ORIE 6300 Mathematical Programming I December 4, 2008

Lecture 26

Lecturer: David P. Williamson *Scribe: David Rowinski*

1 Interior-Point Methods for Linear Programming

Let us first review the main results from the previous lecture. Recall that our objective was to solve the LP:

$$\begin{aligned} \text{Min } & c^T x \\ Ax &= b \\ x &\geq 0. \end{aligned}$$

with dual

$$\begin{aligned} \text{Max } & b^T y \\ A^T y + s &= c \\ s &\geq 0. \end{aligned}$$

The main idea was to find a set of primal feasible and dual feasible points, and continually update these variables using a Newton method until complementary slackness was achieved. Recall that the Newton step is the solution to the system:

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -XSe + \sigma\mu e \end{pmatrix} \tag{1}$$

And the general primal-dual interior-point algorithm takes the following form:

Primal-Dual Interior-Point

$(x^0, y^0, s^0) \leftarrow$ initial feasible point $(x^0, s^0 > 0)$
$\mu^0 \leftarrow \frac{1}{n}(x^0)^T s^0$
$k \leftarrow 0$
While $\mu^k > \epsilon$
Solve $\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{pmatrix} \begin{pmatrix} \Delta x^k \\ \Delta y^k \\ \Delta s^k \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -X^k S^k e + \sigma^k \mu^k e \end{pmatrix}$
$(x^{k+1}, y^{k+1}, s^{k+1}) \leftarrow (x^k, y^k, s^k) + \alpha^k(\Delta x^k, \Delta y^k, \Delta s^k)$
where α^k is such that $x^{k+1}, s^{k+1} > 0$
$\mu^{k+1} \leftarrow \frac{1}{n}(x^{k+1})^T s^{k+1}$
$k \leftarrow k + 1$

where $\mu = \frac{x^T s}{n}$ and $\sigma \in [0, 1]$ is a centering parameter.

This general framework omits several details that must be addressed in any implementation of an interior-point algorithm for linear programming. In particular, we have not specified how the centering parameter σ^k is chosen, as different interior-point algorithms use different methods to select σ^k. Furthermore, we have not considered how to set the threshold ϵ, find an initial solution to the LP, or find the optimal solution to the LP given the solution that the algorithm returns.

1.1 Duality Gap and Termination

Suppose that in some iteration, the current solutions are (x, y, s). We consider the choice $\frac{1}{n}\sum_{i=1}^{n} x_i s_i = \frac{1}{n} x^T s$; that is, given our current solution, solving the system for this value makes the $x_i s_i$ equal for all i.

This quantity has a relation to the nearness to optimality since

$$x^T s = x^T(c - A^T y) = c^T x - b^T y.$$

Thus this is the difference between the primal and dual objective functions; we call this difference the *duality gap*. Since we have primal and dual feasible solutions, we know at optimality these two quantities are equal. Our algorithms will drive this quantity down to a small amount.

We define $\mu \equiv \frac{1}{n} x^T s$. While $\mu \geq \epsilon$, we apply Newton's method as illustrated previously. When $\mu < \epsilon$, the algorithm terminates, so computed solutions are within an additive $n\epsilon$ of optimal.

1.2 Centering Parameter

To balance the movement towards the central path against the movement toward optimal solutions, we maintain a *centering parameter* $\sigma \in [0, 1]$. If $\sigma = 1$, then our update will move towards the center of the feasible region. On the other hand, if $\sigma = 0$, then our update step is in the direction of optimal solutions to the linear programs. A step with $\sigma = 1$ is referred to as a centering step, and a step with $\sigma = 0$ is referred to as an affine-scaling step. The choice of the centering parameter σ provides us with a trade-off between moving towards the central path and moving toward optimal solutions to the linear programs.

1.3 Keeping Steps Bounded Away From Boundary of Feasible Region

During the course of iterating through a sequence of solutions to the linear programs, we can keep the solutions away from the boundary by ensuring that they remain in a *neighborhood* of the central path. Because solutions in the central path are essentially at the same distance from n boundaries of the feasible regions, by maintaining solutions near the central path, we can prevent them from approaching the boundaries of the feasible regions.

There are several common types of neighborhoods used by interior-point algorithms. For a parameter θ, a neighborhood that uses the L_2 norm to measure distance is defined as $N_2(\theta) = \{\text{strictly feasible } (x, y, s) \mid \|XSe - \mu e\| \leq \theta\mu\}$. Note that $\|XSe - \mu e\| \leq \theta\mu$ if and only if $\sum_{i=1}^{n}(x_i s_i - \mu)^2 \leq \theta^2\mu^2$. Figure 1(a) shows an example of a neighborhood $N_2(\theta)$.

1.4 Types of Interior-Point Algorithms

There are several major types of interior-point algorithms for linear programming.

- **Path-Following**: Path-following algorithms use update steps that follow the central path. The extent to which a path-following algorithm follows the central path is determined by the centering parameter σ. The method of choosing σ distinguishes different path-following algorithms.

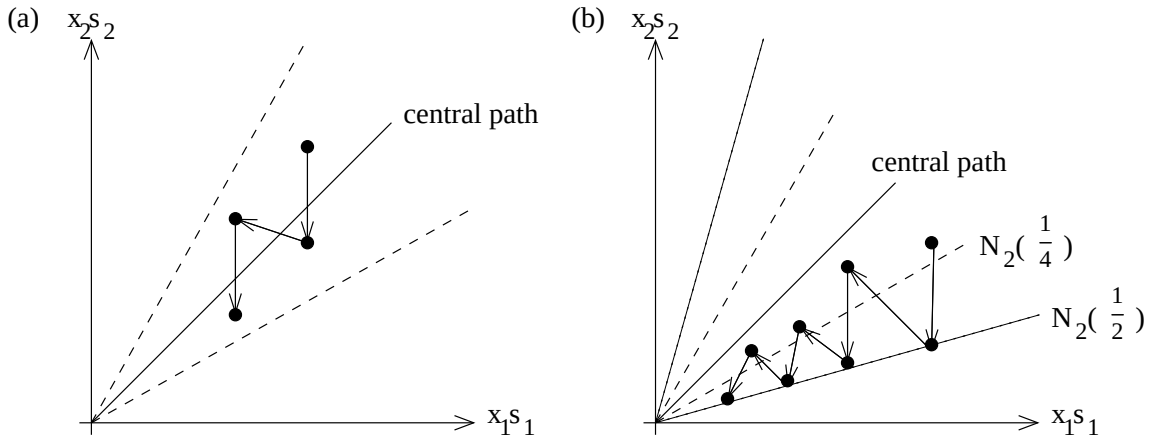

Figure 1: (a) A neighborhood of the central path in the case of $n = 2$ variables. (b) The Predictor-Corrector algorithm alternates between predictor steps, in which it moves as far as possible while remaining in $N_2(\frac{1}{2})$, and corrector steps, in which it takes a full step ($\alpha = 1$), returning to $N_2(\frac{1}{4})$.

  - **Short-Step**: In short-step algorithms, σ is set close to 1 so that the solutions stay near the central path. At most $O(\sqrt{n}\log\frac{1}{\epsilon})$ iterations are needed to achieve $\mu^k \leq \epsilon$. This is the best complexity bound known for an interior-point algorithm.
  - **Long-Step**: In contrast to short-step algorithms, long-step algorithms pick the centering parameter σ to be farther from 1, and as a result the solutions are farther from the central path. The number of iterations required to reduce μ^k so that it is below the threshold ϵ is $O(n\log\frac{1}{\epsilon})$, but long-step algorithms perform better than short-step algorithms in practice.
  - **Predictor-Corrector**: Predictor-corrector algorithms strike a balance between following the central path and moving toward optimal solutions by alternating between steps with $\sigma = 1$ and steps with $\sigma = 0$. These algorithms execute $O(\sqrt{n}\log\frac{1}{\epsilon})$ update iterations, and thus are as fast in theory as the best known interior-point algorithms. A variant on the predictor-corrector approach is the standard code used in practice.

- **Potential-Reduction**: Potential-reduction algorithms make use of a potential function. The potential function approaches $+\infty$ when $x_i s_i \to 0$ for some $i = 1, \ldots, n$, but $\mu \not\to 0$. This situation occurs when the solutions go near a boundary of the feasible region, but do not approach optimal solutions. The potential function approaches $-\infty$ if and only if the solutions (x, y, s) approach optimal solutions to the linear programs. One example of this is the potential function

$$\Phi_\rho(x, s) = \rho\log(x^T s) - \sum_{i=1}^{n}\log(x_i s_i)$$

  for a parameter $\rho > n$. Potential-reduction algorithms perform update steps to reduce the potential function, eventually moving toward optimal solutions. Karmarkar's seminal interior-point algorithm was a potential-reduction algorithm.

1.5 Short-Step Path-Following Algorithm

We now consider a short-step path-following algorithm, characterized by the parameters α and σ. We choose $\alpha^k = 1$ so that we take a full update step in the Newton direction each iteration. We choose the centering parameter to be $\sigma^k = 1 - \frac{0.4}{\sqrt{n}}$ at each iteration. Note that the centering parameter is near unity so that the steps taken by the algorithm are indeed short. We wish to establish a complexity bound for the short-step algorithm.

We first claim the following lemma, which states that as long as the initial points are within some neighborhood of the central path, feasibility is guaranteed, and we stay away from the boundary.

Lemma 1 *If* $(x^0, y^0, s^0) \in N_2(0.4)$*, then* $(x^k, y^k, s^k) \in N_2(0.4) \forall k$.

Next we address the issue of progress by showing that the duality gap goes down by a certain factor at each iteration.

Lemma 2 $\mu^{k+1} = \sigma^k \mu^k = (1 - \frac{0.4}{\sqrt{n}})\mu^k$.

We can use this lemma to prove the following theorem.

Theorem 3 *If the initial duality gap* $\mu^0 = C$*, then after* $k = O(\sqrt{n} \ln \frac{C}{\epsilon})$ *iterations,* $\mu^k \leq \epsilon$.

Proof: The duality gap is $\mu^k \leq (1 - \frac{0.4}{\sqrt{n}})^{(\frac{\sqrt{n}}{0.4} \ln \frac{C}{\epsilon})}$ after $k = \frac{\sqrt{n}}{0.4} \ln \frac{C}{\epsilon}$ iterations. Using the fact that $1 - x \leq e^{-x}$, it follows that $\mu^k \leq (e^{-\frac{0.4}{\sqrt{n}}})^{(\frac{\sqrt{n}}{0.4} \ln \frac{C}{\epsilon})} = e^{-\ln \frac{C}{\epsilon}} = (\frac{\epsilon}{C})C = \epsilon$ □

We can now prove Lemma 2 in somewhat more general form. We omit the superscripts k to make the proof more comprehensible.

Lemma 4 $\Delta x^T \Delta s = 0$ *and* $\mu' = (1 - \alpha(1 - \sigma))\mu$.

Proof: We know from (1) that

$$A^T \Delta y + \Delta s = 0$$

and

$$A \Delta x = 0.$$

Thus

$$\Delta x^T \Delta s = \Delta x^T (-A^T \Delta y) = -(A \Delta x)^T \Delta y = 0.$$

Again from (1),

$$\begin{aligned} & S\Delta x + X\Delta s = -XSe + \sigma\mu e \\ \Longrightarrow\ & s_i \Delta x_i + x_i \Delta s_i = -x_i s_i + \sigma\mu \ \forall i \\ \Longrightarrow\ & s^T \Delta x + x^T \Delta s = -x^T s + n\sigma\mu \end{aligned} \tag{2}$$

Hence,

$$\begin{aligned} \mu' &= \frac{1}{n}(x + \alpha\Delta x)^T (s + \alpha\Delta s) \\ &= \frac{1}{n}(x^T s + \alpha(\Delta x^T s + s^T \Delta x) + \alpha^2 \Delta x^T \Delta s) \\ &= \mu + \frac{\alpha}{n}(-x^T s + n\sigma\mu) \\ &= (1 - \alpha(1 - \sigma))\mu, \end{aligned} \tag{3}$$

where (3) follows from (2). $\square$

For the case of the short-step algorithm, $\alpha^k = 1$ and $\sigma^k = (1 - \frac{0.4}{\sqrt{n}})$ so $\mu^{k+1} = (1 - \frac{0.4}{\sqrt{n}})\mu^k$ as desired.