's worth of minimizers). We consider these cases below.

Case 1: Degenerate feasible region

Consider the following linear program:

$$\text{minimize} - x_0 + 2x_1$$

$$\text{subject to}$$

$$\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

$$x_0, x_1 \geq 0.$$

This corresponds to a degenerate case (the feasible region is a single point). The simplex algorithm from SciPy returns the single feasible point as the solution, which is the origin.

Case 2: Empty feasible region

Consider the following linear program:

$$\text{minimize} - x_0 + 2x_1$$

subject to

$$\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \leq \begin{pmatrix} 0 \\ -1 \end{pmatrix},$$

$$x_0, \; x_1 \geq 0.$$

This linear program defines the empty region to be points in the (x_0, x_1) plane such that $x_0 \leq x_1$ and simultaneously such that $x_1 \geq x_0+1$. These parallel lines never intersect, and there are no points in the plane that satisfy both constraints simultaneously. The program `LP_empty.py` implements this program, and returns an error "Optimization failed. The problem appears to be unbounded."

Case 3: unbounded objective function

We tweak our previous program as follows:

$$\text{minimize } - x_0 - x_1$$

subject to

$$\begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \leq \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

$$x_0, \; x_1 \geq 0.$$

We again get a warning that the problem appears to be unbounded.

4 A little bit of theory

It can be shown (reference?) that all linear programs fall into one of three categories: inconsistent, unbounded, and solvable. We say a linear program is inconsistent if two or more of the inequality constraints are mutually exclusive, e.g. $x_1 \geq 0$ and $2x_1 \leq 0$. In this case, there is clearly no solution to the problem. In the second case, the linear inequality constraints define an unbounded region in Euclidean space and the objective function is also unbounded in the direction that the polytope is unbounded. There is clearly no maximum, since moving in the unbounded direction will increase the objective function without bound. Finally, as long as we have a bounded polytope, we can maximize (or minimize) the objective function. In the next bit, we follow closely the exposition of [**?**]

We will make some transformations to the optimization problem to bring it into another form that provides us with a clear algorithm. Consider (1a) and (1b). We can add slack variables to force the problem to have equality constraints, as in (2). We assume this has been done, and now can write the problem (significantly abusing notation) as

$$\begin{aligned}
&\min\ \mathbf{c}^T\mathbf{x} \\
&\mathbf{A}\mathbf{x} = \mathbf{b} \\
&\mathbf{x} \geq 0.
\end{aligned} \tag{8}$$

We then assume we have a basic feasible solution

$$\mathbf{x} = \begin{pmatrix} \mathbf{B}^{-1}\mathbf{b} \\ \mathbf{0} \end{pmatrix}.$$

The value of the objective function at the BFS is then $\mathbf{c}^T\mathbf{x} = \mathbf{c}_B^T\mathbf{B}^{-1}\mathbf{x}_B =: z_0$, where we've partitioned the cost vector $\mathbf{c}^T = [\mathbf{c}_B^T\ \mathbf{c}_N^T]$ to correspond to basic and nonbasic variables. Then we compute

$$\begin{aligned}
\mathbf{B}^{-1}\mathbf{b} &= \mathbf{B}^{-1}\mathbf{A}\mathbf{x} \\
&= \mathbf{B}^{-1}(\mathbf{B}\mathbf{x}_B + \mathbf{N}\mathbf{x}_N) \\
&= \mathbf{x}_B + \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N.
\end{aligned}$$

Denoting the set of indices of nonbasic variables by R, we can rearrange the last equality less compactly as

$$\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b} - \sum_{j \in R} \mathbf{B}^{-1}\mathbf{a}_j x_j.$$

Under the sum, we have written explicitly the columns of the matrix $\mathbf{A}$ that correspond to nonbasic variables, as well as the elements of $\mathbf{x}$ that that correspond to nonbasic variables.

With this in hand, we can see that the objective function can be written as

$$\begin{aligned}
z &= \mathbf{c}^T\mathbf{x} \\
&= \mathbf{c}_B^T\mathbf{x}_B + \mathbf{c}_N^T\mathbf{x}_N \\
&= \mathbf{c}_B\left(\mathbf{B}^{-1}\mathbf{b} - \sum_{j\in R}\mathbf{B}^{-1}\mathbf{a}_j x_j\right) + \mathbf{c}_N^T\mathbf{x}_N \\
&= \mathbf{c}_B^T\mathbf{B}^{-1}\mathbf{b} - \sum_{j\in R}\mathbf{c}_B\mathbf{B}^{-1}\mathbf{a}_j x_j + \sum_{j\in R} c_j x_j \\
&= z_0 - \sum_{j\in R}(\mathbf{c}_B^T\mathbf{B}^{-1}\mathbf{a}_j - c_j)x_j \\
&= z_0 - \sum_{j\in R}(z_j - c_j)x_j,
\end{aligned}$$

where we have defined $z_j := \mathbf{c}_B^T\mathbf{B}^{-1}\mathbf{a}_j$. In summary, we have cast the LP problem in the following form (in terms of the nonbasic variables):

$$\min z_0 - \sum_{j\in R}(z_j - c_j)x_j \tag{9a}$$

subject to

$$\sum_{j\in R}\mathbf{y}_j x_j + \mathbf{x}_B = \overline{\mathbf{b}}, \quad x_j \ge 0, \quad \mathbf{x}_B \ge 0. \tag{9b}$$

For brevity, we have denoted $\mathbf{y}_j := \mathbf{B}^{-1}\mathbf{a}_j$ and $\overline{\mathbf{b}} := \mathbf{B}^{-1}\mathbf{b}$. Let's take a moment to inspect what we have here. In (9) we can see that we will arrive at a minimum when all of the terms $(z_j - c_j)$ are non-positive. If there is any $j \in R$ such that $(z_j - c_j) > 0$, then we can move in that direction to further decrease the value of the objective function. We will see how this is done next.

Suppose that we have some $k \in R$ such that $(z_k - c_k) > 0$. If there are multiple such k, we take the value of k corresponding to the *most* positive of $(z_j - c_j)$ for $j \in R$. Freezing the problem at this point and setting all other $x_j = 0$ for $j \in R \setminus \{k\}$, we have the simple problem

$$\text{minimize } z = z_0 - (z_k - c_k)x_k \tag{10a}$$

subject to

$$\begin{pmatrix} x_{B_1} \\ x_{B_2} \\ \vdots \\ x_{B_r} \\ \vdots \\ x_{B_m} \end{pmatrix} = \begin{pmatrix} \bar{b}_1 \\ \bar{b}_2 \\ \vdots \\ \bar{b}_r \\ \vdots \\ \bar{b}_m \end{pmatrix} - \begin{pmatrix} y_{1k} \\ y_{2k} \\ \vdots \\ y_{rk} \\ \vdots \\ y_{mk} \end{pmatrix} x_k. \tag{10b}$$

If we have that $y_{ik} \leq 0$, then x_{B_i} will increase as we increase x_k, and so we will not violate the non-negativity constraint on x_{B_i}. On the other hand, if $y_{ik} > 0$, then x_{B_i} will decrease as x_k increases. We do not wish to violate the non-negativity constraints on the elements of x_B, and so we can only increase x_k until we have

$$x_k = \frac{\bar{b}_r}{y_{rk}} \equiv \min_{1 \leq i \leq m} \left\{ \frac{\bar{b}_i}{y_{ik}} \ : \ y_{ik} > 0 \right\}.$$

With this choice for x_k, we then can update (10b) by way of

$$x_{B_i} = \bar{b}_i - \frac{y_{ik}}{y_{rk}} \bar{b}_r \quad i = 1, \ldots, m$$

$$x_k = \frac{\bar{b}_r}{y_{rk}}.$$

Let's put all of this into practice with a problem worked all the way through [2]

5 Geometric interpretation

Include a 2d example solved by hand to make the geometric aspect clear. Use TikZ to show the pictures. Include inconsistent examples and unbounded examples. Include unbounded examples where there is a solution as well as when there isn't a solution (these are easy to cook up).

6 Solvability

7 Modeling problems as LP problems

8 Variations on LP (integer and mixed integer programming)

References

[1] Scipy reference guide, December 2016.

[2] Mokhtar S. Basara, John J. Jarvis, and Hanif D. Sherali. *Linear Programming and Network Flows.* Wiley, second edition, 1990.

[3] https://en.wikipedia.org/wiki/Linear_ programming, 2017.

[4] Thomas McFarland. Example of simplex procedure for a standard linear programming problem, December 2016.