CS602 Applied Algorithms **2019-20 Sem II**

Lecture 18 - II: March 24

Scribe: Priyansh Kimtee *Lecturer: Rohit Gurjar*

Now, we will see the ellipsoid method and an interior point method to solve linear programs.

1 Ellipsoid Method

The main advantage of this method over other two methods is that it does not require an explicit description of the Linear Program(LP), we just need a separation oracle.

1.1 Separation Oracle

Let $C \subseteq \mathbb{R}^n$ be a closed convex set and $x \in \mathbb{R}^n$ not in C. Then x and C can be strictly separated by a **separating hyperplane**.(As seen in earlier lectures).

Separation oracle either says that the given point is inside the convex body or it if outside then gives a certificate for this fact, which is a separating hyperplane.

1.2 Ellipsoid

A 2D ellipse centered at origin can be described as:

$$\frac{x_1^2}{b_1^2} + \frac{x_2^2}{b_2^2} \leq 1$$

where b_1 and b_2 are the lengths of major and minor axes.

An ellipsoid is a higher dimensional generalization of an ellipse. An n dimensional ellipsoid centered at origin can be described as :

$$\frac{x_1^2}{b_1^2} + \frac{x_2^2}{b_2^2} + ... + \frac{x_n^2}{b_n^2} \leq 1$$

where $b_1, b_2, \ldots, b_n$ are the lengths of the axes. To get an ellipsoid in a general orientation, one can apply a rotation transformation on the coordinates.

1.3 Feasibility

As we already know that Feasibility and Optimization are equivalent, we will just describe how the ellipsoid method is used to find a feasible point for the LP, i.e., a point inside the polytope.

Algorithm **Invariant:** An ellipsoid E that contains the given polytope. Let z denote the the center of the current ellipsoid.

1. Initially, we construct an ellipsoid such that the polytope is completely inside it. This can be done by taking a sphere centered at the origin with large enough radius. Initially $z = (0, 0, \ldots, 0)$.

2. We give the point z and the polytope (convex body) to the separation oracle.
   if the point is inside the polytope **then** done
   else it gives a separating hyperplane, say (a, b) such that $a^\top z > b$, but $a^\top x < b$ for each x in the polytope.

3. Next we find a new ellipsoid which contains that half of the current ellipsoid where the polytope lies i.e., $\{x \in E \mid a^\top x < b\}$.

4. Go to step 2 with the new ellipsoid E and its center z.

1.3.1 Running time

- If the initial conditions are good like the polytope should be in the initial sphere (which can be done with the help of constraints and estimating the distance of the farthest point from the origin in the polytope).
- Volume of the polytope should not be too small (reason shown in section 1.3.2)
- If the above condition are satisfied then the algorithm runs in polynomial time.

1.3.2 Analysis of ellipsoid algorithm

We need to show that the volume of the ellipsoid shrinks by a constant factor in each round. That is,

$$\frac{Vol(E_{i+1})}{Vol(E_i)} \leq \delta$$

for some $\delta < 1$. For any ellipsoid E_i and polytope P we can say $E_i \supseteq P$ therefore

$$Vol(P) \leq Vol(E_i)$$

. If the algorithm takes r rounds to finish then

$$\delta^r \geq \frac{Vol(P)}{Vol(E_0)}.$$

taking log on both the sides we get

$$r \leq \log_{\frac{1}{\delta}} \left(\frac{Vol(E_0)}{Vol(P)} \right).$$

Thus, the runtime is small when the volume of the polytope is large enough. In particular, we should have $Vol(P) \neq 0$, i.e., the polytope should be full dimensional.

Note: If the LP is explicit, the separation oracle is very easy, we just have to check each constraint for a point one by one and if the point is not feasible then one of the constraint give you the separating hyperplane. For many combinatorial problems, it turns out that there is an exponentially large number of constraints in the linear program. Ellipsoid method can still be used to optimize if a separation oracle can be designed.

Points to think:

- How to construct new ellipsoid which satisfy the required conditions.
- How to bound the volume of the ellipsoid in each iteration and also the volume of polytope.

2 Interior point method

The starting point of the interior point methods is the Newton Raphson Method. It also uses the duality of convex programs.

2.1 Newton-Raphson Method

Newton-Raphson method is used to find the roots of a polynomial function.

Consider a polynomial function $P(x)$,

- Start with a arbitrary point say x_0 and draw a tangent at this point on the curve $P(x)$. This tangent can be thought of as a linear approximation of $P(x)$ at x_0

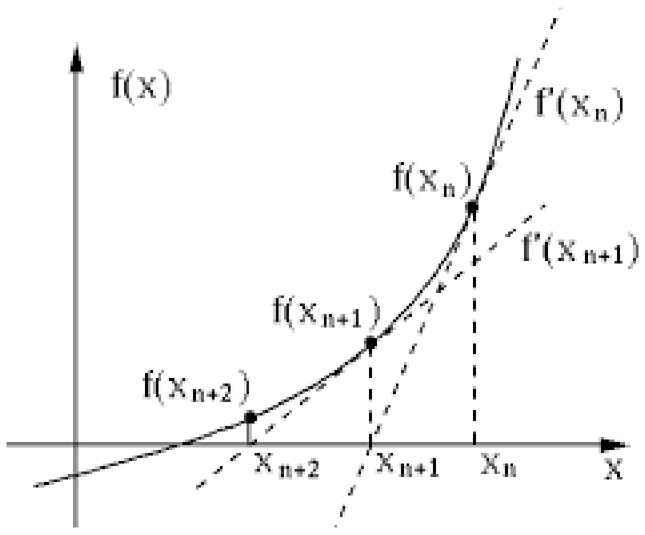

Figure 1: Newton-Raphson Method (Image source: `http://fourier.eng.hmc.edu/e176/lectures/NM/node20.html`)

- Next let the point where tangent intersect x-axis be x_1 (i.e., root of the linear approximation). If $P'(x)$ is the derivative of $P(x)$ then we can write

$$P'(x_0) = \frac{P(x_0)}{x_0 - x_1}$$

$$x_1 - x_0 = -\frac{P(x_0)}{P'(x_0)}$$

for any general $(i+1)^{th}$ step,

$$x_{i+1} - x_i = -\frac{P(x_i)}{P'(x_i)}$$

It is possible that this algorithm may not converge if starting point is not chosen correctly; it might get stuck in a cycle. So, the starting point needs to be chosen carefully. We are not going into the details of how to choose the staring point.

Multivariate Newton-Raphson: The Newton-Raphson method can be generalized to multivariate and multi-valued polynomials. Let $P : \mathbb{R}^n \to \mathbb{R}^m$ be such a polynomial and we are looking for its root. In other words, we want to solve the following system of polynomial equations.

$$\begin{aligned} P_1(x_1, x_2, \ldots, x_n) &= 0 \\ P_2(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ P_m(x_1, x_2, \ldots, x_n) &= 0 \end{aligned} \tag{1}$$

To run Newton-Raphson method, we need the multi-variate analogue of the derivative, which is given by the following Jacobian matrix $J(x)$.

$$\begin{bmatrix} \frac{\partial P_1}{\partial x_1} & \frac{\partial P_1}{\partial x_2} & \cdots & \frac{\partial P_1}{\partial x_n} \\ \frac{\partial P_2}{\partial x_1} & \frac{\partial P_2}{\partial x_2} & \cdots & \frac{\partial P_2}{\partial x_n} \\ & & \vdots & \\ \frac{\partial P_m}{\partial x_1} & \frac{\partial P_m}{\partial x_2} & \cdots & \frac{\partial P_m}{\partial x_n} \end{bmatrix}$$

Similar to the univariate Newton-Raphson, we can write the following relation between the i^{th} and $(i+1)^{th}$ point. Let $J = J(x = x_i)$.

$$\begin{aligned} J(x_{i+1} - x_i) &= -P(x_i) \\ \implies \quad x_{i+1} - x_i &= -J^{-1}P(x_i) \qquad (2) \end{aligned}$$

If the matrix J is not invertible then we take its pseudo inverse J^+ where

$$\begin{aligned} J^+ &= (J^TJ)^{-1}J^T \\ \therefore \qquad x_{i+1} - x_i &= -J^+P(x_i) \end{aligned}$$

Now, suppose we want to use Newton-Raphson to solve the following LP.

$$\begin{aligned} \min \quad c^\intercal x & \\ Ax = b & \qquad (3) \\ x \geq 0 & \qquad (4) \end{aligned}$$

Its dual LP would be:

$$\begin{aligned} \max \quad b^\intercal y & \\ A^\intercal y \leq c & \qquad (5) \end{aligned}$$

Recall that the complementary slackness is given as

$$x_i(a_i^\intercal y - c_i) = 0 \text{ for each } 1 \leq i \leq n, \qquad (6)$$

where a_i is the i-th column of A. First step is to do away with the *minimization* question. We know that any pair (x, y) of feasible solutions that also satisfy (6) will be optimal solutions. So, we can ignore the minimization/maximization question and just ask for a solution that simultaneously satisfies (3), (4), (5), and (6). Note that (6) is not linear but a degree-2 polynomial equation. And so Newton-Raphson comes into the picture. However, we cannot immediately apply Newton-Raphson for this as (4) and (5) are inequalities. The idea is to eliminate the inequality (4) and introduce a **Barrier Function** that helps us put the inequality (4) implicitly in the objective function. We can do this by creating a function that greatly increases the objective if a constraint is about to be violated.

Our new objective function will be $f(x) = c^\intercal x - \mu \sum_i \ln x_i$. Note that $-\ln x_i$ is increases when x_i decreases, and in fact, the rate of increase goes higher and higher as x_i goes close to zero. So, if we are in the quadrant $x > 0$ and are trying to minimize $f(x)$, we are forced to be away from the hyperplanes $x_i = 0$. However, by introducing this barrier function, we have changed our objective function. The parameter μ gives us a tradeoff: when μ is large, more weight is given to satisfying the inequality (3); when μ is small then $f(x)$ is close to being the actual objective function. Our new program becomes

$$\begin{aligned} \min \quad f(x) = c^\intercal x - \mu \sum_i \ln x_i & \\ \text{subject to } Ax = b. & \qquad (7) \end{aligned}$$

We can write the dual of this program similar to how we write it for linear programs. For LP, the c vector in the objective goes in the dual constraints. Here, instead of c, the partial derivative vector of the objective function will go in the dual constraints. Note that when $\mu = 0$, the derivative is just the vector c. The dual program is

$$\begin{aligned} \max \quad b^\intercal y \\ \text{subject to } A^\intercal y = \nabla f \end{aligned}$$

where ∇f is the partial derivative vector of f w.r.t. $x_1, x_2, \ldots, x_n$. That is,

$$\nabla f = (\ldots, c_i - \frac{\mu}{x_i}, \ldots)$$

So, the dual program can be written as

$$\begin{aligned} \max \quad & b^\intercal y \\ \text{subject to} \quad & a_i^\intercal y = c_i - \tfrac{\mu}{x_i} \quad \text{for } 1 \le i \le n. \end{aligned} \tag{8}$$

Again to get rid of the minimization/maximization question, we can use complementary slackness conditions, which is actually empty (as there are no non-negativity constraints on x and y). So, if we can just find a pair (x, y) that satisfies (7) and (8) simultaneously, they will be optimal solutions for the above programs. Hence, we need to solve the following system of polynomial equations.

$$\mu = x_i(c_i - a_i^\intercal y) \quad \text{for } 1 \le i \le n \tag{9}$$

$$Ax = b \tag{10}$$

Now, we can apply Newton-Raphson method to find a solution for the above system. We start with large value of μ as first we want to give more weightage to satisfying $x > 0$. With each Newton-Raphson step, we will slowly decrease μ, as small μ means our objective function is close to the actual objective. The following summarizes the algorithm: Fix $\rho < 1$.

1. Start with some x, y (as discussed before, starting point needs to be chosen carefully).
2. Take a Newton-Raphson step (take Jacobian of (9) and (10) and apply rule (2)).
3. Update $\mu \leftarrow \rho * \mu$ and go to 2.