CSCI5160 Approximation Algorithms — Spring 2017
Lecturer: Siu On Chan — *Scribe: Zhi Zhen Ye*

Notes 2: Ellipsoid Algorithm

This set of lecture notes follow [1].

1. Linear programs as satisfiability problems

Recall that a linear program (LP) takes the form

$$\begin{aligned} v^* = \max \quad & c^T x \\ & Ax \leqslant b \\ & x \geqslant 0 \end{aligned}$$

where $x, c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$ is a m-by-n matrix. Here x represents our LP variables, c represents our linear objective function, and $Ax \leqslant b$ are the linear constraints which means that $(Ax)_i \leqslant b_i, \forall i \in \{1, 2, \ldots, n\}$. The last inequality constraint $x \geqslant 0$ means that x has to be entry-wise nonnegative. Let

$$F \overset{\text{def}}{=} \{x \in \mathbb{R}^n \mid Ax \leqslant b \text{ and } x \geqslant 0\}$$

be the feasible region, which is the set only containing all x satisfying all constraints. We are interested in finding a maximizer $x^* \in F$ where $c^T x^* \geqslant c^T x, \forall x \in F$.

Note that we can perform a reduction from this optimization problem to a satisfiability problem. The way we do this is by doing a binary search for objective value. If we know that $v^* \in [low, up]$, then we can do the binary search by adding constraint $c^T x \geqslant t$ and see if $F \cap \{x \in \mathbb{R}^n \mid c^T x \geqslant t\}$ is non-empty. If so, we know $v^* \in [t, up]$. Otherwise, $v^* \in [low, t)$. And we can keep doing this iteratively. The number of times we incur the satisfiability solver is is $\mathrm{O}(\log_2(\frac{\text{up}-\text{low}}{\epsilon}))$, where ϵ is the precision required.

Algorithm 1: Linear Programs Solver

```
1  function solve ($A, b, c, low, up, \epsilon$)
   Input  : $A, b, c$ :=Parameter of LP
            $low$ := lower bound
            $up$ := upper bound
            $\epsilon$ := required precision, e.g. $10^{-4}$
   Output: near-optimal solution $x^*$ and near-optimal value $v^*$
2  $F := \{x \in \mathbb{R}^n \mid Ax \leqslant b \text{ and } x \geqslant 0\}$
3  $closest\_sol := \mathbf{0}$
   // Already within precision
4  if $up - low > \epsilon$ then
       // Assume feasible returns a bool representing feasibility and a feasible
          solution
5      $is\_feasible, x :=$ feasible$(F \cap \{x \in \mathbb{R}^n \mid c^T x \geqslant low\})$
6      return $x, c^T x$
7  while $up - low > \epsilon$ do
8      med := $\frac{low+up}{2}$
9      $is\_feasible, x :=$ feasible$(F \cap \{x \in \mathbb{R}^n \mid c^T x \geqslant med\})$
10     if $is\_feasible$ then
11         $low \leftarrow c^T x$
12         $closest\_sol \leftarrow x$
13     else
14         $up \leftarrow$ med
15     end
16 end
17 return $closest\_sol, low$
```

Before we proceed, let's introduce the task of checking for feasibility, and a related task of checking whether the constraints can be robustly satisfied.

Problem 1.1 (Feasibility)**.** Given a set of constraints C, consider the feasible set $F \overset{\text{def}}{=} \{x \in \mathbb{R}^n \mid x \text{ satisfies all constraints in } C\}$. If $F \neq \emptyset$, then return any $x \in F$. Otherwise, output "F is infeasible".

Problem 1.2 (Robust feasibility)**.** Given a set of constraints C and a length parameter $r \in \mathbb{R}^+$, consider the feasible set $F \overset{\text{def}}{=} \{x \in \mathbb{R}^n \mid x \text{ satisfies } C\}$. If F contains any cube of length r, return any $x \in F$. Otherwise, output "F is infeasible".

We claim (without giving any details) that to approximately solve a linear program, it suffices to solve robust feasibility.

2. Ellipsoid Algorithm

Algorithm 2: Ellipsoid Algorithm

```
1  function feasible ($C$)
   Input  : $C$ :=The set of constraints
   Output: A feasible solution x satisfying all constraints in C
2  $F := \{x \in \mathbb{R}^n \mid x$ satisfies all constraints in $C\}$
3  $E :=$ initialize as a ball of radius $R$, s.t. $F \subseteq E$
4  repeat
5      $c \leftarrow$ center of $E$
6      if $c \in F$ then return $c$
7      else Find a hyperplane to separate $c$ from $F$. Let $H$ be the corresponding halfspace s.t.
         $F \subseteq H$
8      $E \leftarrow$ smallest ellipsoid containing $E \cap H$
9  until $E$ is too small
10 return "Infeasible"
```

Remark 2.1.

(1) In Line 8, since $F \subseteq E$ and $F \subseteq H$, we know $F \subseteq E \cap H$. So the iterative process is clear.

(2) In terms of solving efficency, we have following claim (without proof):

Claim 2.2. *Vol(new ellipsoid)* $\leqslant (1 - \frac{1}{n^3})$ *Vol(original ellipsoid), where n is the dimension of* $\vec{x}$*, the variables.*

Therefore the ellipsoid E in the algorithm shrinks exponentially. Assume that the separating oracle (a black-box subroutine for finding separating hyperplane) is efficient (polynomial-time in terms of problem size). Checking whether constraints are satisfied or not in LP is also efficient. (Here we gloss over details about finding the smallest ellipsoid containing the intersection.) Therefore LP is (approximately) solvable in polynomial time.

(3) For LP, a separating hyperplane may be given by a violating constraint, that is a row A_i of A where $A_i^T c > b_i$ (and this must exist). Such a row can be found in polynomial time.

(4) For SDP, the variable is a matrix. If $X \succcurlyeq 0$ does not hold, then it has a negative eigenvalue. The associated eigenvector defines a separating hyperplane in this case.

References

[1] Ryan O'Donnell. Lecture 15, a theorist's toolkit. 2013.

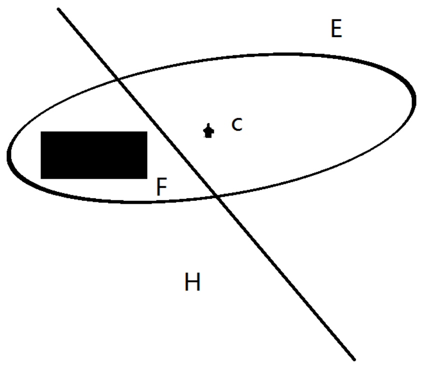

Figure 1. Ellipsoid in a single iteration