The Ellipsoid (Kachiyan) Method

Yinyu Ye
Department of Management Science and Engineering
Stanford University
Stanford, CA 94305, U.S.A.

http://www.stanford.edu/ yyye

Basic Idea

The basic ideas of the ellipsoid method stem from research done in the nineteen sixties and seventies mainly in the Soviet Union (as it was then called) by others who preceded Khachiyan. The idea in a nutshell is to enclose the region of interest in each member of a sequence of ellipsoids whose size is decreasing.

The significant contribution of Khachiyan was to demonstrate in two papers—published in 1979 and 1980—that under certain assumptions, the ellipsoid method constitutes a polynomially bounded algorithm for linear programming.

Linear Feasibility Problem

The method discussed here is really aimed at finding an element of a polyhedral set Y given by a system of linear inequalities.

$$Y = \{y \in R^m : a_j^T y \le c_j, \quad j = 1, \ldots n\}$$

Finding an element of Y can be thought of as being equivalent to solving a linear programming problem, though this requires a bit of discussion.

Two important assumptions

(A1) There is a vector $y_0 \in R^m$ and a scalar $r > 0$ such that the closed ball $S(y_0, R)$ with center y_0 and radius R, that is

$$\{y \in R^m : \|y - y_0\| \leq R\}$$

contains Y.

(A2) There is a known scalar $r > 0$ such that if Y is nonempty, then it contains a ball of the form $S(y^*, r)$ with center at y^* and radius r.

Note that this assumption implies that if Y is nonempty, then it has a nonempty interior.

Ellipsoid Representation

Ellipsoids are just sets of the form

$$E = \{y \in R^m : (y - z)^T B(y - z) \le 1\}$$

where $z \in R^m$ is a given point (called the center) and B is a symmetric positive definite matrix of dimension m. We can use the notation ell(z, B) to specify the ellipsoid E defined above.

Affine Transformation

Let us assume that $E_k = \text{ell}(y_k, B_k^{-1})$, where the positive definite matrix B_k has the factorization $B_k = J_k J_k^{\mathrm{T}}$. Now consider the transformation $y \mapsto y_k + J_k z$.

Let $y \in E_k$. Then $y - y_k = J_k z$ for some vector $z \in R^m$. Now since $y \in E_k$,

$$\begin{aligned} 1 &\geq (y - y_k)^T B_k^{-1} (y - y_k) \\ &= (J_k z)^T (J_k J_k^{\mathrm{T}})^{-1} (J_k z) \\ &= z^T J_k^{\mathrm{T}} (J_k^{\mathrm{T}})^{-1} J_k^{-1} J_k z \\ &= z^T z \end{aligned}$$

so $z \in S(0, 1)$. Conversely, every such point maps to an element of E_k.

Cutting Plane

$$\text{vol}(E_k) = (\det B_k)^{1/2}\text{vol}(S(0,1)).$$

At each iteration of the algorithm, we will have $Y \subset E_k$. It is then possible to check whether $y_k \in Y$. If so, we have found an element of Y as required. If not, there is at least one constraint that is violated. Suppose $a_j^T y_k > c_j$. Then

$$Y \subset \tfrac{1}{2}E_k := \{y \in E_k : a_j^T y \le a_j^T y_k\}$$

This set is “half the ellipsoid” cut through its center.

New Containing Ellipsoid

The successor ellipsoid E_{k+1} is constructed as follows. Define

$$\tau = \frac{1}{m+1}, \quad \delta = \frac{m^2}{m^2-1}, \quad \sigma = 2\tau.$$

Let

$$y_{k+1} = y_k - \frac{\tau}{(a_j^{\mathrm{T}} B_k a_j)^{1/2}} B_k a_j, \quad B_{k+1} = \delta \left(B_k - \sigma \frac{B_k a_j a_j^{\mathrm{T}} B_k}{a_j^{\mathrm{T}} B_k a_j} \right).$$

Theorem 1 *The ellipsoid $E_{k+1} = ell(y_{k+1}, B_{k+1}^{-1})$ defined as above is the ellipsoid of least volume containing $\frac{1}{2}E_k$. Moreover,*

$$\frac{\mathit{vol} E_{k+1}}{\mathit{vol} E_k} = \left(\frac{m^2}{m^2-1}\right)^{(m-1)/2} \frac{m}{m+1} < \exp\left(-\frac{1}{2(m+1)}\right) < 1.$$

The Ellipsoid Algorithm

Input: $A \in R^{m \times n}$ $c \in R^n$, $y_0 \in R^m$ such that Y (as defined) satisfies (A1) and (A2).

Output: $y \in Y$ or determination that $Y = \emptyset$.

Initialization: Set $B_0 = r^2 I,\ K = \lceil 2m(m+1)\log(R/r) \rceil + 1.$

For $k = 0, 1, \ldots, K-1$ do

Iteration k: If $y_k \in Y$, STOP: result is $y = y_k$. Otherwise, choose j with $a_j^T y_k > c_j$. Update y_k and B_k (as defined).

STOP: result is $Y = \emptyset$.

Performance of the Ellipsoid Method

Under the assumptions stated above, the ellipsoid method solves linear programs in a polynomially bounded number of iterations. Bertismas and Tsitsiklis give that bound as $O(m^4 \log(R/r))$.

Computational experience shows that the number of iterations required to solve a linear programming problem is very close to the theoretical upper bound. This means that the method is inefficient in a practical sense. In contrast to this, although the simplex method is known to exhibit exponential behavior on specially constructed problems such as those of Klee and Minty, it normally requires a number of iterations that is a small multiple of the number of linear equations in the standard form of the problem.

LP and LFP

$$
(\mathrm{P}) \qquad \begin{array}{ll} \text{maximize} & c^T x \\ \text{subject to} & Ax \leq b \\ & x \geq 0 \end{array}
$$

$$
(\mathrm{D}) \qquad \begin{array}{ll} \text{minimize} & b^T y \\ \text{subject to} & A^T y \geq c \\ & y \geq 0 \end{array}
$$

$$
c^T x \leq b^T y.
$$

Integer Data

Next, we assume that the data for the problem are all integers. As a measure of the size of the problem above we let $c_j = a_{0j}$ and define

$$L = \sum_{i=0}^{m} \sum_{j=1}^{n} \lceil \log_2(|a_{ij}| + 1) + 1 \rceil .$$

In our discussion above, we made two assumptions about Y. One of the assumptions, (A2), effectively says that if Y is nonempty, then it possesses a nonempty interior. The linear inequalities are relaxed to

$$a_j^{\mathrm{T}} y < c_j + 2^{-L} \quad j = 1, \ldots, n. \tag{1}$$

It was shown by Gács and Lovasz (1981) that if the inequality system (1) has a solution, then so does

$$a_j^{\mathrm{T}} y \le c_j, \quad j = 1, \ldots n.$$

Bounds from L

Therefore, we can bound

$$r \geq 2^{-L}.$$

On the other hand, we can bound

$$R \leq O(2^{L}).$$

Thus,

$$\log(R/r) \leq O(L),$$

which is linear (polynomial) in L.

The sliding objective hyperplane method

Consider

$$
\begin{array}{lll}
 & \text{minimize} & b^T y \\
(\mathrm{D}) & \text{subject to} & A^T y \geq c \\
 & & y \geq 0
\end{array}
$$

At the center, y^k, of the ellipsoid, if a constraint is violated then add the corresponding constraint hyperplane as the cut; otherwise, add objective hyperplane $b^T y \geq b^T y^k$ as the cut.

Desired Theoretical Properties

- Separation Problem: Either decide $x \in P$ or find a vector d such that $d^T x \le d^T y$ for all $y \in P$.
- Oracle to generate d without enumerating all hyperplanes.

Theorem 2 *If the separating (oracle) problem can be solved in polynomial time of* m *and* $\log(R/r)$*, then we can solve the standard linear programming problem whose running time is polynomial in* m *and* $\log(R/r)$ *that is independent of* n*.*

Linear programing when n is exponential large: TSP

Given an undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ where $\mathcal{N}$ is the set of n nodes and length c_e for every edge $e \in \mathcal{E}$, the goal is find a tour (a cycle that visits all nodes) of minimal length. To model the problem, we define for every edge e a variable x_e, which is 1 if e is in the tour and 0 otherwise.

Let $\delta(i)$ be the set of edges incident to node i, then

$$\sum_{e \in \delta(i)} x_e = 2, \ \forall i \in \mathcal{N}.$$

Let $S \subset \mathcal{N}$ and

$$\delta(S) = \{e : \ e = (i, j), \ i \in S, \ j \notin S\}.$$

Then,

$$\sum_{e \in \delta(S)} x_e \geq 2, \ \forall S \subset \mathcal{N}, \ S \neq \emptyset, \mathcal{N}.$$

LP relaxatrion of TSP

$$
\text{(TSP)} \quad \begin{array}{ll} \text{minimize} & \sum_{e\in\mathcal{E}} c_e x_e \\ \text{subject to} & \sum_{e\in\delta(i)} x_e = 2, \ \forall i \in \mathcal{N}, \\ & \sum_{e\in\delta(S)} x_e \geq 2, \ \forall S \subset \mathcal{N}, \ S \neq \emptyset, \mathcal{N}, \\ & 0 \leq x_e \leq 1 \, \forall e \in \mathcal{E}. \end{array}
$$

This problem has an exponential number of inequalities since there are $2^n - 2$ of proper subsets of S

A oracle to check the separation

Given x_e^* we like to check if

$$\sum_{e\in\delta(S)} x_e^* \geq 2, \; \forall S \subset \mathcal{N}, \; S \neq \emptyset, \mathcal{N}.$$

Assign x_e^* as the capacity for every edge $e \in \mathcal{E}$, then the problem is to check if the min-cut of the graph is greater than or equal to 2.

This problem can be formulated as maximum flow problems (how?) and can be solved in polynomial time in the number of nodes.