15-850: Advanced Algorithms **CMU, Spring 2017**
Lecture #17: Interior Point Methods March 4, 2017
Lecturer: Anupam Gupta Scribe: Guru Guruganesh, Nicholas Sieger

17.1 Ellipsoid Wrap-Up

17.1.1 Solving LPs with Ellipsoid

In the previous lecture, we showed that the Ellipsoid Algorithm can determine whether a full-dimensional convex polytope $K \subseteq \mathbb{R}^n$ is nonempty, but we still had to show how to actually solve an LP with Ellipsoid. That takes a bit more work and leads to the following theorem

Theorem 17.1. *Given an LP* $\{\min c^\intercal x : Ax \leq b\}$ *with representation size* $L := \langle A \rangle + \langle b \rangle + \langle c \rangle$ *Ellipsoid can find a basic feasible solution, if one exists, in time* $\text{poly}(L, n)$.

In lieu of the full (and lengthy) proof, we present a proof sketch. Set $K = \{x \in \mathbb{R}^n : Ax \leq b\}$, and define $K_\beta = \{x \in K : c^\intercal x \leq \beta\}$ for $\beta \in \mathbb{Q}$. Now, we binary search over β using the ellipsoid algorithm to determine whether K_β is nonempty. This raises three big problems:

1. Binary search over $\mathbb{Q}$ may not terminate, and even if it does terminate, it might produce exponential-size numbers
2. The Ellipsoid Algorithm needs a starting radius R such that $K_\beta \subseteq E_0$, where E_0 is an initial ellipsoid.
3. The Ellipsoid Algorithm also needs K_β to contain a ball of radius r entirely within it.

For the first issue, it can be shown that the optimal value of β only uses numbers of size $\text{poly}(L)$ (see Homework 4). Once $\langle \beta \rangle$ gets large enough, we find the closest point whose bit-size is bounded by $\text{poly}(L)$ and snap to it. Finding a bounded bit-size approximation to a point in high-dimensions is non-trivial. In one-dimension, we can use the theory of continued fractions to find such a point. These issues are addressed in detail in [GLS88].

For the second issue, we know that the optimal solution will have its size bounded by $\text{poly}(L)$. Hence, we take a ball of size $2^{\text{poly}(L)}$ as our starting ellipsoid.

The third problem is much harder to solve. One immediate requirement of this condition is that K must be full dimensional (in order to fit a tiny ball inside it). However, many LPs that we want to solve are not full dimensional, so we need to "fatten" K. In two dimensions, suppose K was a line-segment. In this case, it is clear that we could fatten it up by adding a small ball to each point, (the resulting shape would look like a small cylinder). In higher dimensions, this becomes very tricky. Much of the details of this procedure are worked out in [GLS88]. The book also contains many details on the relationship between seperation and optimizatoin.

17.1.2 Strong Separation Oracles

The Ellipsoid algorithm is extremely versatile because it does not need a description of a convex body but simple a (strong) seperation oracle.

Definition 17.2. Given a polytope K, a strong separation oracle (SSO) for K is a function $f_K(x)$ with the following property. For any $x \in \mathbb{R}^n$, $f_K(x)$ either reports that $x \in K$ or returns a hyperplane (a, b) such that

$$\begin{aligned} &a^\intercal y \leq b \quad \forall y \in K \\ &a^\intercal x > b \end{aligned}$$

So, if we can implement a poly-time SSO for an LP with an exponential number of constraints, the ellipsoid algorithm can solve this LP in polynomial time!

Example 17.3. Recall the perfect matching polytope for general (non-bipartite) graphs $G = (V, E)$

$$K_{PM} = x \in \mathbb{R}^{|E|} \text{ s.t. } \begin{cases} \forall v \in V,\ \sum_{u \in N(v)} = 1 & \text{and} \\ \forall S \text{ s.t. } |S| \equiv_2 1,\ \sum_{e \in \partial(S)} x_e \geq 1 & \text{and} \\ \forall e \in E,\ x_e \geq 0 \end{cases}$$

This polytope has an exponential number of constraints, yet we can optimize over it using the ellipsoid algorithm by exhibiting a SSO. We present a proof sketch here, and refer to homework 4 for the details.

Given a potential solution x, we can easily check the first constraint. Suppose some odd set $S \subset V$ with $\sum_{e \in \partial(S)} x_{uv} < 1$. The cut $(S, V/S)$ has value < 1, and we can run Min-Cut to find such an S. Note that Min-Cut does not guarantee that $|S|$ is odd. So there is a bit more work to be done, but this is the basic idea.

17.2 Interior Point Methods

The ellipsoid algorithm proved to be much worse than the simplex method in practice despite having theoretical guarantees. Progress was stalled until 1984, when Narendra Karmakar introduced his interior point algorithm [Kar84]. Karmakar showed it ran in weakly polynomial time. This led to a flurry of work which produced better interior point methods. Today interior point methods are used in practice on large LPs and are the method of choice for certain convex programs.

We present an algorithm from Matousek and Gaertner's [GM07] and Stephen Wright's [Wri97] books, which has a very elegant derivation.

17.2.1 Barrier Functions

Consider a general LP with $P = \{\min c^\intercal x : Ax \geq b\}$, which we assume is bounded. Instead of viewing this system as a linear program, we can think of it as a nonlinear optimization problem with this objective function

$$f_P(x) = \min_{x \in \mathbb{R}^n} c^\intercal x + 1_P(x)$$

$$1_P(x) = \begin{cases} \infty & x \notin P \\ 0 & x \in P \end{cases}$$

To actually solve this problem, we will need to take gradients of $f_P(x)$, and these gradients are discontinuous. So we will replace $1_P(x)$ with a *barrier function*

Definition 17.4. Given a convex body $P \subseteq \mathbb{R}^n$, and $x \in P$, let y be a point on the boundary of P. A barrier function defined on the interior of the polytope P $b_P : int(P) \to \mathbb{R}_+$ satisfies the following

$$x \to y \implies b_P(x) \to \infty$$

Think of barrier functions as a "smooth" version of the indicator function $1_P(x)$.

In our setting of an LP with constraints $a_i x \geq b_i$, we will take the barrier function $b_P(x) = \sum_{i=1}^m \log\left(\frac{1}{a_i^\intercal x - b_i}\right)$. Since we assumed our LP was bounded, $b_P(x) \to \infty$ only when x gets closer to the boundary.

Remark 17.5. As we will see, we will need the gradient of the barrier function and computing the gradient of the universal barrier function is essentially as hard as solving an LP. There are many different choices for barrier functions some of them allow fewer iterations (which can be expensive) while others require more iterations but faster computation per iteration. The current barrier function (known as the log barrier function) depends explicitly on the constraints of the polytope. Hence, repeating a constraint many times can force the algorithm to take many iterations. In a breakthrough, Nesterov and Nemirovski [NN94] showed the existance of a "universal barrier" function which only depends on the polytope. Recently Lee and Sidford [LS14] showed how to speed up this computation resulting in the fastest known algorithms for linear programming.

However, we do not want $b_P(x)$ to have too much influence on our solution, so we change our objective function once again.

$$f_\eta(x) = c^\intercal x + \eta\left(\sum_{i=1}^m \log\left(\frac{1}{a_i^\intercal x - b_i}\right)\right)$$
$$x_\eta = \text{argmin}_x f_\eta(x)$$

The goal of introducing η is to control how close we are to x^*; as $\eta \to 0$ our algorithm will choose xs which get closer and closer to x^*. Alternatively, if we increase η, the barrier function dominates, and we find points closer and closer to the center of P. As $\eta \to \infty$, the $x_\eta \to x_c$ which is known as the analytic center of P with respect to b_P.

At a high level, this interior point method does the following.

1. Pick some starting point x_{η_0}
2. Until η is sufficiently small,
   (a) Move to a new $x_{\eta(1-\epsilon)}$ while staying within P
   (b) Set η to $\eta(1-\epsilon)$

Once η is small enough, we can apply the same bit representation arguments as we did with the ellipsoid algorithm to find x^*. All that remains is to determine how we move from x_η to $x_{\eta(1-\epsilon)}$.

Remark 17.6. There is also the detail of how we choose an x_0. Ideally, x_0 would be close to P's center of gravity, but we have already seen that is hard to compute. However, we can run the interior point algorithm "in reverse".

The basic idea is that since an interior point method progressively finds a vertex, if we start with a vertex and increase η we will move closer and closer to the center of P. Since one definition of a vertex of P is a point which satisfies n constraints at equality, we can simply solve a linear system to find a vertex, and then increase η until we are satisfied the final point. Then this point is our x_0. For the remainder of these notes, we will simply assume that we can find an appropriate starting point.

17.2.2 Lagrange Multipliers

To ease the analysis, we will change our LP slightly to $\{\min_x c^\intercal x : Ax = b, x \geq 0\}$. So we are finding the minimizer of the following function as $\eta \to 0$

$$\lim_{\eta\to 0} \min_x f_\eta(x) = c^\intercal x + \eta \left(\sum_{i=1}^{m} \log \left(\frac{1}{a_i^\intercal x - b_i} \right) \right)$$

However, the gradient of $f(x, \eta)$ gives us no control over how the influence of each constraint. It would be nice if we could instead minimize the following objective function over $\{x \in \mathbb{R}^n : Ax = b\}$

$$\lim_{\eta\to 0} \min_{x:Ax=b} f_\eta(x) = c^\intercal x + \eta \sum_{i=1}^{n} \ln \left(\frac{1}{x_i} \right)$$

Now we will transform this into an unconditioned minimization problem using Lagrange Multipliers. (Lagrange Multipliers can be thought of as a penalty for violating constraints. For a more detailed explanation, see Wikipedia, [BV04].)

$$\begin{aligned} \lim_{\eta\to 0} \min_{x,y} f_\eta(x, y) &= c^\intercal x + \eta \sum_{i=1}^{n} \ln \left(\frac{1}{x_i} \right) - \sum_{j=1}^{m} y_i (a_i^\intercal x - b_i) \\ &= c^\intercal + \eta \sum_{i=1}^{n} \ln \left(\frac{1}{x_i} \right) - y^\intercal (Ax - b) \end{aligned} \tag{17.1}$$

At the minimizer, $\nabla_x f_\eta(x, y)$ and $\nabla_y f_\eta(x, y)$ are 0.

$$\begin{aligned} \nabla_x f_\eta(x, y) &= c^\intercal - \eta \frac{1}{x} - y^\intercal A \\ \nabla_y f_\eta(x, y) &= Ax - b \end{aligned}$$

Define $s = \eta(\frac{1}{x_1}, \frac{1}{x_2}, \ldots, \frac{1}{x_n})$, and setting the above to 0, we get

$$A^\intercal y + s = c \tag{17.2}$$
$$Ax = b \tag{17.3}$$
$$\forall i \in [n]\ x_i s_i = \eta \tag{17.4}$$

We have reduced the problem of minimizing $f_\eta(x, y)$ to find the solution of the above non-linear system.

Fact 17.7. *If (x, y, s) satisfy Equations 17.2, 17.3, 17.4, then $x = x^*$*

Proof.

$$c^\intercal x = (y^\intercal A + s^\intercal)x = y^\intercal Ax + s^\intercal x = y^\intercal b + 0$$

Since (17.2) is simply the dual LP, we can conclude that x is an optimal solution by strong duality. □

Unfortunately, (17.4) prevents us from easily solving this system exactly.

17.2.3 The Final Algorithm

Since we cannot solve (17.4) exactly, we will approximate it. For $x, y \in \mathbb{R}^n$ we use the notation $x \circ y = (x_1 \cdot y_1, x_2 \cdot y_2, \ldots, x_n \cdot y_n)$ and define $\mathbf{1}$ to the all-ones vector. We will then solve the following system

$$\begin{aligned} A^\intercal y + s &= c \\ Ax &= b \\ \|x \circ s - \eta 1\|_2 &\leq 0.4\eta \end{aligned} \tag{17.5}$$

Note that as $\eta \to 0$, $x \circ s \to 0$. To solve this system, we do the following

1. Start with $s_0, x_0 > 0$ and y_0, η_0 satisfying

$$A^\intercal y_0 + s_0 = c, \;\; Ax_0 = b$$
$$\eta_0 = \frac{\langle x_0, s_0 \rangle}{n}, \;\; \|x_0 \circ s_0 - \eta_0 1\|_2 \leq 0.4\eta_0$$

2. At each time t, we find $\Delta x, \Delta y, \Delta s$ such that

$$A\Delta x = 0 \tag{17.6}$$
$$A^\intercal \Delta y + \Delta s = 0 \tag{17.7}$$
$$(s_t \circ \Delta x) + (x_t \circ \Delta s) = -(x_t \circ s_t) + \eta_t 1 \tag{17.8}$$

Then set $x_{t+1} = x_t + \Delta x, y_{t+1} = y_t + \Delta y, s_{t+1} = s_t + \Delta s$. Finally, set

$$\eta_{t+1} = \left(1 - \frac{0.4}{\sqrt{n}}\right) \eta_t$$

To show that this algorithm produces a feasible solution, we will show the following claims for all t. We provide a sketch of the overall proof here and postpone the proofs of each claim to the appendix.

1. $Ax_t = b$ $A^\intercal y_t + s_t = c$ This holds by our linear system solver.
2. $\|x_t \circ s_t - \eta_t 1\|_2 \leq 0.4\eta_t$
3. After $T = O(\sqrt{n}L)$ steps, $\eta_T \leq 2^{-\Theta(L)}$
4. $\langle \Delta x, \Delta s \rangle = 0$
5. $\eta_t = \frac{\langle x_t, s_t \rangle}{n}$
6. After $T = \sqrt{n}L$ steps, x can be rounded to x^*

Claims (1) and (2) show that we always maintain a feasible solution, and (3) shows that after $T = O(\sqrt{n}L)$ steps, η_T is as small of a number as we can represent in L bits. Thus, if we knew that the duality gap $c^\intercal x_T - b^\intercal y_T$ is at most η_T, we can round x_T to x^*. However, we can observe the following by our linear system solver

$$\text{duality gap} = c^\intercal x_T - b^\intercal y_T = c^\intercal x_T - (Ax_T)^\intercal y_T = \langle (c^\intercal - A^\intercal y_T), x_T \rangle = \langle s_T, x_T \rangle$$

So if we knew that $\eta_T \propto \langle s_T, x_T \rangle$, we could bound the duality gap. This is given by (5). To show (5), we can use the fact that $\langle x, y \rangle = \langle 1, x \circ y \rangle$, and then apply the update rules for x_t, s_t. This produces $\langle x_{t+1}, s_{s+1} \rangle = \langle 1, x_t \circ s_t + (x_t \circ \Delta s_t) + (s_t \circ \Delta x_t) + (\Delta x_t \circ \Delta s_t) \rangle$ which we could simplify if $\langle 1, \Delta x_t \circ \Delta s_t \rangle = 0$. That problem is handled by a simple technical lemma (3). Therefore, we can round x_T to x^* after $O(\sqrt{n}L)$ steps.

A Proof of claim 2

Proof. The proof is by induction on t. By construction, the claim holds for $t = 0$, so we may assume the claim holds for $t - 1$ and prove it for t.

$$\begin{aligned}
|x_t \circ s_t - \eta_t 1\|_2 &= |(x_{t-1} + \Delta x) \circ (s_{t-1} + \Delta s) - \eta_t 1\|_2 \\
&= \|(x_{t-1} \circ s_{t-1} - \eta_t 1) + (s_{t-1} \circ \Delta x + x_{t-1} \circ \Delta s) + \Delta x \circ \Delta s\|_2 \\
&= \|\Delta x \circ \Delta s\|_2
\end{aligned}$$

Set $s = s_{t-1}, x = x_{t-1}$ Define $D = \operatorname{diag}\left(\sqrt{\frac{x_i}{s_i}}\right)$, and note that

$$\|a \circ b\|_2 \le \frac{1}{2^{3/2}} \|a + b\|_2^2$$

Then,

$$\begin{aligned}
\|\Delta x \circ \Delta s\|_2 &= \|(D^{-1}\Delta x) \circ (D\Delta s)\|_2 \\
&\le \frac{1}{2^{3/2}} \|D^{-1}\Delta x + D\Delta s\|^2 \\
&= \frac{1}{2^{3/2}} \sum_i \frac{x_i}{s_i}(\Delta x_i)^2 + \frac{s_i}{x_i}(\Delta s_i)^2 \\
&= \frac{1}{2^{3/2}} \sum_i \frac{(s_i^2 \Delta x_i)^2 + (x_i^2 \Delta s_i)^2}{s_i x_i} \\
&\le \frac{1}{2^{3/2}} \frac{\|x \circ \Delta s + s \circ \Delta x\|^2}{\min_i x_i s_i} \\
&= \frac{1}{2^{3/2}} \frac{\|\eta_t 1 - x \circ s\|^2}{\min_i x_i s_i}
\end{aligned}$$

$\min_i x_i s_i$ is maximized if $x_i s_i = x_j s_j$ for every i, j. So if we consider a vector which maximizes $\min_i x_i s_i$ and apply the inductive hypothesis,

$$(\min_i x_i s_i - \eta_{t-1})\|1\| \le 0.4\eta_{t-1}$$

$$\min_i x_i s_i \ge \left(1 - \frac{0.4}{\sqrt{n}}\right)\eta_{t-1}$$

Let $\sigma = \left(1 - \frac{0.4}{\sqrt{n}}\right)$. Then,

$$\begin{aligned}
\|\eta_t 1 - x \circ s\|^2 &= \|(x \circ s - \eta_{t-1} 1) + (1 - \sigma)\eta_{t-1} 1\|^2 \\
&= \|(x \circ s - \eta_{t-1} 1)\|^2 + 2\langle (x \circ s - \eta_{t-1} 1), (1 - \sigma)\eta_{t-1} 1 \rangle + \|(1 - \sigma)\eta_{t-1} 1\|^2
\end{aligned}$$

Let's look closer at the middle term

$$2\langle (x \circ s - \eta_{t-1}1), (1-\sigma)\eta_{t-1}1\rangle = 2(1-\sigma)\eta_{t-1}(\langle x \circ s, 1\rangle - \langle \eta_{t-1}1, 1\rangle)$$

By (5)

$$\begin{aligned}
&= 2(1-\sigma)\eta_{t-1}(\langle x, s\rangle - \frac{\langle x, s\rangle}{n}\langle 1, 1\rangle) \\
&= 2(1-\sigma)\eta_{t-1}(\langle x, s\rangle - \langle x, s\rangle) \\
&= 0
\end{aligned}$$

Thus we have

$$\|\eta_t 1 - x \circ s\|^2 = \|(x \circ s - \eta_{t-1}1)\|^2 + \|(1-\sigma)\eta_{t-1}1\|^2$$

By the inductive hypothesis

$$\begin{aligned}
&\leq 0.4^2\eta_{t-1}^2 + (1-\sigma)^2\eta_{t-1}^2 n \\
&= 0.4^2\eta_{t-1}^2 + \left(\frac{0.4}{\sqrt{n}}\right)^2 \eta_{t-1}^2 n \\
&= 2(0.4)^2\eta_{t-1}^2
\end{aligned}$$

Putting everything together gives us

$$\begin{aligned}
|x_t \circ s_t - \eta_t 1\|_2 &\leq \frac{2(0.4)^2\eta_{t-1}^2}{2^{3/2}\sigma\eta_{t-1}} \\
&= \frac{(0.4)}{\sqrt{2}\sigma^2}0.4\eta_t
\end{aligned}$$

It can be easily seen that the fraction is strictly less thatn 1

$$< 0.4\eta_t$$

□

B Proof of Claim 3

Proof. By the second step of the algorithm, $\eta_T = \left(1 - \frac{0.4}{\sqrt{n}}\right)^{\mathsf{T}} \eta_0$. Since $\eta_0 \leq 2^L$, there is some constant K such that

$$\eta_T \left(1 - \leq \frac{0.4}{\sqrt{n}}\right)^{K\sqrt{n}L} 2^L \leq \exp\left\{\frac{-0.4K\sqrt{n}L}{\sqrt{n}}\right\} 2^L = 2^{-\Theta(L)}$$

□

C Proof of Claim 4

Proof.

$$\langle \Delta x, \Delta s\rangle = -\langle A^{\mathsf{T}}\Delta y, \Delta x\rangle = -\Delta y^{\mathsf{T}} A \Delta x = 0$$

□

D Proof of Claim 5

Proof.

$$\begin{aligned}
\langle x_t, s_t \rangle &= \langle 1, x_t \circ s_t \rangle \\
&= \langle 1, (x_{t-1} + \Delta x) \circ (s_{t-1} + \Delta s) \rangle \\
&= \langle 1, (x_{t-1} \circ s_{t-1}) + (s_{t-1} \circ \Delta x) + (x_{t-1} \circ \Delta s) + (\Delta x \circ \Delta s) \rangle \\
&= \langle 1, \eta_t 1 + (\Delta x \circ \Delta s) \rangle \\
&= \eta_t n + \langle 1, \Delta x \circ \Delta s \rangle \\
&= \eta_t n + \langle \Delta x, \Delta s \rangle \\
&= \eta_t n \text{ By } (4)
\end{aligned}$$

□

E Proof of Claim 6

Proof. By (5),

$$\eta_T = \frac{\langle x_T, s_T \rangle}{n} = \frac{\langle x_T, c - A^\intercal y_T \rangle}{n} = \frac{1}{n} (c^\intercal x_T - b^\intercal y_T) = \frac{1}{n}(\text{duality gap})$$

By (3), $\eta_T \leq 2^{-\Theta(L)}$, so x_T, y_T are nearly optimal with an error of $2^{-\Theta(L)}$. Thus we can round x_T to x^*. □

References

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization.* Cambridge university press, 2004. 17.2.2

[GLS88] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 1988. 17.1.1

[GM07] Bernd Gärtner and Jiří Matoušek. *Understanding and using linear programming.* Universitext. Springer, Berlin, 2007. 17.2

[Kar84] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984. 17.2

[LS14] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in o(sqrt(rank)) iterations and faster algorithms for maximum flow. In *Foundations of Computer Science (FOCS)*, pages 424–433. IEEE, 2014. 17.5

[NN94] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming.* SIAM, 1994. 17.5

[Wri97] Stephen J. Wright. *Primal-dual Interior-point Methods.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997. 17.2