6.854J / 18.415J Advanced Algorithms
Fall 2008

18.415/6.854 Advanced Algorithms October 20, 2008

Lecture 12 - Ellipsoid algorithm

Lecturer: Michel X. Goemans

In this lecture we describe the ellipsoid algorithm and show how it can be applied to the linear programming problem.

1 Ellipsoid algorithm

1.1 Definitions

An *ellipsoid* is denoted by

$$E(a, A) = \{x \in \mathbb{R}^n : (x - a)^T A^{-1}(x - a) \leq 1\},$$

with center $a \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n \times m}$ that is positive definite.

Recall that A is *symmetric* if $A = A^T$. A matrix is *positive definite* if it is symmetric and $\forall x \neq 0$, we have $x^T Ax > 0$. The inverse of a positive definite matrix is also positive definite. Symmetric matrices have only real eigenvalues, and positive definite matrices have only real positive eigenvalues.

1.2 Problem statement

Given $P \subseteq \mathbb{R}^n$ bounded, closed, *convex*, find $x \in P$ or show that $P = \emptyset$.

1.2.1 Assumption: Separation oracle

The first issue is how the convex set P is given. We assume that we have a "separation oracle" for P which does the following. Given a, the oracle either

1. affirms that $a \in P$, or
2. outputs $c \in \mathbb{R}^n$ such that $P \subseteq \{x \in \mathbb{R}^n : c^T x < c^T a\}$.

Think of c as the normal vector of the plane separating a and P, pointing away from P. Such a hyperplane exists because P is convex and closed.

An algorithm for our problem would be judged based on how many times it queries the oracle. We would like the number of queries to be polynomial in terms of the input data.

1.2.2 Assumption: Outer ball and minimum volume

As such, the problem is hopeless, since we do not know where to search for a point $x \in P$, and P may even contain just a single point x. So we make two further assumptions. They are

- $P \subseteq$ "big ball", i.e. $P \subseteq B(0, R)$, a ball with center 0 and radius $R > 0$. This tell us where out search can be confined.
- If $P \neq \emptyset$, P has "sufficient" volume. Let's say we are given $r > 0$ such that we are guaranteed that P contains some ball of radius r if P is non-empty.

We consider the size of our input to be $n + \log R - \log r$.

1.3 Sketch of the algorithm

Here is an outline of the ellipsoid algorithm:

- Start with ellipsoid $E_0 = (a_0, A_0)$.
- Maintain an ellipsoid $E_k = (a_k, A_k) \supseteq P$. At iteration k, ask the oracle if a_k belongs to P.
  - If answer is yes, then we are done.
  - If a_k does not belong to P, then the oracle provides a c_k such that $P \subseteq \{x \in R^n : c^T x < c_k^T a_k\}$. Thus, the separating hyperplane slices E_k and P is on one side of this hyperplane. We then determine a smaller ellipsoid E_{k+1} such that

$$E_{k+1} \supseteq E_k \cap \{x : c_k^T x < c_k^T a_k\}. \tag{1}$$

  - (Refer to Fig. (1)).
  - Notice that $E_k \supseteq P$ and we iterate on. If we can show that volume of E_{k+1} decays exponentially, then in "few" iterations, we either find a point in P, or reach $\text{Vol}(E_{k+1}) < \text{Vol}(B(0, r))$ and conclude that $P = \emptyset$.

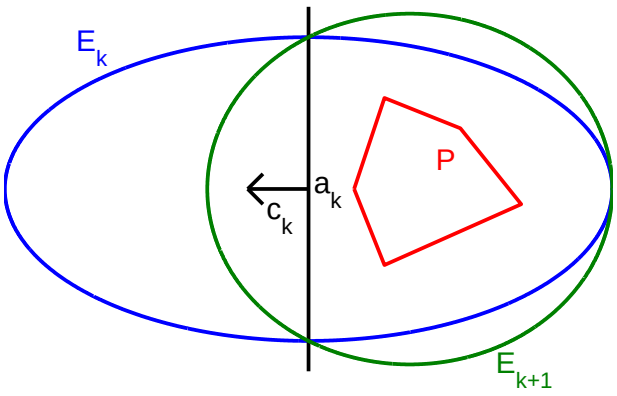

Figure 1: Diagram illustrating a single iteration of the ellipsoid algorithm.

1.4 Bounding volume of ellipsoids

Proposition 1 *Given $E_k = E(a_k, A_k)$ and c_k, we can find E_{k+1} such that Eq. (1) is satisfied and*

$$\frac{Vol(E_{k+1})}{Vol(E_k)} < \exp\left(-\frac{1}{2(n+1)}\right).$$

Let us first focus on the simple case in which our ellipsoid is the unit ball centered at the origin.

Claim 2 *Proposition 1 holds for the special case where $E_k = E(0, I)$ and $c_k = -e_1$.*

Proof: By symmetry, E_{k+1} is an axis-aligned ellipsoid with center along the x_1 axis. It has to contain all points with $x_1 = 0$. See Fig. (2). Formally, we want $E_{k+1} \supseteq E_k \cap \{x : x_1 \geq 0\}$, and one can show that it is enough to guarantee that (i) $e_1 \in E_{k+1}$ and (ii) for all x with $\|x\| = 1$ and $x_1 = 0$, we have $x \in E_{k+1}$.

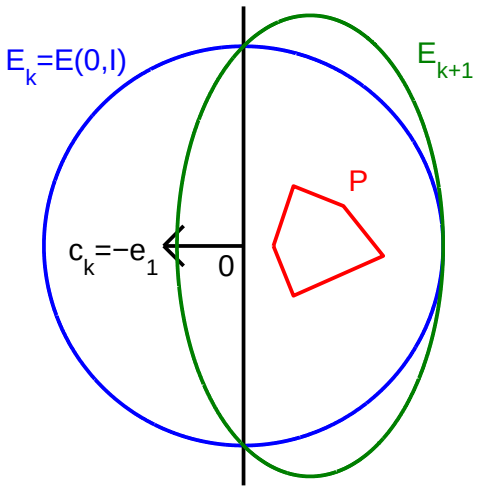

Figure 2: Diagram illustrating the case where $E_k = E(0, I)$.

We propose the following

$$E_{k+1} = \left\{ x : \left(\frac{n+1}{n}\right)^2 \left(x_1 - \frac{1}{n+1}\right)^2 + \frac{n^2-1}{n^2} \sum_{i=2}^{n} x_i^2 \leq 1 \right\}$$
$$= E\left(\frac{1}{n+1} e_1, \frac{n^2}{n^2-1}\left(I - \frac{2}{n+1} e_1 e_1^T\right)\right).$$

It is easy to verify that this ellipsoid satisfies the constraints above. Since the volume of an ellipsoid is proportional to the product of its axis lengths, we obtain:

$$\frac{\text{Vol}(E_{k+1})}{\text{Vol}(E_k)} = \frac{n}{n+1} \cdot \left(\frac{n^2}{n^2-1}\right)^{\frac{n-1}{2}}$$
$$< \exp\left(-\frac{1}{n+1}\right) \exp\left(\frac{1}{n^2-1}\frac{n-1}{2}\right)$$
$$= \exp\left(-\frac{1}{2(n+1)}\right),$$

where we have used the fact that $1 + x < e^x$ whenever $x \neq 0$ (for $x = 0$ we have equality). □

Next, we do a slightly more general case.

Claim 3 *Proposition 1 holds when $E_k = E(0, I)$, $c_k = d$ and $\|d\| = 1$.*

Proof: From the previous simple case, it is clear that the following E_{k+1} works.

$$E_{k+1} = E\left(-\frac{1}{n+1} d, \frac{n^2}{n^2-1}\left(I - \frac{2}{n+1} dd^T\right)\right).$$

□

Proof of Proposition 1:

In general, we can transform $E(a_k, A_k)$ to $E(0, I)$ and map c_k into some d. We can then find an ellipsoid E' as in the proof of Claims 2 and 3, and map it back to obtain E_{k+1}. Denote the linear transformation that maps $E(a_k, A_k)$ into $E(0, I)$ as T. Here is a picture:

$$\begin{array}{ccc} E_k & \stackrel{T}{\rightarrow} & E(0,1) \\ & & \downarrow \\ E_{k+1} & \stackrel{T^{-1}}{\leftarrow} & E' \end{array}$$

Recall that we have

$$E(a, A) = \{x : (x-a)^T A^{-1}(x-a) \leq 1\}.$$

By Cholesky decomposition (since A is positive definite), we can write $A = B^T B$ for some matrix B. If we let $y = (B^{-1})^T(x-1)$, then we have

$$\begin{aligned} (x-a)^T B^{-1}(B^{-1})^T(x-a) &\leq 1 \\ (\Leftrightarrow) \quad y^T y &\leq 1, \end{aligned}$$

so we have a unit ball in the y space. Thus, our linear transformation T and its inverse are:

$$\begin{aligned} T(x) &= y = (B^{-1})^T(x - a_k), \\ T^{-1}(y) &= a_k + B^T y. \end{aligned}$$

We need an equivalent "half-space" constraint after applying T. From Eq. (1),

$$\begin{aligned} c_k^T x &< c_k^T a_k \\ c_k^T(B^T y + a_k) &< c_k^T a_k \\ c_k^T B^T y &< 0. \end{aligned}$$

Hence, in the new space, the unit normal vector of the separating plane is

$$d = \frac{Bc_k}{\sqrt{c_k^T B^T B c_k}}.$$

From Claim 3, we can find an ellipsoid E' in the y space. For convenience (and aesthetic pleasure), let $b = B^T d$.

Apply T^{-1} to E' to obtain

$$\begin{aligned} E_{k+1} &= E(a_{k+1}, A_{k+1}) \\ a_{k+1} &= a_k - \frac{1}{n+1} B^T d = a_k - \frac{1}{n+1} b \\ A_{k+1} &= B^T \left(\frac{n^2}{n^2-1} \left(I - \frac{2}{n+1} dd^T \right) \right) B = \frac{n^2}{n^2-1} \left(A_k - \frac{2}{n+1} bb^T \right). \end{aligned}$$

Since affine transformations preserve the ratios between volumes, we immediately have the desired bound. Here are the details.

$$\begin{aligned} \mathrm{Vol}(E(0, I)) &= \det((B^{-1})^T)\mathrm{Vol}(E_k) \\ \mathrm{Vol}(E_{k+1}) &= \det(B^T)\mathrm{Vol}(E'). \end{aligned}$$

Rearranging, we have

$$\frac{Vol(E_{k+1})}{Vol(E_k)} = \frac{Vol(E')}{Vol(E(0, I))} < \exp\left(-\frac{1}{2(n+1)} \right).$$

□

1.5 Running time

From Proposition 1, we know that $\text{Vol}(E_k) < \text{Vol}(E_0)\exp\left(-\frac{k}{2(n+1)}\right)$. If P is nonempty, then the ellipsoid algorithm terminates in

$$\#\text{ iterations} = O\left(n\log\frac{\text{Vol}(E_0)}{\text{Vol}(P)}\right).$$

By our assumption on P containing a ball of radius r if non-empty, we have that $\frac{\text{Vol}(E_0)}{\text{Vol}(P)} \leq \left(\frac{R}{r}\right)^n$, and thus the number of iterations is

$$\#\text{ iterations} = O\left(n^2(\log R - \log r)\right).$$

If P is empty, by the same number of iterations, we are guaranteed of its emptyness.

We conclude this section by noting a small subtlety. To compute d, we have to be able to find B such that $A = B^TB$. Cholesky decomposition takes $O(n^3)$ and guarantees that numbers in B have size polynomially bounded by the size of numbers in A. But we have to take square roots (in the calculation of d), so we might have to deal with irrational numbers. As a result, we may have to do some rounding to make E_{k+1} slightly bigger. We have to argue that the volume decrease factor is still reasonable, say $\exp\left(-\frac{1}{3(n+1)}\right)$, but this detail shall be omitted.

2 Applying ellipsoid algorithm to linear programming

2.1 Linear programming problem

In the linear programming problem, we are asked to find

$$\min\{c^Tx : Ax = b, x \geq 0\}$$

with inputs A, b, c. The size of the input, from last lecture, is

$$L = m + n + \log\det_{\max} + \log b_{\max} + \log c_{\max}.$$

To apply the ellipsoid algorithm, we will need to

1. Go from an optimization problem to a feasibility problem.
2. Show that the initial convex set is *bounded* and argue about how big the bounding ellipsoid has to be. Argue about termination and provide an inner ball if P is nonempty. i.e. we want P to be *full-dimensional*.

2.2 Optimization to feasibility

We will convert the optimization problem to a feasibility problem as follows:

1. Check feasibility of $Ax = b, x \geq 0$.
2. If answer is infeasible, we are done because LP is infeasible.
3. Otherwise, check feasibility of dual. Dual is $\max\{b^Ty : A^Ty \leq c\}$. Check for feasibility of $A^Ty \leq c$.
   - If dual is not feasible, we are done because LP is unbounded.
   - Otherwise, both primal and dual are feasible. Their solutions have to match by strong duality. Hence, we check for feasibility of $Ax = b, x \geq 0, A^Ty \leq c, c^Tx = b^Ty$ to find a solution for both primal and dual.

2.3 Outer and inner cubes

Here we describe how to go from a system of linear inequalities to an equivalent one (in terms of feasibility) which if non-empty is full-diemnsional and has enough volume.

Proposition 4 *Let $P := \{x : Ax \leq b\}$ and e be the vector of all ones. Assume that A has full column rank n*[1]*. Then P is nonempty iff $P' = \{x : Ax \leq b + \frac{1}{2^L}e, -2^L \leq x_j \leq 2^L \text{ for all } j\}$ is nonempty.*

This proposition allows us to choose E_0 to be a ball centered at the origin containing the cube $\left[-2^L, 2^L\right]^n$. Also, if there exists a $\hat{x}$ such that $A\hat{x} \leq b$ then

$$A\left(\hat{x} \pm \frac{1}{2^{2L}}\right) \leq b + \left(\frac{1}{2^{2L}} n a_{\max}\right) e \leq \frac{1}{2^L} e \quad \text{where } a_{\max} \text{ is max entry of } A.$$

That gives us a little cube around $\hat{x}$. The time for finding an x in P' is thus $O(n \cdot nL)$, because the ratio of the volumes of $\left[-2^L, 2^L\right]^n$ to $\left[-\frac{1}{2^{2L}}, \frac{1}{2^{2L}}\right]^n$ is 8^{Ln}. Recall that finding x in P takes $O(n \log \frac{Vol(E_0)}{Vol(P)})$ iterations. That means LP takes polynomial time in L.

Proof of Proposition 4: We first prove the forward direction. Suppose $P \neq \emptyset$. Our only worry is whether there is any element in P inside the big box. This has been done in previous lecture. We consider a vertex x in P (this exists because A has full column rank). This implies that x is defined by $A_S x = b_S$, where A_S is a submatrix of A. Using Cramer's rule, we can write x as

$$x = \left(\frac{p_1}{q}, \frac{p_2}{q}, \cdots, \frac{p_n}{q}\right)$$

with $|p_i| < 2^L$ and $1 \leq q < 2^L$.

We now work on the converse. $\{x : Ax \leq b\} = \emptyset$ implies, by Farkas' Lemma, there exists a y such that $y \geq 0$, $A^T y = 0$, and $b^T y = -1$. We can choose a vertex of $A^T y = 0$, $b^T y = -1$, $y \geq 0$. Rewrite this as

$$\begin{pmatrix} A^T \\ b^T \end{pmatrix} y = \begin{pmatrix} 0 \\ -1 \end{pmatrix}, y \geq 0.$$

By Cramer's rule, we can bound the components of a basic feasible solution y as:

$$y^T = \left(\frac{r_1}{s}, \cdots, \frac{r_m}{s}\right),$$

with $0 \leq s, r_i \leq \det_{\max} \begin{pmatrix} A^T \\ b^T \end{pmatrix}$. Expanding the determinant along the last row, we see that $\det_{\max} \begin{pmatrix} A^T \\ b^T \end{pmatrix} \leq m b_{\max} \det_{\max}(A)$. Using the fact that $2^L > 2^m 2^n \det_{\max}(A) b_{\max}$, we obtain $0 \leq s, r_i < \frac{m}{2^m 2^n} 2^L \leq \frac{m}{2^{m+1}} 2^L$.

Therefore,

$$\left(b + \frac{1}{2^L} e\right)^T y = \underbrace{b^T y}_{-1} + \frac{1}{2^L} e^T y = -1 + \frac{m^2}{2^{m+1}} < 0.$$

(The last inequality holds for $m \geq 1$.) By Farkas' Lemma again, this y shows that there is no x satisfying $Ax \leq b + \frac{1}{2^L} e$, i.e. P' is empty. □

[1] **Small detour:** We have previously dealt with the constraint problem $Ax = b, x \geq 0$. If this is non-empty, then we have a vertex in the feasible solution. However, there is *not* guaranteed if the constraints are of the form $Ax \leq b$. But if we have $\text{rank}(A) = n$, $A \in \mathbb{R}^{m \times n}$, then a non-empty P will always contain a vertex. In our case, since we convert from the problem with constraints $x \geq 0$, we would have inequalities $-Ix \leq 0$ and full column rank.

2.4 Obtaining a solution

There is one last problem. If the ellipsoid method returns a x in P', x might not be in P.

One solution is to round the coefficients of the inequalities to rational numbers and "repair" these inequalities to make x fit in P. This is called simultaneous Diophantine approximations, and will not be discussed.

We can solve this problem by another method. We give a general method for finding a feasible solution of a linear program, assuming that we have a procedure that checks whether or not the linear program is feasible, e.g. ellipsoid algorithm.

Assume, we want to find a solution of $Ax \leq b$. The inequalities in this linear program can be written as $a_i^T x \leq b_i$ for $i = 1, \cdots, m$. We use the following algorithm:

1. $I \leftarrow \emptyset$.
2. For $i \leftarrow 1$ to m do
   - If the set of solutions of
   $$\left\{ \begin{array}{ll} a_j^T x \leq b_j & \forall j = i+1, \cdots, m \\ a_j^T x = b_j & \forall j \in I \cup \{i\} \end{array} \right\}$$
   is nonempty, then $I \leftarrow I \cup \{i\}$.
3. Finally, solve x in $a_i^T x = b_i$ for $i \in I$ with Gaussian elimination.

We assume that the solution is a vertex and satisfies some equalities. If at step 2, making inequality i an equality makes the problem infeasible, then the vertex cannot depend on this inequality and we can discard it.