Lecture 21

Interior Point Methods[*]

In 1984, Karmarkar introduced a new weakly polynomial time algorithm for solving LPs [Kar84a], [Kar84b]. His algorithm was theoretically faster than the ellipsoid method and Karmarkar made some strong claims about its performance in practice. The algorithm was controversial at the time of its introduction, but there have been many improvements both in theory and practice since then. The method is now considered to be better than simplex, especially on large LPs.

The method can in general be used to solve SDPs, cone programmming, etc., but in this lecture we will keep it simple and only discuss LPs. In particular, we will follow the framework of Renegar from 1988 [Ren88] which gives better theoretical performance than Karmarkar's original algorithm.

21.1 Setup and Algorithm

We consider the problem

$$\begin{array}{ll} \max & c^T x \\ s.t. & Ax \geq b \end{array}$$

and let $L = \langle A \rangle + \langle b \rangle + \langle c \rangle$. Assume A is $m \times n$, $m \geq n$ and that $P := \{x \mid Ax \geq b\}$ is non-empty and bounded. Assume $\beta <$ Opt and consider

$$P_\beta := \left\{x \mid ax \geq b, c^T x \geq \beta\right\}.$$

We define the "barrier function" for the constraint $a_i x \geq b_i$ to be $\ln(a_i x - b_i)$. In some sense, the constranint we care about most is the one corresponding to the objective function, so we will give that one more weight. This leads us to define

$$f_\beta(x) := m \ln(c^T x - \beta) + \sum_{i=1}^{m} \ln(a_i x - b).$$

[*]Lecturer: Ryan O'Donnell. Scribe: Deepak Bal.

This function is defined for x in the interior of P_β and we may say it is $-\infty$ on the boundary of P_β. We call x *valid* if $x \in \text{int}(P_\beta)$. We have given weight m to the constraint $c^T x \geq \beta$ and can think of this as the constraint added m times. One might want to consider putting other weights on the constraints but it turns out this is best, theoretically.

Remark 21.1. f_β is strictly concave which implies it has a unique maximizer, ω_β which we call the *(analytic) center of* P_β

One should note that ω_β and f_β are not actually functions of P_β, but rather functions of the representation of P_β. Extraneous constraints will affect their values.

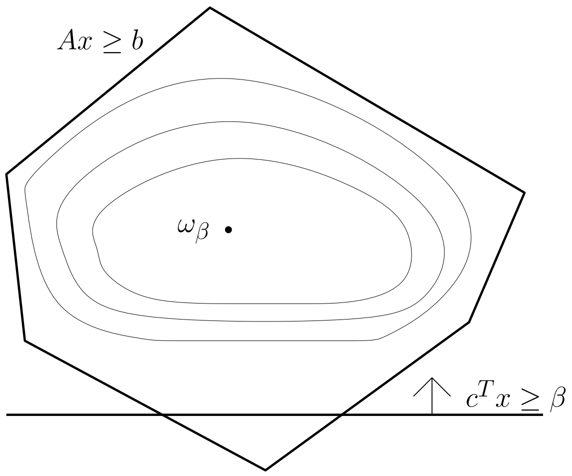

Figure 21.1: The polytope P_β with contour lines of f_β and analytic center ω_β.

The **rough algorithm** is as follows

0. Initialize $x^{(0)}, \beta^{(0)}$

For $j = 1, 2, 3, \ldots$

1. $\beta^{(j)} := \beta^{(j-1)} +$ a little.
2. $x^{(j)} :=$ approximation to new center $\omega_{\beta^{(j)}}$ of $P_{\beta^{(j)}}$ "starting from" $x^{(j-1)}$.

We only raise $\beta^{(j)}$ "a little" each step because if it is raised too much, then $\omega_{\beta^{(j)}}$ might move far from $\omega_{\beta^{(j-1)}}$ in which case, we would have to potentially do too much work in Step 2. Now we give a few more details in the **(more) precise algorithm**:

0. "Usual Tricks": Set $\beta^{(0)} = 2^{-O(L)}$ since we have proven earlier in the course that this is a bound on the optimum value. Make a new instance P' which is bounded and feasible and with the origin as the analytic center of $P'_{\beta^{(0)}}$. Finding the optimal value of this new instance should allow us to read off the optimal value for the original instance. Set $x^{(0)} = \omega_{\beta^{(0)}} = 0$, the center of the new polytope.

For $j = 1, 2, 3, \ldots$

1. Set $\beta^{(j)} := \beta^{(j-1)} + \delta(c^T x^{(j-1)} - \beta^{(j-1)})$ where $\delta = \frac{.05}{\sqrt{m}}$
2. Do one step of Newton's method for maximizing $f_{\beta^{(j)}}$ starting from $x^{(j-1)}$ to get $x^{(j)}$ (We will discuss this in more detail later).

In practice one might do multiple steps of Newton's method to get a better approximation of the center, but for theoretical purposes, one is enough.

21.2 Analysis

21.2.1 The Progress Theorem

Theorem 21.2. (Progress Theorem) *For every iteration j of the algorithm, we have*

1. *$x^{(j)}$ is valid for $P_{\beta^{(j)}}$*
2. $\left(\beta_{\text{Opt}} - \beta^{(j)}\right) \leq \left(1 - \frac{.01}{\sqrt{m}}\right)\left(\beta_{\text{Opt}} - \beta^{(j-1)}\right)$

where β_{Opt} refers to $c^T x^*$, the optimal value.

Corollary 21.3. *After $O(\sqrt{m}L)$ iterations, $\beta^{(j)}$ and $x^{(j)}$ are "within $2^{-O(L)}$" of* Opt *and we may round to get the optimum value.*

The details of the rounding are omitted but it involves the continued fraction representation of a point in the ending range. See Homework 2 solutions for more details.

So the total time for this method is

$$O(\sqrt{m}L) \times (\text{time to do one Newton step}).$$

"In practice", the number of iterations is typically bounded by something like $O(1)$ or $O(\log m)$, so people don't really spend a lot of time tying to improve that factor. A lot of past and current research concerns improving that second term, the time required to do one Newton step.

21.2.2 Newton's Method

Since f_β is strictly concave, finding the ω which maximizes f_β is the same as finding ω such that $\nabla f_\beta(\omega)$. Newton's method for maximizing a function given a starting point x is to essentially find the best fitting quadratic at x and maximize that. For a function of a single variable we have, by the Taylor expansion, that

$$f(x+h) \approx f(x) + hf'(x) + \frac{1}{2}h^2 f''(x).$$

This quadratic in h is maximized at $h = -\frac{f'(x)}{f''(x)}$ (the denominator is non-zero here if the function is strictly concave). So one step of Newton's method would move from x to $x+h$. Newton's method often refers to root finding and we see that the maximization algorithm is just finding the root of f'.

Now in multiple dimensions, things become more complicated but have a similar feel. One step of Newton's method will move from $\vec{x}$ to $\vec{x}+\vec{h}$ with

$$\vec{h} := -(Hf(x))^{-1} \cdot \nabla f(x)$$

where Hf is the Hessian matrix of f (the ij entry of Hf is the ij mixed partial of f). So one can see how similar in spirit this looks to the expression for h in the single variable case.

Now for $f = f_\beta$, we can just write down all these things. After some computation we get

$$Hf(x) = A^T D^{-2} A + \frac{m}{c^T x - \beta} cc^T$$

where

$$D := \operatorname{diag}(a_1 x - b_1, \ldots, a_m x - b_m)$$

and

$$\nabla f(x) = \frac{a_1}{a_1 x - b_1} + \frac{a_2}{a_2 x - b_2} + \ldots + \frac{a_m}{a_m x - b_m} + m \frac{c}{c^T x - \beta}. \tag{21.1}$$

To compute $\vec{h}$ we have to pay the cost of a few matrix products and inversions to $O(L)$ bits of precision. So basically $O(n^3)$ arithmetic operations (each taking $\widetilde{O}(L)$ time). Sparse linear solvers help are useful in these computations. So Renegar's result will get us to $O(m^{3.5}L)$ operations, whereas Karamakar's original result took $O(m^4 L)$ operations. The current best known performance is a result of Vaidya from 1989 which takes $O(m^2)$ amortized time per iteration leading to a total of $O(m^{2.5}L)$ operations. This result uses the existence of matrix multiplication in time $O(n^{2.4})$.

21.2.3 Proofs

Say we have just finished the j^{th} iteration. Let $\beta = \beta^{(j)}, \omega = \omega^{(j)}, x = x^{(j)}$. In the $(j+1)^{th}$ iteration we let $\beta' = \beta^{(j+1)} = \beta + \delta$-fraction of the distance from β to $c^T x$. We can basically think of x as being really close to ω. After shifting β to β', the center ω moves to some ω' and we compute an approximation x' to the new center. We want ω far from y such that $c^T y = \beta$ that way a δ-fraction is a good chunk of the way to β_{Opt}. That's why we weighted the objective constraint so much.

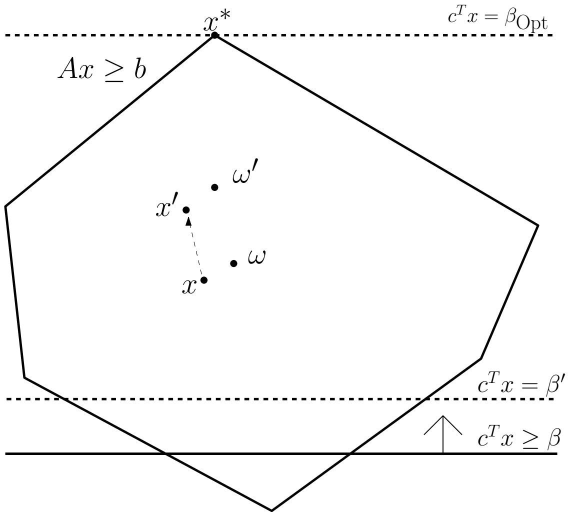

Figure 21.2: At each step, we approach the optimum.

Definition 21.4. For any $y \in \mathbb{R}^n$, $1 \leq i \leq 2m$,

$$Rat_i(y) := \frac{\text{signed distance of } y \text{ to } i\text{th hyperplane}}{\text{signed distance of } \omega \text{ to } i\text{th hyperplane}} = \begin{cases} \dfrac{a_i y - b_i}{a_i \omega - b_i} : & 1 \leq i \leq m \\ \dfrac{c^T y - \beta}{c^T \omega - \beta} : & m+1 \leq i \leq 2m \end{cases}$$

Note that we have y is valid for P_β if and only if $Rat_i(y) \geq 0 \, \forall i$, since the denominator is always positive. We have the following key property of Rat:

Lemma 21.5. *For any $y \in \mathbb{R}^n$,*

$$\operatorname*{avg}_{i=1}^{2m} \{Rat_i(y)\} = 1$$

Proof.

$$\begin{aligned} \operatorname*{avg}_{i=1}^{2m} \{Rat_i(y)\} = 1 &\iff \sum_{i=1}^{2m} Rat_i(y) = 2m \\ &\iff \sum_{i=1}^{2m} (Rat_i(y) - 1) = 0 \\ &\iff \sum_{i=1}^{2m} \frac{a_i(y-\omega)}{a_i\omega - b_i} = 0 \qquad \text{with } a_i = c, b_i = \beta \text{ for } i > m \\ &\iff \nabla f(\omega) \cdot (y - \omega) = 0 \end{aligned}$$

which is true since ω is the maximizer of f. □

Definition 21.6. Let $\epsilon_i(y) = 1 - Rat_i(y)$ for $i = 1, 2, \ldots, 2m$. Then $\sum_{i=1}^{2m} \epsilon_i(y) = 0$ for all y. Let

$$\text{err} = \text{err}^{(j)}(x) = \sqrt{\sum_{i=1}^{2m} \epsilon_i(x)^2}$$

This definition allows us to state the key technical lemma into which all the calculations and real hard work go.

Lemma 21.7. *At each iteration of the algorithm, we have the following invariant:*

$$err^{(j)} \leq .02 \implies err^{(j+1)} \leq .02$$

Note that $\text{err}^{(0)} = 0$ and that err $\leq .02$ implies x is valid since we get that $|\epsilon_i(x)| \leq .02$ for all i which implies that $Rat_i(x) \geq .98 > 0$. Not being too greedy in raising δ comes into play in the proof of this lemma.

Using these lemmas we can give a proof of the Progress Theorem.

Proof. (of Theorem 21.2) We have proved that $x^{(j)}$ is valid for $P_{\beta^{(j)}}$ by the argument above (since $Rat_i(x) \geq 0$ for all i). So we must prove the second property of the Progress Theorem. Let $\Delta := \beta_{\text{Opt}} - \beta$. Then we have

$$c^T\omega - \beta \geq \frac{\Delta}{2}.$$

To see this, let x^* be the optimum, so that $c^Tx^* = \beta_{\text{Opt}}$. Then x^* is in P_β so $Rat_i(x^*) \geq 0 \, \forall i$, and $\text{avg}_i \{Rat_i(x^*)\} = 1$. Hence

$$Rat_{m+1}(x^*) = \cdots = Rat_{2m}(x^*) \leq 2.$$

So $\dfrac{c^Tx^* - \beta}{c^T\omega - \beta} \leq 2$ and this is equivalent to the claim. Now err $\leq .02$ implies that $\sqrt{m\epsilon_{m+1}^2(x)} \leq .02$ by throwing away the first m terms and equating the last m terms. So $\epsilon_{m+1}(x) \leq \frac{.02}{\sqrt{m}}$. Hence

$$c^Tx - \beta \geq \left(1 - \frac{.02}{\sqrt{m}}\right)(c^T\omega - \beta) \geq \left(1 - \frac{.02}{\sqrt{m}}\right)\frac{\Delta}{2} \geq .49\Delta.$$

So to finish the proof, we note that

$$\begin{aligned}
\beta_{\text{Opt}} - \beta' &= \beta_{\text{Opt}} - \left(\beta + \delta\left(c^Tx - \beta\right)\right) \\
&\leq \beta_{\text{Opt}} - (\beta + .49\delta\Delta) \\
&= \beta_{\text{Opt}} - \beta - .49\frac{.05}{\sqrt{m}}(\beta_{\text{Opt}} - \beta) \\
&\leq \left(1 - \frac{.01}{\sqrt{m}}\right)(\beta_{\text{Opt}} - \beta)
\end{aligned}$$

□

So it remains to prove Lemma 21.7. We will sketch this below. We will make use of the following result. As expected, an apostrophe refers to the next step. So $Rat'(x), \epsilon'$ will refer to those quantities with resect to ω', β'.

Lemma 21.8.

$$\operatorname*{avg}_{i=1}^{2m}\left\{\frac{Rat_i(x)}{Rat'_i(x)}\right\} = 1 + \frac{1}{2}\left(\frac{1-\epsilon_{m+1}}{1-\epsilon'_{m+1}}\right)\delta\epsilon'_{m+1}$$

The proof of this lemma is about a page of calculations, but it is nothing crazy.

Proof. (Sketch of Lemma 21.7) The proof of this lemma comes in two parts.

1. $\mathrm{err}^{(j)}(x^{(j)}) \le .02 \implies \mathrm{err}^{(j+1)}(x^{(j)}) \le .1$
2. $\mathrm{err}^{(j+1)}(x^{(j)}) \le .1 \implies \mathrm{err}^{(j+1)}(x^{(j+1)}) \lesssim (.1)^2 \le .02$

All that we say about (2.) is that it is basically due to the fact that Newton's method has quadratic convergence. Really, we don't even need this much. We just need to show with a little bit of work that we improve by a factor of 5.

We now show (1.) modulo some small details. Examining the left hand side of Lemma 21.8 we have

$$\operatorname*{avg}_{i=1}^{2m}\left\{\frac{Rat_i(x)}{Rat'_i(x)}\right\} = \operatorname{avg}\left\{\frac{1-\epsilon_i}{1-\epsilon'_i}\right\} = \operatorname{avg}\left\{(1-\epsilon_i)\left(1+\epsilon'_i+(\epsilon'_i)^2+\dots\right)\right\}$$

by the geometric series expansion. Then ignoring higher order terms (this is not too bg of a lie) and using the facts that $\sum \epsilon_i = \sum \epsilon'_i = 0$ and $\sum \epsilon_i\epsilon'_i \le \sqrt{\sum \epsilon_i^2}\sqrt{\sum \epsilon_i'^2} \le .02\mathrm{err}'$ by Cauchy-Schwartz and the inductive hypothesis, we get that

$$\begin{aligned} LHS &\approx \operatorname{avg}\left\{1-\epsilon_i+\epsilon'_i-\epsilon_i\epsilon'_i-\epsilon_i'^2\right\} \\ &\ge 1 - \frac{1}{2m}\sum \epsilon_i\epsilon'_i + \frac{1}{2m}\sum \epsilon_i'^2 \\ &\ge 1 + \frac{1}{2m}\left(\mathrm{err}'\right)^2 - \frac{.02}{2m}\mathrm{err}'. \end{aligned}$$

On the other hand, for the right hand side of the lemma, we have

$$RHS \le 1 + .51\delta\epsilon'_{m+1} \le 1 + .51\delta\frac{\mathrm{err}'}{\sqrt{m}}$$

where the first inequality follows from the fact that $\frac{1-\epsilon_i}{1\epsilon'_i} \le \frac{1}{1-\delta}$ which is not too hard to show. The second inequality is because $\mathrm{err}' \ge \sqrt{m\epsilon'^2_{m+1}} = \sqrt{m}\epsilon'_{m+1}$

Putting these together, we see that

$$\frac{1}{2m}\mathrm{err}'^2 - \frac{.02}{2m}\mathrm{err}' \le .51\frac{.05}{\sqrt{m}}\cdot\frac{\mathrm{err}'}{\sqrt{m}}.$$

So

$$\mathrm{err}'^2 \le (.051+.02)\mathrm{err}',$$

and we conclude that $\mathrm{err}' \le .071 \le .1$ as desired. ∎

Bibliography

[Kar84a] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, pages 302–311, 1984.

[Kar84b] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[Ren88] James Renegar. A polynomial-time algorithm, based on newton's method, for linear programming. *Mathematical Programming*, 40:59–93, 1988.