Ellipsoid method

Shubh Maheshwari
Center for Visual Information Technology
shubh.maheshwari@students.iiit.ac.in

***Abstract*— The ellipsoid method is an iterative method for minimizing convex functions. When specialized to solving feasible linear optimization problems with rational data, the ellipsoid method is an algorithm which finds an optimal solution in a finite number of steps. The class of ellipsoid optimization problems includes some of the most well known convex optimization examples. In this report, first we discuss the basics of ellipsoid optimization, then we solve 2 problems using ellipsoid method. Lastly we explain the time complexity of the method**

I. Introduction

The simplex algorithm was the first algorithm proposed for linear programming, and although the algorithm is quite fast in practice, no variant of it is known to be polynomial time. The Ellipsoid algorithm is the first polynomial-time algorithm discovered for linear programming. The Ellipsoid algorithm was proposed by the Russian mathematician Shor in 1977 for general convex optimization problems, and applied to linear programming by Khachyan in 1979. Contrary to the simplex algorithm, the ellipsoid algorithm is not very fast in practice; however, its theoretical polynomiality has important consequences for combinatorial optimization problems as we will see. Interior-point algorithms are also quite practical but they do not have the same important implications to combinatorial optimization as the ellipsoid algorithm does. Hence, ellipsoid method is a very important theoretical tool for developing polynomial time algorithms for a large class of convex optimization problems, which are much more general than linear programming.

II. Ellipsoid Method

The ellipsoid method generates a sequence of ellipsoids whose volume uniformly decreases at every step, thus enclosing the optimal solution of a convex function.

A. Iterative Algorithms

Ellipsoid method is an iterative algorithm. An iterative method is a mathematical procedure that uses an initial guess to generate a sequence of improving approximate solutions for a class of problems, in which the n-th approximation is derived from the previous ones. A specific implementation of an iterative method, including the termination criteria, is an algorithm of the iterative method. An iterative method is called convergent if the corresponding sequence converges for given initial approximations.

**The footnote marks may be inserted manually

B. Basic Procedure

The problem being considered by the ellipsoid algorithm is:

Given a bounded convex set $P \in R^n$ find $x \in P$.

We will see that we can reduce linear programming to finding an x in P = $\{x \in R^n : Cx \leq d\}$

The ellipsoid algorithm works as follows. We start with a big ellipsoid E that is guaranteed to contain P. We then check if the center of the ellipsoid is in P. If it is, we are done, we found a point in P. Otherwise, we find an inequality $c^T x \leq d_i$ which is satisfied by all points in P (for example, it is explicitly given in the description of P) which is not satisfied by our center. The ellipsoid algorithm is the following

- Let E_0 be an ellipsoid containing P
- while center a_k of E_k is not in P do
  - Let $c^T x \leq c^T a_k$ be such that $\{x : c^T x \leq c^T a_k\} \supseteq P$
  - Let $E_k + 1$ be the minimum volume ellipsoid containing $E_k \cap \{x : c^T x \leq c^T a_k\}$
  - k $\Leftarrow$ k + 1

The ellipsoid algorithm has the important property that the ellipsoids constructed shrink in volume as the algorithm proceeds; this is stated precisely in the next lemma. This means that if the set P has positive volume, we will eventually find a point in P. We will need to deal with the case when P has no volume (i.e. P has just a single point), and also discuss when we can stop and be guaranteed that either we have a point in P or we know that P is empty.

C. Ellipsoid

Before we can state the algorithm more precisely, we need to define ellipsoids:

Given a center a, and a positive definite matrix A, the ellipsoid E(a, A) is defined as $\{x \in R^n : (x-a)^T A^{-1}(x-a) \leq 1\}$

One important fact about a positive definite matrix A is that there exists B such that $A = B^T B$, and hence $A^{-1} = B^{-1}(B^{-1})^T$. Ellipsoids are in fact just affine transformations of unit spheres. To see this, consider the (bijective) affine transformation $T : x \Leftarrow y = (B^{-1})^T(xa)$. It maps $E(a, A) \Leftarrow \{y : y^T y \leq 1\} = E(0, I)$

D. Initialization/Starting Ellipsoid

Now, we need to consider using the ellipsoid to find a feasible point in P or decide that P is empty. As starting ellipsoid, we can use the ball centered at the vector (1/2 ,1/2, ,1/2) and of radius $\frac{1}{2}\sqrt{n}$ (which goes through all 0, 1 of n

vectors). This ball has volume Vol(E0) = $\frac{1}{2^n}(\sqrt{n})^n Vol(B_n)$, where B_n is the unit ball. The $Vol(B_n)$ has the upper bound of $\pi^{n/2}$.

E. Steps

Now that we have initialization for the iteration method, we can define a set of steps to reach from step k to step k+1. For every step k, we have the center of the ellipsoid at a_k and its conic transformation A_k

While $a_k \notin P$ do:

- Let $c^T x \leq d$ be an inequality that is valid for all $x \in P$ but $c^T a_k > d$
- Let $b = \frac{A_k c}{\sqrt{c^T A_k c}}$
- Let $a_{k+1} = a_k - \frac{1}{n+1} b$
- Let $A_{k+1} = \frac{n^2}{n^2-1}(A_k - \frac{2}{n+1} b b^T)$

For each iteration we can claim that: $\frac{Vol(E_{k+1})}{Vol(E_k)} < exp(-\frac{1}{2(n+1)})$ Hence we can say that the algorithm should find $x \in P$ in less than $2(n+1)ln\frac{Vol(E_o)}{Vol(P)}$ steps.

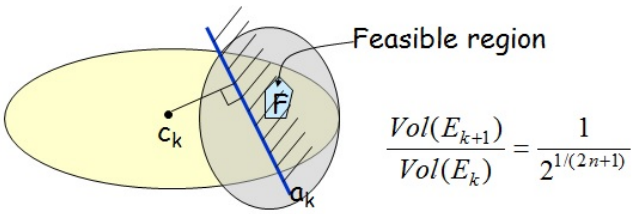


Fig. 1: Reduction in volume of E_k at each iteration

F. From feasibility to Optimization

First, let us show how to reduce such an optimization problem to the problem of finding a feasible point in a polytope. Let $c^T x with c \in R^n$ be our objective function we would like to minimize over P. Assume without loss of generality that $c \in Z^n$. Instead of optimizing, we can check the non-emptiness of

$$\grave{P} = P \cap \{x : c^T x \leq d + \frac{1}{2}\}$$

for $d \in Z$ and our optimum value corresponds to the smallest such d. As $S \subseteq \{0,1\}^n$, d must range in $[n_c max, n_c max]$ where $c_{max} = max_i c_i$. To find d, we can use binary search (and check the non-emptiness of $\grave{P}$ with the ellipsoid algorithm). This will take $O(log(nc_{max})) = O(logn + logc_{max})$ steps, which is polynomial.

G. Termination Criterion

We will now argue that if $\grave{P}$ is non-empty, its volume is not too small. Assume that $\grave{P}$ is non-empty, say $v_0 \in \grave{P} \cap \{0,1\}^n$. Our assumption that P is full-dimensional implies that there exists $v_1, v_2, , v_n \in P\{0,1\}^n = S$ such that the "simplex" $v_0, v_1, , v_n$ is full-dimensional. The v_is may not be in $\grave{P}. Instead, define$w$_i$ for i = 1, ... , n by:

$$w_i = \{ v_{\ i}, if c^T v_i \leq d + \frac{1}{2}, and v_o + a(v_i - v_o), otherwise,$$

where $a = \frac{1}{2nc_{max}}$. This implies that $w_i \in \grave{P}$ as

$$c^T w_i = c^T v_0 + a c^T (v_i v_0) d + \frac{1}{2nc_{max}} nc_{max} = d + \frac{1}{2}$$

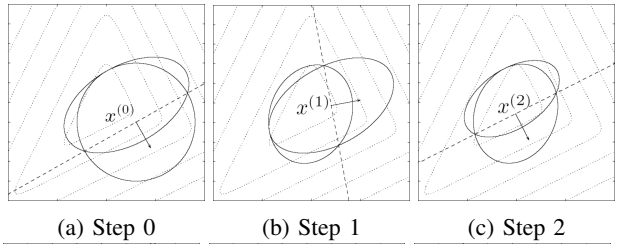
We have that $\grave{P}$ contains C = conv($v_0, w_1, w_2, ..., w_n$) and Vol(C) is $\frac{1}{n!}$ times the volume of the parallelipiped spanned by $w_i v_0 = {}_i(v_i v_0)$ (with ${}_i \in \{a, 1\}$) for i = 1, ... , n. This paralleliped has volume equal to the product of the ${}_i$ (which is at least a^n) times the volume of a parallelipiped with integer vertices, which is at least 1. Thus,

$$Vol(\grave{P}) \geq Vol(C) = \frac{1}{n!}(\frac{1}{2nc_{max}})^n$$

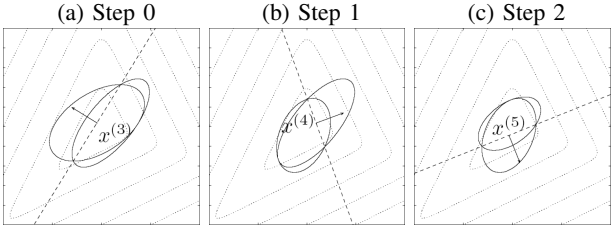
This is polynomial.

H. Example

The figure below is an example of ellipsoid method. We start with the a circle and our goal is to reach the shown ellipse. The figure shows the first 6 iterations of the method.



(a) Step 0 (b) Step 1 (c) Step 2



(d) Step 4 (e) Step 5 (f) Step 6

I. Deep-cut ellipsoid algorithm

If an upper bound on the objective function is available at each iteration, deep cuts can be used to improve performance over the basic ellipsoid algorithm. Suppose that at iteration k, we have an upper bound on the optimal objective value: f f (k) best f(x (k)). (One such bound is the best point found so far, since the iterates x (k) do not in general produce monotonically decreasing objective values). Given g f(x), we have for all z

J. Ellipsoid algorithm with constraints

The (deep-cut) ellipsoid algorithm is readily modified to solve the constrained problem minimize $f_0(x)$ subject to fi(x) 0, i = 1, . . . , m. In this section we describe one such modification. Once again, we generate a sequence of ellipsoids of decreasing volume, each of which is guaranteed to contain a feasible minimizer. If x(k) is feasible $(f_i(x(k)) \leq 0$ for i = 1, . . . , m) then we form E(k+1) exactly as in the (deep-cut) ellipsoid algorithm; we call this an objective iteration. If x(k)is infeasible (fi(x(k)) ¿ 0 for some i) then we form

E(k+1) as in the ellipsoid algorithm, but using a subgradient of a violated constraint function fi instead of the objective. We call this a constraint iteration.

In a constraint iteration, all points we discard are infeasible. In an objective iteration, all points we discard have objective value greater than or equal to the current, feasible point. Thus in each case, we do not discard any minimizers, so that the ellipsoids will always contain any minimizers that are in the initial ellipsoid. The same proof as for the basic ellipsoid algorithm shows that this modified algorithm works provided the set of feasible -suboptimal points has positive volume. (This can be related to Slaters condition, assuming the constraint functions are Lipschitz.)

III. Problems

A. Unconstrained example

We consider the problem of minimizing a piecewise affine function

$$minimize : f(x) = max_{i=1,\ldots,m}(a_i x + b)$$

with variable $x \in R^n$. We use n = 20 variables and m = 100 terms, with problem data a_i and b_i generated from a unit normal distribution

with variable $x \in R^n$. We use n = 20 variables and m = 100 terms, with problem data a_i and b_i generated from a unit normal distribution

1) Solution: We use the basic ellipsoid algorithm described above, starting with $P_o = I$, and $x_o = 0$

Figure 3 shows the convergence of $f^{(k)}$ and $f(x^{(k)}) - \sqrt{g^{(k)T} P^k g(k)}$ (a lower bound on f^*), to f^* with the iteration number)

Figure 4 shows the convergence of $f_{best}^{(k)} = min_{i=1,\ldots,k} f(x^{(i)})$ is the best point found at iteration k.

The slow convergence shown in this example is in fact typical for this method

B. Constrained Example

We consider the problem of minimizing a piecewise affine function with additional constraints on x.

$$minimize : f(x) = max_{i=1,\ldots,m}(a_i x + b)$$

$$subject : |x_i| \leq 0.1$$

1) Solution: We use the constrained ellipsoid algorithm, described above, starting with $P_o = I$, and $x_o = 0$

Figure 5 shows the convergence of $f_{best}^{(k)} - f^*$

IV. Time Complexity

As we proved one of the main advantage of using ellipsoid method over simplex is that the solution is garunteed in polynomial time.

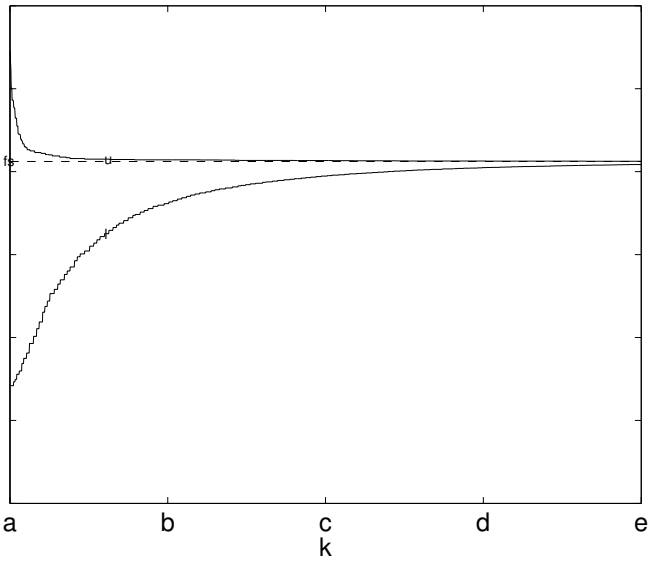


Fig. 3: Convergence of $f_{best}^{(k)}$ and $f(x^{(k)}) - \sqrt{g^{(k)T} P^k g(k)}$ (a lower bound on f^*), to f^* with the iteration number

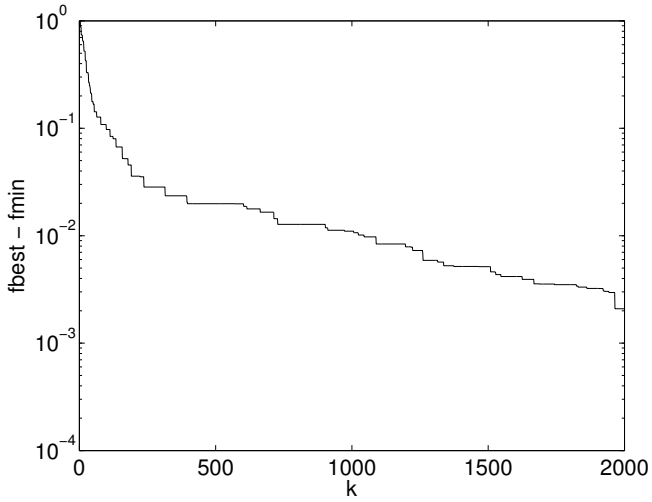


Fig. 4: The sub optimality gap $f_{best}^{(k)} - f(x^{(k)})$ versus iteration k

V. CONCLUSIONS

In this report, first we discussed the basics of ellipsoid optimization, an iterative method for minimizing convex functions. We looked into the initialization, optimality and termination criterion for the method. Then we solved 2 examples using ellipsoid method. Lastly we explained the time complexity of the method.

References

[1] The Ellipsoid Algorithm for Linear Programming Lecturer: Sanjeev Arora, COS 521, Fall 2005 Princeton University Scribe Notes: Siddhartha Brahma
[2] Ellipsoid Method, S. Boyd, Notes for EE364b, Stanford University
[3] Ellipsoid method, Wikipedia
[4] Lecture notes on the ellipsoid algorithm, Massachusetts Institute of Technology,Combinatorial Optimization, Michel X. Goemans

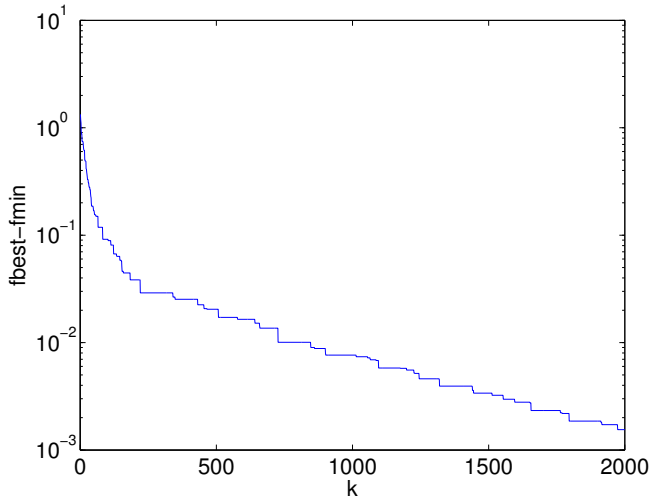


Fig. 5: The suboptimality gap $f_{best}^{(k)} f$ versus iteration k for the constrained ellipsoid algorithm.