Lecture 1

Solving Linear Programs with the Ellipsoid Algorithm

November 12, 2004
Lecturer: Kamal Jain
Notes: Chris Ré

1.1 Overview

Recall from last time:

Given a convext set C, $C \subseteq EV(E) = R$, a ball $B(\delta, \epsilon) \subset C$ and a polytime separation oracle then we can find a point inside C in polynomial time. This is polynomial n and $log(\frac{1}{\epsilon})$.

Remark. The existence of a ball is necessary because of the perturbation of the center of the ellipsoid.

Our goal today is to solve a linear program in weakly polynomial time. Concretely given $Ax \leq b$ want to see if there exists a feasible x. We will be assuming the LP is explicitly given.

Remark. We can get the separating hyperplane required of the oracle by subsituting the point x into $Ax \leq b$ and finding which constraints are violated. These form the separating hyperplane.

Notation $Size(\frac{p}{q}) = log(|P| + 2) + log(|Q| + 2)$.
$Size(A) = \sum_{ij} Size(a_{ij})$
$Size(A, b) = Size(A) + Size(b)$.

We will find an algorithm polynomial in $Size(A, b)$.

It is easy to see that $Size(det(A)) \leq nlog(n)Size(A)$. We can just sum the columns. The important thing to note is that the determinant of a matrix is polynomially sized in the matrix size. Below we will show that there is an easily describable solution - that is a solution which is polynomially sized as above.

Lemma 1.1. *$Ax \leq b$ is feasible $\leftrightarrow$ $Ax \leq b$ and $\forall i$ $|x_i| \leq R_0$ with $R = 2^{log(A,B)}$*

If the LHS is infeasible, it is clear that the right-hand side is also infeasible.
If the LHS is feasible, we need to do some work. Notice, that if there is a corner point we can reach our conlcusion because it is of small size. However, corner points don't always exist. Just consider $x + y <= 2$. We need to generalize this in some way .

Definition 1.1 (face). A *face* is a hyperplane described by H,c i.e. $Hx = c$. Then we define $face(H, c) = \{x|x$ s.t. $Hx = c$ and $Ax \leq c$. Further this set must be *minimal* with respect to inclusion and non-trivial.

It is the set intersection of the solution sets of $Hx = c$ and $Ax \leq c$. It should be clear that faces always exist.

Proof. Now we assume the LHS is feasible. Consider $\forall j \in S\ Ax_j = b$ and there is a solution to the LP. Choose such a maximal such S. Notice we can find S by considering the inequalities in any order. Now, we want to show that we only need to consider the equality conditions for a small solution.

Assume that z satisifies A_S, the set of tight equalities, but $\exists j \notin S$ that violates the inequality. That is $\sum_i a_{ij} z > b_j$.

We know that there exists some feasible solution, by our selection of S, call this solution y. Now we draw the line y to z. Move along this line, we must cross into the set. At this point a new constraint will have gone tight, this contradicts the maximality of S. Therefore, we do not need to concern ourselves with the inequalities when dealing with a maximal set S. Notice that the same procedure can get started by picking two points one on the exterior and one on the interior.

This means there is a small solution. Let A_S be the matrix of equality constraints, zeroing out the constraints to make it square. $A_S x = b_s$, and x. By cramer's rule, we get that $x = {A_S}^{-1} b_s = \frac{1}{det(A_S)} A^T b_s$. We know that the determinant is poly sized, so we have a small sized solution. ☐

To run the ellipsoid algorithm, we'd like to get a polynomially sized ball of feasibility around x. This ball is because our running time bound for the ellipsoid algorithm depends on a volume bound. Our idea is that we can create a new program that is feasible iff the original one is. This new program will be formed by relaxing constraints in each direction. Our plan is to show that relaxing a sufficiently small ball around the constraints (the b_js) has our property. We will then use this fact, to show that there is such a set.

Lemma 1.2. *$Ax \leq b$ is feasible $\leftrightarrow \exists \epsilon_0$ such that $\forall \epsilon < \epsilon_0 Ax \leq b + \epsilon$.*

Proof. It is clear that a feasible on the LHS is a feasible point on the right hand side. We need to show that if the LHS is infeasible, we can choose the poly sized ball such that the other program remains infeasible.

Now if $Ax \leq b$ is infeasible using the completness of Farkas's Lemma, we get a y such that: $y^T Ax = 0$ and $y^T b = -1$. By direct calulation, y is a small certificate i.e. $|y| \leq 2^{Poly(Size(A,b))}$. Notice now that $y^T(b + \epsilon) \leq 0$ because we can choose ϵ such that $y^T \epsilon \leq 1 - y^T b$. ☐

Lemma 1.3. *$Ax \leq b \leftrightarrow \exists \epsilon_0\ A(x + \epsilon_0) \leq b + \epsilon_1$*

Proof. Direct calculation, we get to choose ϵ_0. Choose it so that $A\epsilon_0 \leq \epsilon_1$, this is easy to do by continuity. Applying the cramer's rule style argument above, we see that ϵ_0 is poly sized. ☐

We now have a program, namely the second one which has a ϵ ball around the solution. The ball has volume since it is of full dimension, and thus the ellipsoid algorithm can be used.

1.1.1 Total Unimodular Functions

Most of the times LPs arise by relaxing integrality constraints. Here we are concerned with programs of the form.

$Ax \leq b\ \ x \in \{0, 1, -1\}$

Definition 1.2. Unimodular A matrix A is called *Unimodular* if $det(A) = \pm 1$

Definition 1.3. Total Unimodular A matrix A called is *Total Unimodular* if every submatrix of A is unimodular

What we will like to show is that any optimal solution to the program can be converted to an integral solution.

An example of Total Unimodular matricies are when we write a directed graph G as a matrix using the following rule. $\forall (u, v) = e_i \in E(G) \to a_{ui} = -1 a_{vi} = 1$.

Proof. Proceed by induction, consider matricies of size $n \times n$. Consider a column, this corresponds to an edge.

Case 1 No portion of this edge is remaining in this sub-matrix. In other words, the nodes that correspond to rows are not incident to this edge. In this case the whole column consists of 0s, therefore the determinat of this matrix is 0.

Case 2 There is one row in the calculation, corresponding to a node incident to the considered edge (column). The induction hypothesis tells us the sub matrix has determinant in $\{0, \pm 1\}$. So multiplying by ± 1 keeps us in $\{0, \pm 1\}$.

Case 3 It must be the case that all edges/nodes are present, else we apply one of the first two cases. In this case, we can add two rows not affecting the determinant to get a column of 0s. Therefore the determinant of this square matrix is 0.

□

1.1.2 Fractional Flow

We are talking here about the integrality of the flow in a max-flow/min-cut style algorithm. There are two ways to see this, one inline with the above is to examine the dual created above. The second way inline with the ford-fulkerson algorithm is to look at the residual graph.