UBC CPSC 536N: Sparse Approximations Winter 2013

Lecture 4 — January 14, 2013

Prof. Nick Harvey *Scribe: Alexandre Fréchette*

This lecture is about the ellipsoid method. We cover the basis of its geometric foundation and study how it is used in solving optimization problem, notably linear programs.

Our main concern is to answer the following question: "Are linear programs in **P**?". Recall that **P** is the class of computational problems that can be solved efficiently, that is in time less than n^c for some $c > 0$, where n is the size of the problem's input.

In the case of a linear program $\max c^T x : Ax \leq b$, there are two ways to define the input size:

- The total number of bits needed to represent c, A and b, or *bit size* of the input.
- The total number of entries of c, A and b, that is $md + m + d$ for $c \in \mathbb{R}^d$, $A \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^m$.

If a problem is solvable in time polynomial in its bit size, then it runs in *polynomial time*. On the other hand, if it is solvable in time polynomial in the number of entries in the input, then it runs in *strongly polynomial time*.

There are many different algorithms for solving linear programs (not all necessarily theoretically efficient). For example, the simplex method does not solve linear programs in strong polynomial time, but is one of the most vastly used algorithm in practice.

Nevertheless, we show that linear programs, amongst other optimization problems, can be solved in polynomial time[1] We achieve this using the ellipsoid method.

1 Geometry of Ellipsoids

First, we cover some basic definitions and geometric results about ellipsoid.

Definition 1.1. Let $B = \{x \in \mathbb{R}^n : \|x\| \leq 1\}$ be the unit ball, and $f : \mathbb{R}^n \to \mathbb{R}^m$ be an affine map. Then $E = f(B)$ is an *ellipsoid*.

An affine map is a transformation that preserves straight lines, and can also be thought of as a linear transformation followed by a translation.

[1] As of now, solving linear programs in *strongly* polynomial time is an open question.

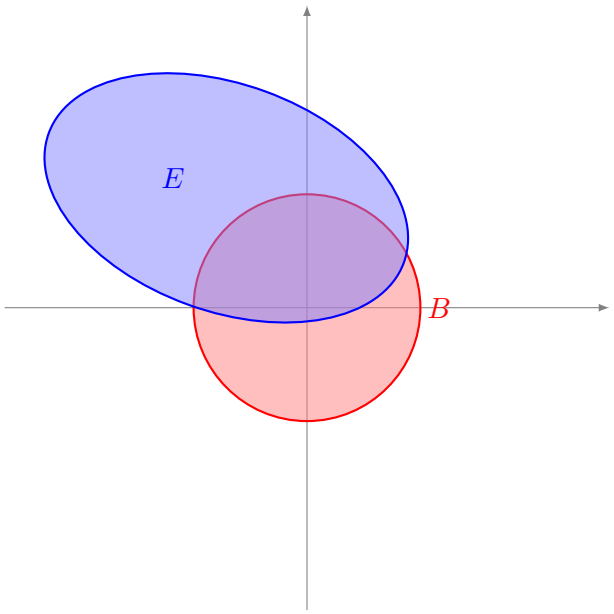


Figure 1: An ellipsoid E as an affine transformation of the unit ball B in $\mathbb{R}^2$.

We restrict to the case where $n = m$ and f is invertible, that is $f(x) = Ax + b$ where A is a square, non-singular matrix. In this case, it serves to have a more compact representation of E:

Claim 1.2.

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

Proof.

$$\begin{aligned} E &= f(B) \\ &= \{f(x) : x \in \mathbb{R}^n, \|x\| \leq 1\} \\ &= \{y : y \in \mathbb{R}^n, \|f^{-1}(y)\| \leq 1\} \end{aligned}$$

since $n = m$ and f is invertible,

$$= \{y \in \mathbb{R}^n : \|A^{-1}(y-b)\| \leq 1\}$$

since $f(x) = Ax + b$,

$$E = \{y \in \mathbb{R}^n : (y-b)^T {A^{-1}}^T A^{-1}(y-b) \leq 1\}$$

□

We shorten the notation by

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

for the positive definite matrix $M = AA^T$ and vector b. The volume of $E(M, b)$ is

$$\mathsf{vol}(E(M,b)) = \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)}\sqrt{\det(M)},$$

which is obtained from the volume of the ball in $\mathbb{R}^n$. Note that $E(aM^{-1}, 0)$ is a level set of $f(x) = x^T Mx$, that is $E(aM^{-1}, 0) = \{x \in \mathbb{R}^n : x^T Mx \leq a\}$.

The main geometric problem at the heart of the ellipsoid method is to iteratively cover half-ellipsoids with a smaller volume ellipsoid.

Consider the simpler problem of covering a half ball with an ellipsoid. We let B be our unit ball centred at the origin, and $H_u = \{x \in \mathbb{R}^n : u^T x \geq 0\}$ be an hyperplane through the origin, cutting B in half - see Figure 2.

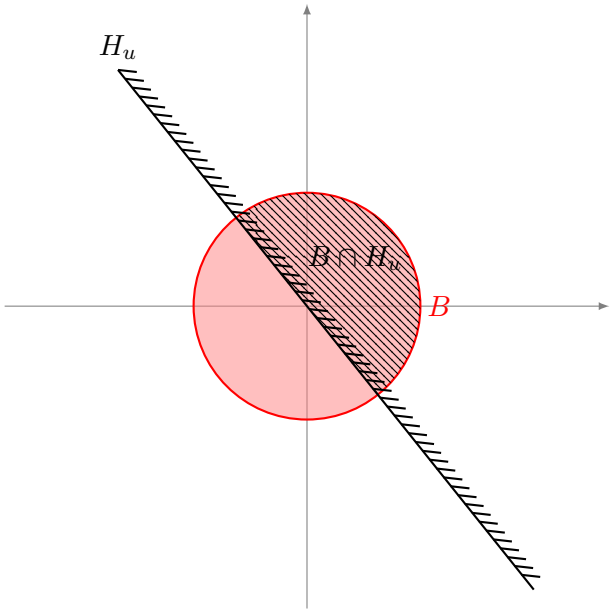


Figure 2: The unit ball B centred at the origin in $\mathbb{R}^2$ separated in half by the plane H_u passing by the origin.

Can we find an ellipsoid containing $B \cap H_u$? Well B is a trivial ellipsoid, and it obviously contains $B \cap H_u$. So, more interestingly, can we find an ellipsoid E containing $B \cap H_u$ with volume strictly less than the volume of B (i.e. a "smaller" ellipsoid) - see Figure 3?

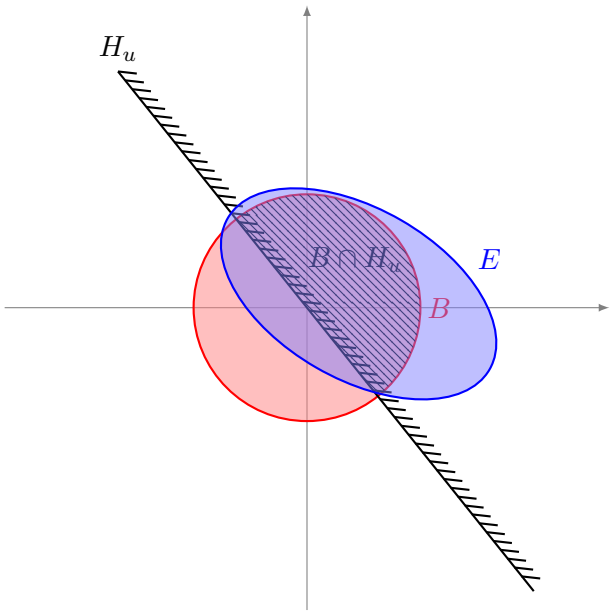


Figure 3: An ellipsoid E covering the region $B \cap H_u$.

Lemma 1.3. Let $B = \{x \in \mathbb{R}^n : \|x\| \leq 1\}$ and $H_u = \{x \in \mathbb{R}^n : x^T u \geq 0\}$ for an arbitrary unit vector u. Then there is an ellipsoid $E(M, b)$ of volume strictly smaller than the volume of B such that $B \cap H_u \subseteq E(M, b)$.

Proof. If we let $M = \frac{n^2}{n^2-1}(I - \frac{2}{n+1}uu^T)$ and $b = \frac{u}{n+1}$, then $B' = E(M, b)$ is such that

- $B \cap H_u \subseteq B'$,
- $\frac{\mathsf{vol}(B')}{\mathsf{vol}(B)} \leq e^{-\frac{1}{4}(n+1)} \leq 1 - \frac{1}{8(n+1)}$

For additional details, see Prof. Harvey's Lecture 4 Extra Notes. □

It will turn out to be crucial that B' provided in Lemma 1.3 has a volume shrinkage factor that is polynomial in n.

Moreover, recall that we need to cover half *ellipsoids* with a smaller volume ellipsoid. Can this be done in a same manner? We since B in Lemma 1.3 is almost a generic ellipsoid, so we can certainly leverage the result.

Theorem 1.4. Let E be an ellipsoid centred at origin and $H_a = \{x \in \mathbb{R}^n : x^T a \geq 0\}$ for an arbitrary unit vector a. Then there is an ellipsoid E' of volume strictly smaller than the volume of E such that $E \cap H_u \subseteq E(M, b)$.

For the proof and additional details, see Prof. Harvey's Lecture 4 Extra Notes.

Theorem 1.4 is all the geometry we need for the ellipsoid method.

2 Ellipsoid Method

At its base, the ellipsoid algorithm only tries to find a feasible solution to a system of linear inequalities. It proceed by repeatedly finding smaller and smaller ellipsoid that should contain

the feasible polyhedral region P - see Figure 4. We stop once we have reached a small enough ellipsoid so that it could never contain our region.

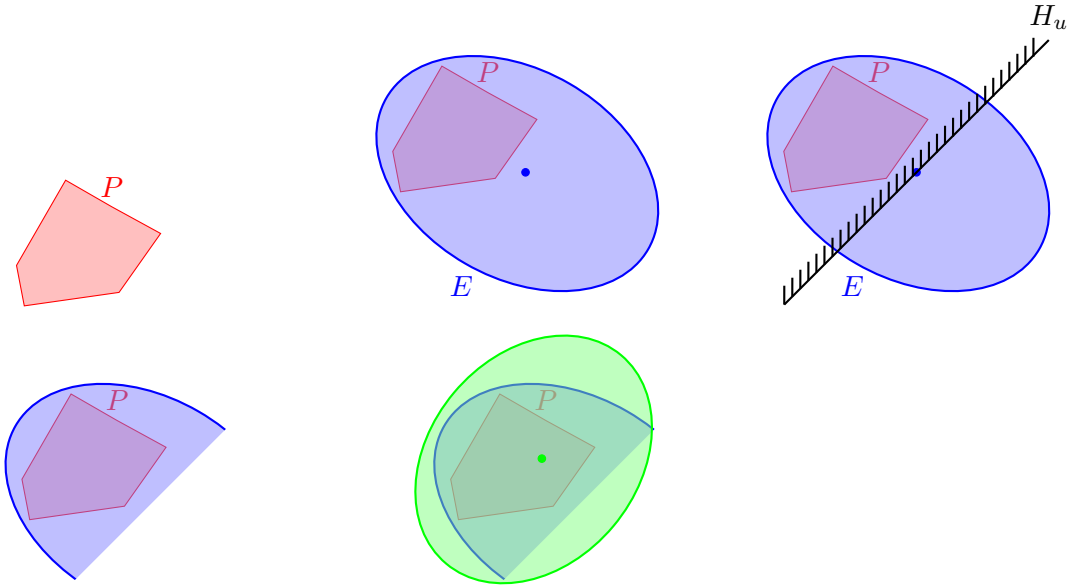


Figure 4: Two steps of the ellipsoid method for finding a feasible point in P. We start with an ellipsoid covering P. Then in the first step, the ellipsoid E's center is not in P, so the ellipsoid is halved by an hyperplane H_u. This half-ellipsoid is then covered by a new ellipsoid of smaller volume. This time, its center is in P.

Formally, we have a region $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ and we want to find $x \in P$ or return that $P = \emptyset$. We start with an ellipsoid $E(M, z) \supset P$ (centred at z). If $z \notin P$, then it violates some constraint of P, say $a_i^T x \leq b_i$. This is our halving hyper-plane, so $P \subseteq \{x \in \mathbb{R}^n : a_i^T x \leq a_i^T z\}$. Hence, $P \subseteq E(M, z) \cap \{x \in \mathbb{R}^n : a_i^T x \leq a_i^T z\}$. By Theorem 1.4, we can find an ellipsoid $E(M', z')$ covering $E(M, z) \cap \{x \in \mathbb{R}^n : a_i^T x \leq a_i^T z\}$. We're back at square one, but with an ellipsoid of smaller volume. So we repeat until either an ellipsoid's center is in P, or the ellipsoid is *small enough* so that it can't contain P. See Algorithm 2.1 for pseudo-code.

Algorithm 2.1 Ellipsoid Method

Require: A polyhedral region $P = \{x \in \mathbb{R}^n : Ax \leq b\}$, and two radii $R > 0$ and $r > 0$.
Ensure: A point $x \in P$ or $P = \emptyset$.

Let $E(M, z)$ be an ellipsoid such that $P \subseteq E(M, z)$ - typically $E(M, z) = B(0, R)$.
while true do
  if $\mathsf{vol}(E(M, z)) < \mathsf{vol}(B(0, r))$ **then**
    return $P = \emptyset$
  else if $z \in P$ **then**
    return z
  else
    Let $a_i^T x \leq b_i$ be a constraint of P violated by z.
    Let $H = \{x \in \mathbb{R}^n : a_i^T x \leq a_i^T z\}$.
    Let $E(M', z')$ be an ellipsoid covering $E(M, z) \cap H$.
    $M \leftarrow M'$, $z \leftarrow z'$
  end if
end while

There are two main assumptions with the ellipsoid method. The first one is that the feasible region P is contained in a finite ellipsoid, without loss of generality assume a ball $B(0, R)$ - this is our starting state. Secondly, we assume there is a radius $r > 0$ small enough so that P cannot be contained in any ellipsoid of volume less than the volume of $B(0, r)$ - this is our stopping criterion. We'll come back to those issues.

For now, let's consider the running time of the ellipsoid method. Let E_i be the ellipsoid we create at the i-th iteration. Note that $E_0 = B(0, R)$. We want to know how long it takes to meet our stopping criterion, that is $\mathsf{vol}(E_i) < \mathsf{vol}(B(0, r))$.

Claim 2.1.

$$\mathsf{vol}(E_k) \leq e^{\frac{-k}{4(n+1)}} \mathsf{vol}(E_0).$$

Proof. By Theorem 1.4, we know that $\mathsf{vol}(E_{i+1}) \leq e^{\frac{-1}{4(n+1)}} \mathsf{vol}(E_i)$. Hence,

$$\mathsf{vol}(E_k) \leq \mathsf{vol}(E_0) \prod_{i=1}^{k} e^{\frac{-1}{4(n+1)}} = e^{\frac{-k}{4(n+1)}} \mathsf{vol}(E_0).$$

□

Theorem 2.2. The ellipsoid method with input parameters R and r does at most

$$4n(n+1)\log(\frac{R}{r})$$

iterations.

Proof. Suppose $k > 4n(n+1)\log(\frac{R}{r})$, then $\frac{-k}{4(n+1)} < n\log(\frac{r}{R})$ and thus $e^{\frac{-k}{4(n+1)}} < (\frac{r}{R})^n = \frac{\mathsf{vol}(B(0,r))}{\mathsf{vol}(B(0,R))}$. By Claim 2.1,

$$\mathsf{vol}(E_k) \leq \mathsf{vol}(E_0) e^{\frac{-k}{4(n+1)}} < \mathsf{vol}(B(0, R)) \frac{\mathsf{vol}(B(0, r))}{\mathsf{vol}(B(0, R))} = \mathsf{vol}(B(0, r)),$$

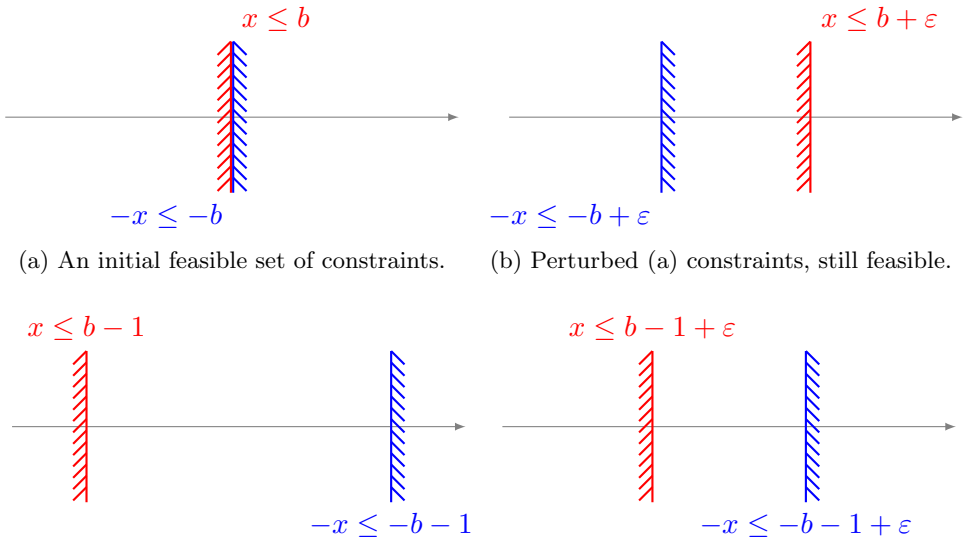
so the stopping criterion is met. Hence, we can never go above $4n(n+1)\log(\frac{R}{r})$ iterations. □

Our treatment of the ellipsoid method leaves us with two seemingly strong assumptions, plus additional details that we have swept under the rug. Let's see what we can say about those here. This time, assume we want to run the ellipsoid method on a polyhedron $P = \{x \in \mathbb{R}^m : Ax \leq b\}$, where $A \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^d$ are given in a binary file. Let n be the number of bits to store that file.

First, the ellipsoid method needs to compute square roots (e.g. to compute the volume of an ellipsoid), and thus deal with irrational numbers. This now becomes a numerical computing problem, and the solution is to approximate irrational numbers with rational, incurring an error. The approximations proliferate, it gets messy, but can be dealt with.

Second, we assume we can only work with bounded polyhedron, hence the need of an input radius $R > 0$ for our starting bounding ellipsoid $B(0, R)$. However, we can add constraint $-U \leq x_i \leq U$ for every $i \in [m]$ based on the actual values in A and b. For instance, $U = 16^{d^2 n}$ is a valid choice, taking in account the total bit size n required to represent A and b. In this case, P with these added new U-constraints is contained in $B(0, R)$ for $R < nU$.

Lastly, we needed the polyhedron to contain a small ball $B(0, r)$, hence the need of an input radius $r > 0$. To fix this, we can perturb the constraints by an amount small enough so that the perturbed polyhedron is feasible if and only if the original one is, and, if feasible, it contains a small ball - see Figure 5 for an example. Concretely, we can add ε to every b_i, where $\varepsilon = \frac{1}{U^2}$. The perturbed polyhedron then must contain a $B(0, r)$ of radius $r = \frac{\varepsilon}{2^{dn}} > \frac{1}{U^3}$.



(a) An initial feasible set of constraints. (b) Perturbed (a) constraints, still feasible.

(c) An initial infeasible set of constraints. (d) Perturbed (c) constraints, still infeasible.

Figure 5: One dimension examples of perturbation by $\varepsilon > 0$.

We got rid of the input dependence on R and r - we can now provide those values based on A and b. So what is the final running time of the ellipsoid method for solving linear programs? Well each iteration consists of basic matrix operations (i.e. defining the new ellipsoid) and basic polyhedron information (i.e. test $z \in P$ and return violated inequality if needed), and all of this runs in time polynomial in n (bit size of the input A and b) and d (dimensionality of our problem). Moreover, we now know that $R < nU$ and $r > \frac{1}{U^3}$ for $U = 16^{d^2 n}$. Hence, by

Theorem 2.2, we will do at most $4d(d+1)\log(\frac{R}{r})$ iterations, which is at most $40d^6n^2$ - polynomial in n and d.

2.1 Solving Linear Programs

So now we have a method for finding a feasible point in a polyhedron P, or returning that P is empty. How can use this to solve an optimization problem $\max\{c^T x : x \in P\}$ for $P = \{x \in \mathbb{R}^n : Ax \leq b\}$? Recall from the Strong Duality Theorem that the primal linear program has an optimal solution if and only if the dual has an optimal solution, and also if and only if the following system is feasible

$$\begin{aligned} Ax &\leq b && \text{(primal feasibility)} \\ A^T y &= c && \text{(dual feasibility)} \\ y &\geq 0 && \text{(dual feasibility)} \\ c^T x &\geq b^T y && \text{(matching primal and dual objective values)} \end{aligned} \tag{2.1}$$

The last constraint $c^T x \geq b^T y$ along with weak duality (i.e. $c^T x \leq b^T y$) implies $c^T x = b^T y$. Hence, to solve our linear program, we can just find a feasible point in the polyhedron defined by Equation (2.1).

However, there might be situations where we do not care to solve the dual problem (e.g., if the primal has many constraints, the dual with be very high-dimensional). In this case, we can use a slightly modified version of the ellipsoid to perform a (multi-dimensional) binary search for the optimal linear program value. For instance, suppose we know the optimal value is in $[L, U]$. We can then add the constraint $c^T x \geq \frac{L+U}{2}$ to P and check if this new polyhedron is feasible (using the ellipsoid method). If yes, then we know the optimal is of value at least $\frac{L+U}{2}$. So update $L \leftarrow \frac{L+U}{2}$ and repeat the search on P. Otherwise, all solutions are of values less than $\frac{L+U}{2}$, so update $U \leftarrow \frac{L+U}{2}$ and repeat the search - see Figure 6.

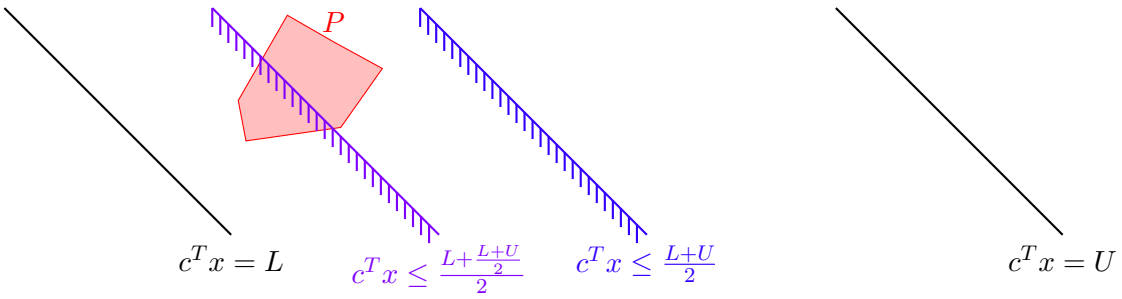


Figure 6: Binary searching for the optimal value by segmenting space based on objective value and checking for feasibility.

3 Separation & Optimization

If we look back at the actual functioning of the ellipsoid method, it turns out it uses very little information about the polyhedra. In fact, only lines 5 and 8 in Algorithm 2.1 require something from the polyhedra P. That is, together the lines test membership of z in P, and following a

negative response it must be able to produce a violated inequality. One could outsource both of these operations to what is known as a *separation oracle*.

Definition 3.1. Given a polyhedron $P \subseteq \mathbb{R}^n$ and a point $z \in \mathbb{R}^n$, a *separation oracle* is an efficient algorithm that answers the question $z \in P$, and, if $z \notin P$, produces a valid inequality for P violated by z.

Here is an example of a separation oracle.

Example 3.2 (Separation Oracle for the Unit Ball). Let $P = B(0,1)$ be the unit ball in $\mathbb{R}^n$. Given a vector $z \in \mathbb{R}^n$, we can efficiently test that $z \in P$ by testing that $\|z\| \leq 1$. If $z \notin P$, we need to provide a violated inequality. Consider $a = \frac{z}{\|z\|}$, the inequality coming from the tangent of P at z - see Figure 7.

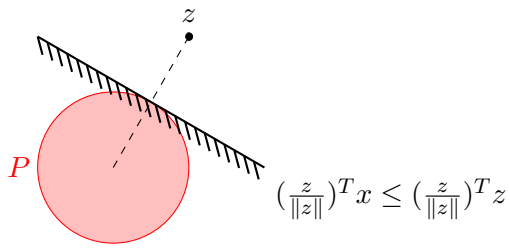


Figure 7: A valid inequality for the two dimensional ball P that is violated for a point z outside of P.

Then for all $x \in P$, we get that $a^T x = \frac{z^T x}{\|z\|} \leq \|x\|$ using the Cauchy-Schwarz inequality[2]. On the other hand, for z we get that $a^T z = \frac{z^T z}{\|z\|} = \|z\| > 1 \geq \|x\|$ for any $x \in P$. Hence, $a^T x < a^T z$ is a valid inequality for P that is violated by z. Finally, both membership testing and the valid inequality can be solved efficiently (in fact in closed form in this case).

Having abstracted the details of the polyhedron inside a separation oracle, we get that the ellipsoid method works for any convex set P as long as a separation oracle is available. This leads to the following theorem:

Theorem 3.3 (Feasibility Theorem - Grötschel-Lovasz-Schrijver, 1981). For any convex set $P \subseteq \mathbb{R}^n$ with a separation oracle, you can find a feasible point efficiently.

There are two main caveats. The efficiency of finding feasible points depends on the size of the bounding ellipsoid containing P, and the smallest ball inside P. Moreover, the fact that numerical approximation of irrational numbers is needed means we only get approximately feasible points.

As feasibility implies optimization in the case of linear programs, it turns out we can also use the "binary search" form of the ellipsoid method to optimize linear functions over arbitrary convex sets:

Theorem 3.4 (Optimization Theorem - Grötschel-Lovasz-Schrijver, 1981). For any convex set $P \subseteq \mathbb{R}^n$ with a separation oracle, you can can solve the optimization problem $\max c^T x : x \in P$ for any $c \in \mathbb{R}^n$.

Furthermore, this can be generalizes to arbitrary non-linear (convex) objective functions.

[2]The Cauchy-Schwarz inequality states that $\langle x, z \rangle \leq \|x\|\|z\|$ for any inner product $\langle \cdot, \cdot \rangle$.

This can be achieved by noting that the level sets of a convex objective function are convex, and modifying the binary search procedure accordingly.