CS 435 : Linear Optimization Fall 2008

Lecture 11: The Simplex Algorithm

Lecturer: *Sundar Vishwanathan*
Computer Science & Engineering Indian Institute of Technology, Bombay

We saw last time that to solve the Linear Programming problem, it was enough to find an extreme point where the cost is maximised amongst all extreme points. The so called **Simplex Algorithm** considers extreme points in a certain order. It visits extreme points one by one. It may not visit all extreme points of the polytope but when it terminates, it will find the optimum. This, we will prove.

The algorithm consists of the following two steps.

1. Start at an extreme point.
2. Move to a *neighbouring* extreme point of greater cost if one exists and repeat this step. If no such neighbour exists then exit with this point as the optimum point.

We begin with some questions.

1. How do we identify the *first* extreme point?
2. What do we mean by neighbouring extreme points? How do we find them?
3. Is the algorithm correct? That is, when it terminates do we have an optimum?
4. How many iterations does this algorithm perform before stopping, in the worst case?

Answering these questions will be our next task. We will leave the first as an exercise (this is not easy). We begin with the second.

1 The notion of directions

Let us consider an example in two dimensions.

Example 1 Consider two half-planes in 2D given by

$$\begin{aligned} 3x + 4y &\leq 12 \\ 7x + 3y &\leq 6 \\ x &\leq 0 \\ -y &\leq 0 \end{aligned}$$

One of the extreme points is $(0, 0)$. Another is $(0, 2)$. Consider the third extreme point which is the intersection of the two lines defining the border of the first two half-planes. We would like to say that $(0, 2)$ is a neighbour of this point. The coordinates of the third point, say (x, y), is the solution to:

$$\begin{bmatrix} 7 & 3 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 6 \\ 12 \end{bmatrix}$$

We will refer to this 2×2 matrix as A below.

The directions of potential neighbouring extreme points are the vectors $\begin{bmatrix} 4 \\ -3 \end{bmatrix}$ and $\begin{bmatrix} -3 \\ -7 \end{bmatrix}$, respectively. Why? How did we compute these? One way to define a direction is as a difference between two vectors. Can you think of other ways?

If we write v_1 and v_2 as columns of a matrix B, $\begin{bmatrix} -3 & -7 \\ 4 & -3 \end{bmatrix}$ then we observe that AB is a diagonal matrix with negative entries on the diagonal.

Consider a corner p of the unit cube in three dimensions. We see that its neighbours are the three corners you get by walking along the three edges which meet at p. Note that each edge is the intersection of two of the three planes intersecting at p. Let the bottom face of the cube be labelled $1, 2, 3, 4$ in the clockwise direction and the top face $5, 6, 7, 8$ with 5 on top of 1. Then the neighbours of 7 are $6, 8$ and 3.

We generalize this notion to n dimensions. Consider the usual linear optimization problem with constraints $Ax \leq b$ and cost function $c^{\intercal}x$. Consider an extreme points x_0. The point x_0 satisfies

$$A'x_0 = \mathbf{b}'A^{''}x_0 < \mathbf{b}^{''}. \tag{1}$$

As before A' is the matrix formed by some n linearly independent rows of the matrix A and b' the corresponding entries from b. These rows are satisfied by x_0 with equality. The other rows constitute $A^{''}$. In a sense, the first set of equalities define the extreme point. These equalities are said to be the *support* of the extreme point.

How do we determine the neighbours of x_0?

By the previous example, it looks like the neighbours share $n - 1$ hyperplanes in their support. Then, to find one neighbour, it makes sense to remove one of the hyperplanes from A', add one from $A^{''}$ and check if the resultant point is feasible. The points we get thus, which are feasible, are the neighbours.

Exercise: Prove that for each hyperplane in A', our assumptions imply that there will be exactly one such hyperplane in $A^{''}$, for which the resultant point will be feasible.

A procedure to find the neighbours is obvious from the above description. The time taken is $O(mn)$ gaussian eliminations.

We wish to make this process faster. The idea is simple when one is told of it. There are n rays emanating from x_0 which connect it to its neighbours. The idea is to determine the vectors corresponding to these directions. As we shall see this can be determined rapidly, and once we have the directions, we can also determine the neighbours efficiently. I recommend you try the latter problem before we give a solution. We proceed with the former.

Consider a point x_i on a line l_i passing through x_0 which connects it to some neighbour. This line is determined by the intersection of some $n - 1$ rows out of the n linearly independent rows of A'. That is, any point in this line satisfies $n - 1$ of these inequalities with equality and one with strict inequality. Assume that the ith one is the strict inequality. In other words (of a mathematician),

$$A'_j x_j = \mathbf{b}'_j \quad 1 \leq j \leq n, j \neq i \tag{2}$$
$$A'_i x_i < \mathbf{b}'_i. \tag{3}$$

The direction of the vector along the line l_i from the point x_0 towards x_i is $x_i - x_0$.

What can we say about $A'(x_i - x_0)$? For all rows of A' except the ith row, the dot product of this row with x_i and x_0 is the same (look at the assumptions above.) Also, $A'x_i < \mathbf{b}'_i$ and $A'x_0 = \mathbf{b}'_i$. Hence $A'(x_i - x_0)$ is a vector with zeroes everywhere except in the ith position which contains a strictly negative value. The constant at the ith position depends on where on the line we picked x_i. The farther it is from x_0, the greater the magnitude. Let us pick it so that $A'(x_i - x_0) = -e^i$, the vector with zeroes everywhere and -1 in the ith position. This offers a further clue which we expand and expound on below.

Consider a matrix Z, having as columns, vectors $(x_i - x_0)$, for $i = 1, 2, ..n$.

$$Z = \begin{bmatrix} x_1 - x_0, & x_2 - x_0, & \ldots, & x_n - x_0 \end{bmatrix}$$

What can you say about $A'Z$?

$$A'Z = \begin{bmatrix} A'x_1 - A'x_0, & A'x_2 - A'x_0, & \ldots, & A'x_n - A'x_0 \end{bmatrix} \tag{4}$$

The above properties of x_i imply that:

$$(A'x_i)_j = (A'x_0)_j; \quad j \neq i, \tag{5}$$

$$(A'x_i)_i - (A'x_0)_i = -1. \tag{6}$$

In other words, for each i, the i^{th} column of the matrix $A'Z$ has a -1 in the i^{th} position and zeroes in all other positions. Putting it in the language of matrices:

$$A'Z = -I. \tag{7}$$

Now, what can we say about the direction vectors of the lines from x_0 in terms of A'?

THEOREM 1 *The direction vectors are the columns of the negative of the inverse of matrix A'.*

How much time does it take to find the direction vectors?

2 Finding neighbouring points with greater cost

The iterative step in the simplex algorithm says "move to a neighbour with greater cost". Once we know the direction of each neighbour, we first determine which of the neighbours has greater cost. Note that if one of the neighbours x_i has greater cost then every point on the line segment joining x_i and x_0 has cost more than that of x_0. To slow down the exposition, we provide a proof. But you should try this yourself before you read it. PROOF: Let x' be any arbitrary point on the line segment joining x_0 and x_i such that $x' \neq x_0$. Recall that any point on the line segment joining x_0 and x_i can be written as a convex combination of these two. So,

$$x' = \lambda x_i + (1-\lambda)x_0, \tag{8}$$

where $\lambda > 0$. Now we employ an old trick. The value of a linear function at a convex combination of two points is a suitable weighted average of the two.

$$c^{\mathsf{T}}x' = \lambda c^{\mathsf{T}}x_i + (1-\lambda)c^{\mathsf{T}}x_0 \tag{9}$$

or,

$$c^{\mathsf{T}}x' = \lambda(c^{\mathsf{T}}x_i - c^{\mathsf{T}}x_0) + c^{\mathsf{T}}x_0 \tag{10}$$

And since $c^{\mathsf{T}}x_i > c^{\mathsf{T}}x_0$, $c^{\mathsf{T}}x' > c^{\mathsf{T}}x_0$. □

So we first check if x_i has greater cost than x_0. Or equivalently check if $c^T(x_i - x_0) > 0$. After having ascertained that moving in a particular direction increases cost, we need to actually find the neighbouring point in that direction. Given the direction vector v_i from x_0, any point x' on the line joining these two can be written as

$$x' = x_0 + tv_i \quad t \geq 0. \tag{11}$$

When $t = 0$, this gives x_0 and as t increases this point moves further from x_0. For small values of t, this point will be feasible and as we increase t there will come a time when it will no longer be feasible.

We hence seek the maximum value of t such that $x_0 + tv_i$ is feasible. How do we determine this?

We need the maximum t such that

$$A(x_0 + tv_i) \leq \mathbf{b} \tag{12}$$

Notice that the point $x' = x_0 + tv_i$ satisfies

$$A'_jx' = b'_j, \quad \forall j \neq i, \quad 1 \leq j \leq n // A'_ix' < b'_i \tag{13}$$

Hence we need to only focus on the inequalities in A''.

We increase the value of t gradually until one of the other inequalities, say the kth, becomes an equality. That is,

$$A''_k x' = b_k \tag{14}$$

This x' will be the required neighbour.
Argue that under the assumptions we have made, exactly one of the other inequalities will become an equality, so that the resultant point is feasible. How do we find this k? Algorithmically, we determine the intersection of the line with each of the hyperplanes in A'' and choose the point closest to x_0. The others will be infeasible (why?). In other words, consider each row in A'' in turn and see which yields the smallest value of t.

Thus,

$$t_k = min_s \frac{b_s - A_s x_0}{A_s v_i}, \tag{15}$$

where s ranges over all those rows of A which do not belong to A' such that $A_s v_i > 0$.

Question: What happens when $A_s v_i < 0$? Check that algebraically we can ignore these rows. Geometrically, what can you say about the point of intersection of such a hyperplane with the line? Explain with a figure.

How much time does it take to find a neighbouring point of greater cost if one exists?