MIT OpenCourseWare
http://ocw.mit.edu

6.854J / 18.415J Advanced Algorithms
Fall 2008

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.

18.415/6.854 Advanced Algorithms October 27, 2008

Lecture 14

Lecturer: Michel X. Goemans

1 Introduction

For this lecture we'll look at using interior point algorithms for solving linear programs, and more generally convex programs . Developing originally in 1984 by Narendra Karmarkar, there have been many variants (with some of the keywords 'path following', 'primal-dual', 'potential reduction', etc.) on interior point algorithms, especially through the late 80s and early 90s. In the late 90s, people began to realize that interior point algorithms could also be used to solve semidefinite programs (or, even more generally, convex programs). As much as possible, we will discuss linear programming, semidefinite programming, and even a larger class called conic programming in a unified way.

2 Linear Programming

We will start with linear programming. Remember that in linear programming, we have:
Primal: Given $A \in \mathbb{R}^{m\times n}$, $c \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$, find $x \in \mathbb{R}^n$:

$$\begin{array}{rl} \text{Min} & c^T x \\ \text{s.t.} & Ax = b, x \geq 0. \end{array}$$

Its dual linear program is:
Dual: Find $y \in \mathbb{R}^m$:

$$\begin{array}{rl} \text{Max} & b^T y \\ \text{s.t.} & A^T y \leq c. \end{array}$$

We can introduce non-negative slack variables and rewrite this as:
Dual: Find $y \in \mathbb{R}^m$, $s \in \mathbb{R}^n$:

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

We know that, for a feasible solution, x in the primal, and a feasible solution (y, s) in the dual, we know by complementary slackness that they will both be optimal (for the primal and the dual resp.) iff $x^T s = 0$. Since this is the component-wise product of two non-negative vectors, we can equivalently say:

$$x_j s_j = 0 \qquad \forall j.$$

2.1 Using the Interior Point Algorithm

The interior point algorithm will iteratively maintain a strictly feasible solution in the primal, such that for all values of j, $x_j > 0$. Similarly in the dual, it will maintain a y and an s such that for all values of j, $s_j > 0$. Because of this strict inequality, we can never reach our optimality condition stated above; however, we'll get very close, and once we do, we can show that a jump from this non-optimal solution (for either the primal or the dual) to a vertex of improved cost (of the corresponding program) will provide an optimal solution to the (primal or dual) program.

In some linear programs, it may not be possible to start with a strictly positive solution. For example, for any feasible solution to the program, it may be that $x_j = 0$, so we may be unable to find a strictly feasible solution with which to start the algorithm. This can be dealt with easily, but we will not discuss this. We'll assume that the primal and dual both have strictly feasible solutions.

3 Semidefinite Programming

As introduced in the previous lecture, in semidefinite programming, our variables are the entries of a symmetric postitive semidefinite matrix X. Let S^n denote the set of all real, symmetric and $n \times n$ matrices. For two such matrices A and B, we define an inner product

$$A \bullet B = \sum_i \sum_j A_{ij} B_{ij} = Trace(A^T B) = Trace(AB).$$

Semidefinite programming (as a minimization problem) is

$$\begin{array}{lll} \text{Min} & C \bullet X & \\ \text{s.t.} & A_i \bullet X = b_i & i = 1...m \\ & X \succeq 0. & \end{array}$$

Remember that for a symmetric matrix M, $M \succeq 0$ means that M is positive semidefinite, meaning that all of its (real) eigenvalues $\lambda \geq 0$, or equivalently, $\forall x, x^T M x \geq 0$.

3.1 Dual for SDP

When working with linear programs, we know the existence of a dual linear program with a strong property: Any feasible dual solution provides a lower bound on the optimum primal value and, if either program is feasible, the optimum primal and optimum dual values are equal. Does a similar dual for a semidefinite progrm exist? The answer if yes, although we will need some additional condition. We claim that the dual takes the following form.

Dual: Find $y_i \in \mathbb{R}^n$, and $S \in S^n$:

$$\begin{array}{ll} \text{Max}_{y \in \mathbb{R}^m} & b^T y \\ \text{s.t.} & \sum_i y_i A_i + S = C \\ & S \succeq 0. \end{array}$$

3.1.1 Weak Duality

For weak duality, consider any feasible solution x in the primal, and any feasible solution (y, S) in the dual. We have:

$$\begin{aligned} C \bullet X &= \left(\sum_i y_i A_i + S\right) \bullet X \\ &= \sum_i y_i (A_i \bullet X) + S \bullet X \\ &= \sum_i y_i b_i + S \bullet X \\ &= b^T y + S \bullet X \\ &\geq b^T y, \end{aligned}$$

the last inequality following from Lemma 1 below. This is true for any primal and dual feasible solutions, and therefore we have $z \geq w$, where:

$$\begin{aligned} z &= \min\{C \bullet X : X \text{ feasible for primal}\}, \\ w &= \max\{b^T y : (y, S) \text{ feasible for dual}\}. \end{aligned}$$

Lemma 1 *For any $A, B \succeq 0$, we have $A \bullet B \geq 0$.*

Proof of Lemma 1: Any positive semidefinite matrix A admits a Cholesvky decomposition: $A = V^T V$ for some $n \times n$ matrix V. Thus,

$$A \bullet B = Trace(AB) = Trace(V^T V B) = Trace(V B V^T),$$

the last inequality following from the fact that, for (not necessarily symmetric) square matrices C and D, we have $Trace(CD) = Trace(DC)$. But VBV^T is positive definite (since $x^T V B V^T x \geq 0$ for all x), and thus its trace is nonnegative, proving the result. $\square$

A similar lemma was used when we were talking about linear programming, namely that if $a, b \in \mathbb{R}^n$ with $a, b \geq 0$ then $a^T b \geq 0$.

3.1.2 Strong Duality

In general, it's not true that $z = w$. Several things can go wrong.

In defining z, we wrote: $z = \min C \bullet X$. However, that min is not really a min, but rather an infimum. It might happen that the infimum value can be approached arbitrarily closely but no solution may attain that value precisely. Similarly in the dual, the supremum may not be attained.

In addition, in semidefinite programming, it is possible that the primal may have a finite value, but that the dual may be infeasible. In linear programming, this was not the case. If the primal had a finite feasible value and was bounded, the dual was also finite and with the same value. In semidefinite programming, the primal can be finite, while the dual may be infeasible or vice versa.

In addition, both the primal and dual could be finite, but they could be of differing values.

That all said, in the typical case, you do have strong duality ($z = w$), but only necessarily under certain conditions.

3.1.3 Introducing a Regularity Condition

Assume that the primal and dual have a strictly feasible solution. This means that for the primal:

$$\begin{aligned} \exists X \quad \text{s.t. } & A_i \bullet X = b_i \quad i = (1 \ldots m). \\ & X \succ 0. \end{aligned}$$

'$A \succ 0$' denotes that A is a positive-definite matrix, meaning that $\forall a \neq 0, a^T X a > 0$, or equivalently that all its eigenvalues λ_i satisfy $\lambda_i > 0$.

Likewise, in the dual, there exists y and S such that:

$$\begin{gathered} \textstyle\sum_i y_i A_i + S = C \\ S \succ 0. \end{gathered}$$

If we assume this 'regularity condition' that we've defined above, then the primal value z is finite and attainable (i.e. it is not an infimum, but actually a minimum), and the dual value w is attained and furthermore $z = w$. This is given without proof.

4 Conic Programming

Conic Programming is a generalization of both Linear Programming and Semidefinite Programming. First, we need the definition of a cone:

Definition 1 *A* cone *is a subset C of $\mathbb{R}^n$ that has the property that for any $v \in C$ and $\lambda \in \mathbb{R}^+$, λv is also in C.*

Conic Programming is constrained optimization over K, a *closed convex cone*, with a given inner product $\langle x, y\rangle$. We can, for example, take $K = \mathbb{R}^n$ and $\langle x, y\rangle = x^T y$ for any $x, y \in \mathbb{R}^n$; this will lead to linear programming. Conic programming, like LP and SDP, has both a primal and a dual form; the primal is:

Primal: Given $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$:

$$\begin{aligned} \min \quad & \langle c, x\rangle \\ \text{s.t.} \quad & Ax = b \\ & x \in K. \end{aligned}$$

More generally, we could view K as a cone in any space, and then A is a linear operator from K to $\mathbb{R}^m$. To form the dual of a conic program, we first need to find the *polar cone*, K^*, of K. The polar cone is defined to be the set of all s such that for all x in K, $\langle s, x\rangle \geq 0$. For instance, the polar cone of $\mathbb{R}^n_+$ is $\mathbb{R}^n_+$ itself (indeed if $s_j < 0$ then we have $s \notin K^*$ since $\langle e_j, s\rangle < 0$; conversely, if $s \geq 0$ then $\langle x, s\rangle \geq 0$). In the case that $K = K^*$, we say that K is *self-polar.* Similarly, the polar cone of PSD, the set of positive semidefinite matrices, is also itself.

We also define the *adjoint* (operator) A^* of A to be such that, for all x and y, $\langle A^* y, x\rangle = \langle y, Ax\rangle$. For example, if the inner product is a standard dot product and A is the matrix corresponding to a linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^m$, then $A^* = A^{\mathrm{T}}$. To write the conic dual, we introduce a variable $y \in \mathbb{R}^m$ and $s \in \mathbb{R}^n$ and optimize:

Dual:

$$\begin{aligned} \max \quad & \langle b, y\rangle \\ \text{s.t.} \quad & A^* y + s = c \\ & s \in K^*. \end{aligned}$$

4.0.4 Weak Duality

We can prove weak duality – that the value of the primal is at least the value of the dual – as follows. Let x be any primal feasible solution and (y, s) be any dual feasible solution. Then

$$\langle c, x\rangle = \langle A^* y + s, x\rangle = \langle A^* y, x\rangle + \langle s, x\rangle = \langle y, Ax\rangle + \langle s, x\rangle = \langle b, y\rangle + \langle s, x\rangle \geq \langle b, y\rangle,$$

where we have used the definition of K^* to show that $\langle s, x\rangle \geq 0$. This means that z, the infimum value of the primal, is at least the supremum value w of the dual.

4.0.5 Strong Duality

In the general case, we don't know that the two values will be equal. But we have the following statement (analogous to the regularity condition for SDP): if there exists an x in the *interior* of K, such that $Ax = b$, and a s in the interior of K^*, with $A^*y + s = c$, then the primal and the dual both obtain their optimal values, and those values are equal.

4.1 Semidefinite Programming as a Special Case of Conic Programming

LP is a special case of conic programming, if we let $K = \mathbb{R}^n_+$ and take the inner product to be the standard dot product $\langle a, b\rangle = a^{\mathrm{T}}b$. We can also make any SDP into a conic program; first, we need a way of transforming semidefinite matrices into vectors. Since we are optimizing over *symmetric* matrices, we introduce a map $svec(M)$ that only takes the lower triangle of the matrix (including the diagonal). To be able to use the standard dot product with these vectors, *svec* multiplies all of the off-diagonal matrices by $\sqrt{2}$. So *svec* maps X to

$$(x_{11}, x_{22}, \ldots, x_{nn}, \sqrt{2}x_{12}\sqrt{2}x_{13}, \ldots, \sqrt{2}x_{(n-1)n}).$$

As a result:

$$\langle svec(X), svec(Y)\rangle = \sum_{i=1}^{n} x_{ii}y_{ii} + \sum_{1\le i<j\le n} \sqrt{2}x_{ij}\sqrt{2}y_{ij} = \sum_{1\le i,j\le n} x_{ij}y_{ij} = Tr(AB) = A \bullet B.$$

This means that using the basic dot product as the inner product is compatible with the inner product used in SDP. So we can formulate an SDP as a conic program by letting $K = \{svec(X) : X \succeq 0\}$, which is a closed convex cone. To show convexity, we need to show that if A and B are matrices in PSD, then $\lambda A + (1-\lambda)B$ is also in PSD for $0 \le \lambda \le 1$. Indeed, for any vector v, we have

$$v^{\mathrm{T}}(\lambda A + (1-\lambda)B)v = \lambda\left(v^{\mathrm{T}}Av\right) + (1-\lambda)\left(v^{\mathrm{T}}Bv\right) \ge 0.$$

Then, we can let the matrix A be a matrix that is the composition of the corresponding A_i of the semidefinite program, so that

$$A\ svec(X) = (A_i \bullet X)_{i=1,\ldots,m}.$$

Now that the semidefinite program is cast into a conic program, we could write the conic dual, and one could verify that what we get is precisely the dual of the semidefinite program we defined earlier.

Instead of mapping the space of symmetric matrices (say $p \times p$) into $\mathbb{R}^n$ (with $n = \binom{p+1}{2}$) using $svec(\cdot)$, one could simply define $K = \{X \in S^p : X \succeq 0\}$ and $\langle X, Y\rangle = X \bullet Y$. Now our linear operator $A : S^n \to \mathbb{R}^m$ then maps X into $(A_i \bullet X)_{i=1,\cdots,m}$. Its adjoint $A^* : \mathbb{R}^m \to S^n$ is defined by:

$$\langle A^*(y), X\rangle := \langle y, A(X)\rangle = \sum_{i=1}^{m} y_iA_i \bullet X,$$

implying that A^* maps y to $\sum_{i=1}^{m} y_iA_i$. The dual SDP now arises as the dual conic program.

4.2 Barrier Functions

To solve the conic program, we will require a *barrier function* F. This is a function from $int(K)$, the interior of K, to $\mathbb{R}$ such that

1. F is strictly convex,
2. $F(x_i) \to \infty$ as $x_i \to x \in \partial K$, where ∂K is the boundary of K.

We will use the barrier function to "punish" candidate solutions that are close to the boundary of K, keeping the current point inside K. "Good" barrier functions, that result in a fast overall algorithm, have more properties that will be described in a later lecture. For $K = \mathbb{R}^n_+$, a good barrier function is

$$F(x) = -\sum_i \log(x_i).$$

As any one of the coordinates approaches 0, the log approaches $-\infty$, so the total function goes to ∞. One can also check that this function is strictly convex.

For $K = svec(PSD^p)$ or more simply $K = PSD^p$ (the set of symmetric $p \times p$ positive semidefinite matrices), the interior of K is the set of positive definite matrices, which all have strictly positive determinants. (This is because the determinant is equal to the product of the eigenvalues, which are all strictly positive for a positive definite matrix.) So we can use the following barrier function:

$$F(X) = -\log(\det(X)).$$

As X approaches the boundary of K, the determinant goes to zero, and F goes to infinity. One can also check that this function is strictly convex (its Hessian, the matrix of second derivatives, can be shown to be positive definite).

4.3 A Primal-Dual Interior-Point Method

Once we have a barrier function, we will set the objective function of the primal to $\langle c, x\rangle + \mu F(x)$, where μ is a parameter that we will adjust through the course of the algorithm. Assuming that we start with an initial candidate that belongs to $int(K)$, we can ignore the constraint that $x \in K$, since that will be enforced through the barrier function, since there will be an infinite penalty for leaving K. Our primal barrier problem $BP(\mu)$ will be:

$$\min\{\langle c, x\rangle + \mu F(x) : Ax = b\}.$$

Analogously, for the dual, we change the objective function to $\langle b, y\rangle - \mu F^*(s)$, where F^* is a barrier function for the dual; we can also eliminate the constraint that $s \in K^*$. Our dual barrier problem, $BD(\mu)$, is:

$$\max\{\langle b, y\rangle - \mu F^*(s) : A^* y + s = c\}.$$

The basic method of the algorithm is to have a current value of μ, and keep track of the optimal solutions in the primal $BP(\mu)$ and dual $BD(\mu)$. As long as μ is not zero, there is a unique optimum solution for both, since the objective function is the sum of a linear function and a strictly-convex function, which results in a strictly-convex function. We will steadily decrease μ, and keep track of the optimal solutions as they change; the paths the optimum solutions trace out is called the *central path* (or *central trajectory*). We will show that the (primal and dual) central paths will converge to an optimum value of the primal and dual original programs.

In the special case of linear programming, once we are sufficiently close, we can round the current solution to the nearest vertex to obtain an optimum solution. For semidefinite programming, though, we do not have such an algorithm to convert a solution for small enough μ to an optimum solution.

Let's characterize the optimum solution to $BP(\mu)$ and $BD(\mu)$. We derive now the so-called KKT optimality conditions. If there were no constraints in the conic program, then the minimum would be found when the gradient of the objective function is zero. If there are affine constraints like $Ax = b$, however, the minimum will occur when the gradient is normal to the affine space of feasible solutions. Otherwise, we could move along the projection of the gradient on the feasible space, and improve our objective function.

For simplicity, let's first look at the case when $K = K^* = \mathbb{R}^n_+$, and the barrier function is $F(x) = -\sum_i \log(x_i)$. The objective function of the primal is $\langle c, x\rangle - \mu F(x)$, and the partial derivatives are

$$\frac{\partial}{\partial x_j}\left(\langle c, x\rangle - \mu F(x)\right) = c_j - \frac{\mu}{x_j}$$

so the gradient is $c - \mu x^{-1}$, where x^{-1} denotes the vector $\{1/x_i\}$. But since this gradient is normal to the constraint Ax, the gradient must be of the form $A^{\mathrm{T}}y$ for some y. So if we let $s = \mu x^{-1}$, then we know $c - s$ is of the form $A^{\mathrm{T}}y$, or equivalently,

$$\begin{aligned} A^{\mathrm{T}}y + s &= c \\ s &= \mu x^{-1}. \end{aligned}$$

The last constraint is equivalent to

$$x_j s_j = \mu \tag{1}$$

for all j.

Now, looking at the dual: the gradient with respect to y is b, which must be of the form Ax for some x. The gradient with respect to s is μs^{-1}, which must equal the same x. This means that

$$\begin{aligned} Ax &= b \\ s &= \mu x^{-1}, \end{aligned}$$

and the last equality is again equivalent to (1).

So if we denote by $x(\mu)$ the optimum solution to the primal $BP(\mu)$ and by $(y(\mu), s(\mu))$ the optimum solution to the dual $BD(\mu)$, one observes that each of them is a certificate of optimality for the other and furthermore:

$$x_j(\mu)s_j(\mu) = \mu.$$

This means that the duality gap in the original primal/dual pair of linear programs is $x^T s = n\mu$ and therefore the duality gap goes to 0 as μ goes to 0. Thus the central path $(x(\mu), y(\mu), s(\mu))$ will converge to optimum solutions to both the primal and dual linear programs.