Supplementary Notes on Linear Programming

A linear program is an optimization problem. In an n dimensional space, whose points are described by variables x1, … , x n, we have a "feasible region" which is a "polytope" by which we mean a region whose boundaries are defined by linear constraints.

In two dimensions, a polytope is a polygon, (which can be unbounded and sometimes is, and can even conceivably be empty) In three dimensions a polytope can be envisioned as a polished gem with planar facets.

The standard LP consists of the problem of finding the maximum value of some linear function of the n variables, among points of the feasible region. This amounts to maximizing the dot product of your **r** variable with some vector **c,** which means finding a point in the polytope with maximum component in the direction of **c**.

We use the following terminology. The points of our n dimensional space that obey a single one of our linear constraints as equalities, define a **hyperplane**. In three dimensions a hyperplane is just a plane, and in fact a hyperplane is the generalization of a plane in 3D to higher dimensions. It is a plane-like region of n-1 dimensions in an n dimensional space. A hyperplane that actual forms part of the boundary of the feasible region is called an n-1 face of that region.

The points that obey two constraints lie on the intersection of the hyperplanes defined by them, the n-2 dimensional region of this intersection form part of the boundary of the feasible region they form an n-2 face of it.

And we have similar definitions for the intersection of k of the constraints that form part of the boundary of the feasible region: these are called n-k faces of it.

A **zero face** of the feasible region, also called a vertex of it, is a point which lies on the intersection of n (linearly independent) constraints and is on the boundary of the feasible region. Two vertices are adjacent if n-1 of their defining constraints are the same, and the intersection of the n-1 shared constraints is a 1-face that is called the edge joining them.

The linear function, called the **objective function**, that we seek to maximize in our feasible region defines a direction on our space, and we seek a point that maximizes the component of the position vector r in that direction, In one or two dimensions it is easy to see that there is always such a point that is a vertex of the feasible polytope (assuming that the feasible region is bounded) The same statement is actually true in any dimension: there is always a vertex of the feasible region at which a maximum is achieved.

There are many approaches to "solving" an LP, {which means finding a vertex at which the maximum of the objective function is achieved, by determining all the components of the position vector at that vertex, and the value of the maximum as well.)

The Simplex Algorithm of Dantzig (and perhaps von Neumann) is the classical approach to it. It has been remarkably successful in practice, and its successes contributed greatly to the development of the field of operations research, in which it has been a major tool.

Geometrically, it can be described very simply. You begin at a vertex of the feasible region. You then move to an adjacent vertex across a 1-face (edge) at which the objective function is larger than what it was. This is the basic step. You repeat this step until you are at the maximum point (or, if there is no maximum point you find a direction in which the feasible region is unbounded and the objective function increases as you move in that direction)

In three dimensions you can imagine yourself climbing along the edges of a diamond, always upward, until you reach the top.

So how do we actually do this?

We suppose first that all our constraints are inequalities, which each requires that feasible points lie either on or on one side of a hyper-plane that it defines. The feasible region is then the intersection of the feasible "half-spaces" so defined by each constraint. We also assume, for convenience of our discussion, that all of our variables are constrained to be positive, and that the origin of our coordinates is a vertex of the feasible region. We will relax these assumptions later.

A given point in our n-dimensional space is characterized by its n coordinates, but we can define additional coordinates as well. At each point **r, r**=(x1, … , xn) , for each additional constraint beyond those that require our coordinates to be positive, we can calculate the distance from **r** to that constraint (with a positive sign if it is on the feasible side of the constraint.) If there are m of these additional constraints, we can define n+m variables associated with each point **r** in our space, these being the n original coordinates and the (signed) distances to each of the additional constraints.

The actual LP problem can then be defined by m equality constraints among these n+m variables, as we shall see, and by the objective function..

The initial form of the equation has the original position variables as "basis variables" among the n+m variables, and the m variables that define distances to the other constraints are expressed in terms of these, again as we shall see.

The basic step of the algorithm consists of switching one of these basis variables with one of the other (distance to constraint) variables, so that we have a new set of basis variables that differs by one substitution from the original ones.

What does this basic step have to do with moving from one vertex of the polytope to another?

We associate with each set of basis variables the point at which all of them are set equal to 0. This is initially the origin of our coordinate system, which, by our assumption here is a vertex of our feasible region. When we make a switch to our basis, we switch from the initial origin to the origin of the new basis, The operation of removing one variable from the basis and replacing it by another is called a pivot, and the new origin will be a vertex of the feasible region if the old one was one.

So the simplex algorithm consists of a sequence of pivot operations, and we move from our initial origin to the origin in a sequence of new bases, until we reach a vertex, if any exist, at which the objective function is maximized.

Each pivot moves us across a 1-face of the feasible polytope to a new origin-vertex. We choose ourselves where to move. And we choose always to move to a new vertex at each stage that has a larger value for the objective function than the previous origin-vertex had. Since the objective function keeps increasing at each step, the algorithm cannot cycle, and, since there are only a finite number of possible vertices, it must eventually reach the top value in the polytope, if there is one.

Is this really doable?

Yes. The geometric picture can be translated into an algebraic one with remarkable ease. It is easy and straightforward to find a pair of variables to pivot on, and to perform a pivot, as we shall now see.

So let us now define the problem algebraically.

Our n original variables are x1,…, xn. We refer to a typical variable as xj. We refer to a typical constraint as Ai, which is a vector whose j-th component is Aij.

Our m constraints can be defined by the matrix {Aij} and take the form

$$SUM_j\, Aij\ xj <= bj, \quad \text{for each } i$$

We also have the constraints that our variables are all positive, xj>=0 for each j

Our objective function can be written as the dot product (**c,r**) or SUM_j cj xj.

For each constraint i the distance variable to it is called its "slack" variable, and is usually denoted by si. The i-th constraint then becomes si>=0, where si is defined by the equation

$$si + SUM_j\, Aij\ xj = bi.$$

So we have n+m variables (x's and s's) the constraints are all that these variables are non-negative, and there are m linear equations among them.

Notice that the initial origin will only be feasible when setting the x's to 0 makes the s's all non-negative. This means here that all the b's must be non-negative. In fact, we really want all the b's to be positive, so if any are 0 we add a tiny amount to each to make them all positive.

A pivot consists of picking an x variable, xj0, and an s variable, si0, and interchanging their roles here.

And what does that mean?

Well you use the i0 equation to eliminate xj0 from all other constraints and the objective function, substituting for it according to the i0 equation everywhere. You also divide the i0 equation by Ai0j0 so the coefficient of xj0 in it will be 1 as was the coefficient of si before the pivot.

And that is all there is to a pivot. Which leaves two questions.

What pivot should we perform?
How actually do we do it?

What we are doing with an (si0, xj0) pivot is to move ourselves from the origin along the xj0 axis in the positive direction until we meet the first constraint which will be the i0-th constraint. .

So we want to move along an axis in which moving positively increases the objective function, This means that cj0 should be positive. You can pick any j0 for which this condition holds to pivot with.

Once you choose j0 then i0 is defined by the first constraint that you reach along the 1-face defined by the xj0 axis.

And how?

Well as you increase xj0 for those constraints i for which Aij0 is negative, the left hand side of the constraint decreases as xj0 increases. This means you will never hit the constraint. (you are moving away from it.),

On the other hand you move toward constraints for which Aij0 is positive when you increase xj0. And Aij0 represents the derivative of that constraint with respect to that variable, And bi represents the distance from the origin to that constraint. So the amount you can increase xj0 until you hit the constraint is bi/Aij0;

You must stop increasing xj0 at the first constraint you encounter, which is therefore the i that minimizes bi/Aij0 among those i for which Aij0 is positive, That i will be i0.

And the objective function at the point you reach, which will be the new origin after this pivot, will increase from its previous value by this increase times the derivative of the objective function with respect to xj0, (which is cj0) hence by bi0*cj0/Ai0j0.

Doing pivots by hand is tedious and errors are hard to avoid. Fortunately it easy to do them on a spreadsheet, You need to enter two instructions for each pivot, and do some copying.

Consider the following example:

	Constraints
1:	x1 +3x2 – x3 + x4 <= 2
1	-x1 +x2 +2x3 +x4 <=1
2	x1 - 2x2 +x2 –x4 <=1

Maximize x1+x2+x3+x4

s

We set up a "tableau" with a column for each variable including the three slack variables, and also for the –b's which we get by moving the right hand sides here to the left side. The tableau for this LP then looks like.

s1	s2	s3	x1	x2	x3	x4	-b	
1	0	0	1	3	-1	1	-2	first equation
0	1	0	-1	1	2	1	-1	second
0	0	1	1	-2	1	-1	-1	third
0	0	0	1	1	1	1	0	objective function

.

At this point you can pivot on any xj since all coefficients in the objective function are positive. If you choose x1 you must pivot on the third equation, and the objective function will increase by 1. If you choose x2 you must pivot on the first equation, and the objective function will increase by 2/3. If you choose x3 you must pivot on the second, and the objective function will increase by ½. If you choose x4 you must pivot on the second, and the objective function will increase by 1.

We have here a 4 by 8 matrix and suppose its upper left entry is in box A4 of the spreadsheet.
Suppose we decide to pivot on row 2 and column x4 which would be the entry in g5.
You could then enter in say a11: = a4 –a$5*$g4/g5 and copy it into the 4 by 8 area of which it is the upper left corner.
This will eliminate x4 from all the equations, but the second, and it will entirely obliterate the second equation. To bring it back you enter into a12: =a5/$g5 and copy that across row 12.
That completes the pivot.
You now have a new tableau and can pivot again. To do so you can copy the entry in a11 into a18 and change the indices in it that have dollar signs to reflect the new pivot and then copy it.

When are you done?

When every coefficient in the objective function (except the constant term in the –b column) is non-positive, you are at the top and have the solution.

What else can happen?

You could end with a positive coefficient cj in the objective function and every Aij in its column is negative. This means increasing xj moves away from all constraints. You can therefore move on forever, always increasing the objective function, without violating any constraint. This is unbounded.

And what happens when you are at the top?
The value of the objective function at the solution will be the constant term (in the –b column) in the objective function.

The value of thecurrent basis variables will all be 0,
The value of each of the other variables will be the corresponding b in the equation in which it occurs,

Try doing this and see what happens.

This is essentially the simplex algorithm, and this is all there is to it.

However there are several technical points as follows.

1,. What do you do when you have a variable that is not constrained to be positive?

You can find a constraint to pivot on and take it out of the basis. Then leave it alone and do not consider pivoting on that row. This is because setting this variable to 0 does not represent a constraint so it should never be a basis variable.

2. What do you do when you have an equality constraint?
Then there is no point in introducing a slack variable for it. You can change one of your variables into a slack variable by using the equation to remove that variable from all the other equations and the objective function. (This could however make some of the other b's negative)

3, What do you do if some b's are 0?
Just change them slightly to be very small numbers. This will prevent pivoting in circles.

4, What do you do if your origin is not initially feasible? (that is, one or more b's are negative,)
This is kind of fun. We create a new origin that is feasible.
How?
We introduce a new dimension with coordinate xnew. The old world represents the hyperplane xnew=1.
The new origin occurs where xnew=0.
And how do we do this?
We can add terms di(xnew-1) to the left hand side of any i equation for which bi is negative. This will increase bi by di and if we choose di such that bi+di =1 (or anything else positive) for each such i all the b's will become positive and our new origin will be feasible.

Notice that when xnew=1 these added terms all disappear and the equations have the same content as before.

We also add a constraint: xnew<=1 and we add a new objective function, xnew.

So now we have added a new variable and a new constraint and a new objective function. This may seem strange but there is no harm in having two objective functions.

One strategy is to maximize the new objective function, xnew, first. If the maximum occurs at xnew =1, then obviously xnew is no longer in the basis, so you can ignore its equation, and forget it and proceed to maximize the original objective function.

It can happen that the maximum of this new objective function is less than 1. In that case there is no real world at xnew =1, which means that there are no feasible points at all.

There is another strategy that involves pivoting on linear combinations of the two objective functions, that is kind of fun, but we will not go into it here.

The short answer to the last question is: finding a feasible origin involves solving a modified LP starting from a feasible origin.

We first give an example for making the origin feasible when it is not.

So here is an LP for which the origin in the x variables is not feasible, because it violates two constraints.

3x1+ 2x2 –x3 <= 1.
-x1+ 2x2 +x3 <= -1.
x1 - x2 + x3 <= -2
maximize x1+x2+x3
all x's must be positive

We add a new variable, x4 and a new constraint,x4=<1, and add 2(x4-1) to the left side of constraint 2 and 3(x4-1) to the left side of constraint 3. We also require x4 to be non-negative, and make x4 our new objective function.

Notice that when x4 =1 happens, if it does, x4 will be out of the basis, and so will appear in only one equation. When x4=1 happens, the added terms are all 0s so that we are back to our old problem.

So our new constraints are

$$3x1+ 2x2 –x3 <= 1.$$
$$-x1+ 2x2 +x3 +2x4 <= 1.$$
$$x1 - x2 + x3 +3x4 <= 1$$
$$x4 <= 1$$

and we have two objective functions, x4 and x1 + x2 + x3. We use the x4 objective function first. We first pivot on the x4 column and row 3 (which increases that objective function to 1/3.) After a second pivot, we find that the maximum of x4 occurs at .375, which means there are no feasible points with x4=1 and hence the feasible region is empty.

If we change the problem to

$$3x1 - x2 – 2x3 <= 2.$$

$$2x1+ 2x2 +x3 <= 10.$$
$$x1 - x2 + x3/10 <= -2$$
$$(\text{or } x1 – x2 +x3/10 +3x4 <= 1\_$$
$$x4 <= 1$$

The following sequence of pivots seems to give a feasible answer.

s1	s2	s3	s4	x1	x2	x3	x4	b	
1	0	0	0	1	-1	-2	0	-2	
0	1	0	0	2	1	1	0	-10	
0	0	1	0	1	-1	0.1	**3**	-1	
0	0	0	1	0	0	0	1	-1	
				1	1	1	0		of1
							1		of2
1	0	0	0	1	-1	-2	0	-2	
0	1	0	0	2	**1**	1	0	-10	
0	0	0.333333	0	0.333333	-0.33333	0.033333	1	-0.33333	
0	0	-0.33333	1	-0.33333	**0.333333**	-0.03333	0	-0.66667	
0	0	0	0	1	1	1	0	0	
0	0	-0.33333	0	-0.33333	0.333333	-0.03333	0	0.333333	
1	0	-1	3	0	0	-2.1	0	-4	
0	1	1	-3	3	0	1.1	0	-8	
0	0	0	1	0	0	0	1	-1	x
0	0	-1	3	-1	1	-0.1	0	-2	
0	0	1	-3	2	0	1.1	0	2	
0	0	0	-1	0	0	0	0	1	x
			x				x		

That is, if you omit the rows and columns marked with an x, you find that the origin is feasible and you can pivot to a solution. The columns to be omitted are those of x4 and the slack variable s4, and the rows are those of the x4+s4=1 equation and that of the second objective function.

Duality

If you look at the pivot operation, you see that the step of replacing Aij by Aij-Ai0j*Aij0/Ai0j0 is symmetric on transposing the A matrix. (and –b is like another column

of it and c another row.) This change describes how the rows and columns other than the i0 row and j0 column change under the pivot operation.

The change that takes place to the i0 row and to the columns that are involved in the pivot is not exactly symmetric but is almost so. The i0 row gets divided by Ai0j0. The j0 column gets changed into all 0's except for 1 in the i0 row. But the column for the variable traded for it, which used to have this exact form, now has entries given by the old Aj0 column divided by minus Ai0j0. If we ignore the position of this column, we see that in content it gets divided by –Ai0j0 while the row got divided by +Ai0j0.

All this means that there is a symmetry that involves transposing the A matrix, but it is not the mere transpose. You have to transpose the A matrix and also reverse its sign. If you reverse the sign and transpose the equation Aij0 new = Aij0/Ai0j0 you get Ai0j new = -

Ai0j/Ai0j0 (the sign is changed because there are two A's on the right which product doesn't change sign, and only one on the left. Because there is an odd number of A's in every term of the other transformation, the change in sign does not appear: everything changes sign.

So an i0j0 pivot has the same effect as far as content of the matrix is concerned as transposing A (which switches b to c) and reversing the sign of its elements (so b ends up as –c) and making a j0i0 pivot.

We usually describe this transposed problem by keeping the sign of A fixed. We can accomplish this by multiplying the matrix for the transposed problem by -1. This has the effect of switching less than or equal with greater than or equal, and switching maximizing to minimizing. (maximizing –b is minimizing b)

The transposed problem that pivots the same way as the original LP is called the dual to it, while the original is called the primal problem. (Though if you start with the dual, the primal is dual to it.)

After multiplying by -1 we find that if the primal problem is to maximize c.x subject to Ai.x<=bi, subject to xj>=0, the dual is to minimize b.y subject to Sum yiAij >=cj and yi>=0 for each i.

There are two simple but remarkable features about the dual. First, it not only changes under a pivot exactly as the primal changes, but the condition for it to be a solution is identical to that for the primal. And the value of its optimum at the solution is the same for both problems.

Recall that for the primal problem you are at a solution when the b's are all non-negative, and the c's are non-positive. Similarly, you are feasible for the dual when the c's are all non-positive, and at the solution then when you also have the b's all non-negative. (which implies that making any combination of your current y basis variables positive cannot decrease your dual objective function)

This means that when you have found a solution for either problem you can read off a solution for the other. (for the primal the values of the basis variables are all 0 and the others are the b's of the equations they appear in. for the dual, the values of the dual variables corresponding to the basis variables are given by the –c's in their columns, while those corresponding to the non-basis variables are all 0.

(Notice that if you are feasible for the primal but not at a maximizing vertex, the dual problem corresponding to where you are is not feasible. But so what?)

What dual variables correspond to what?

For each xj there is a dual slack variable, tj and for each si there is a dual variable yi.

The second wonderful property is derivable by evaluating the sum over i and j of yiAijxj

Recall that the primal problem, after the sj slack variables were introduced obeyed the equations Sum Aijxj + si = bi The dual variables with similar slack variables tj (which go on the other side because the inequalities are reversed, obey Sum yjAij = tj + cj

Putting these together, we can evaluate Sum yiAijxj either as Sum biyi – Sum siyi or as Sum tjxj + Sum cjxj. All of this tells us

Sum biyi (which is the dual objective function) = Sum cjxj (the primal objective function) + Sum_i si yi + Sum_j tjxj

Notice that if the xj define a feasible point for the primal then the si and xj are non-negative for all arguments i or j. And if the yi define a feasible point for the dual, the tj and yi are non-negative as well. This means the last two sums on the right here are non-negative When the x variables are feasible and the y variables are feasible.

This tells us that the value of the dual objective function at any feasible point is greater than the value of the primal objective function at any feasible point, And it is greater by the two last sums above. The fact that the primal and dual solution optimizing values are the same then tells us that at least one of each pair si and yi , and each pair xj and tj must be 0 at the solutions optimizing points.

We have not really talked about variables that do not have positivity constraints, and about constraints that are equalities rather than inequalities. When a constraint is an equality, its slack variable is always 0. Thus the fact that its dual can be negative changes nothing in the discussion here. Their product is always 0.

There are many problems that can be described by linear programs and the equality of primal and dual objective functions at their solution turns out to tell us interesting things about the problem. For example, the Philip Hall marriage theorem is a duality theorem, as is the more general "maximum flow = minimum cut" statement about network flows.