Semidefinite Programming
Lecture 6

scribe: Reuben Settergren

March 6 1995

Recall, from lecture two, the following facts about the Kronecker product:

1. $\mathbf{vec}(ABC) = (C^T \otimes A)\mathbf{vec}(B)$
2. $(AB) \otimes (CD) = (A \otimes C)(B \otimes D)$
3. $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$
4. If $\lambda_1 \ldots \lambda_n$ and $\mathbf{q}_1 \ldots \mathbf{q}_n$ are the eigenvalues and eigenvectors of A, and $\omega_1 \ldots \omega_m$ and $\mathbf{p}_1 \ldots \mathbf{p}_m$ are the eigenvalues and eigenvectors of B, the eigenvalues of $A \otimes B$ are $\lambda_i \omega_j$, with corresponding eigenvectors $\mathbf{q}_i \otimes \mathbf{p}_j$.

We can add here a definition of a Kronecker sum of matrices A and B, with the same dimensions: $A \otimes I + I \otimes B$. Notice that since any basis is a set of eigenvectors for I — namely eigenvectors of A, $\mathbf{q}_1 \ldots \mathbf{q}_n$, or eigenvectors of B, $\mathbf{p}_1 \ldots \mathbf{p}_n$ — by number 4 above, $A \otimes I$ and $I \otimes B$ share the common set of eigenvectors $\mathbf{q}_i \otimes \mathbf{p}_j$. From this, it is easy to see that the eigenvalues of $A \otimes I + I \otimes B$ are $\lambda_i + \omega_j$, all pairwise sums of eigenvalues of A and B.

1 Newton's method with LP

The lagrangian form for the solution of a standard Linear Program is:

$$\begin{aligned} A\mathbf{x} &= \mathbf{b} \\ A^T\mathbf{y} + \mathbf{z} &= \mathbf{c} \end{aligned}$$

along with any of the following sets of equations:

$$\begin{aligned} x_i z_i - \mu &= 0 \\ x_i - \frac{\mu}{z_i} &= 0 \\ z_i - \frac{\mu}{x_i} &= 0 \end{aligned}$$

To solve this system with Newton's method, replace $\mathbf{x}$ with $\mathbf{x}_0 - \Delta\mathbf{x}$, $\mathbf{y}$ with $\mathbf{y}_0 - \Delta\mathbf{y}$, and $\mathbf{z}$ with $\mathbf{z}_0 - \Delta\mathbf{z}$, and, ignoring products of Δ variables, solve. For instance, use the first set of equations to get:

$$\begin{aligned} A\Delta\mathbf{x} &= A\mathbf{x}_0 - \mathbf{b} \\ A^T\Delta\mathbf{y} + \Delta\mathbf{z} &= A^T\mathbf{y}_0 + z_0 - \mathbf{c} \\ (\mathbf{x}_0)_i\Delta\mathbf{x}_i + (\mathbf{z}_0)_i\Delta\mathbf{x}_i &= -\mu + (\mathbf{x}_0)_i(\mathbf{z}_0)_i + \Delta\mathbf{x}_i\Delta\mathbf{z}_i \end{aligned}$$

yields, after discarding $\Delta\mathbf{x}_i\Delta\mathbf{z}_i$ from the last equation, the linear system

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Diag(\mathbf{z}_0) & 0 & Diag(\mathbf{x}_0) \end{bmatrix} \begin{pmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \\ \Delta\mathbf{z} \end{pmatrix} = \begin{pmatrix} R \\ H \\ S \end{pmatrix}$$

It is this system of equations which requires solution for each iteration of an interior point LP algorithm.

2 Newton's method for SDP

Now we apply a similar procedure to the lagrangian form of a SDP:

$$\begin{aligned} \mathcal{A}\mathrm{vec}X &= \mathbf{b} \\ \mathrm{Mat}(\mathcal{A}^T\mathbf{y}) + Z &= \mathbf{c} \end{aligned}$$

along with any of the following sets of equations:

$$XZ - \mu I = 0 \tag{2.1}$$
$$Z - \mu X^{-1} = 0 \tag{2.2}$$
$$X - \mu Z^{-1} = 0 \tag{2.3}$$
$$XZ + ZX - 2\mu I = 0 \tag{2.4}$$

Similarly to the previous section, if X is replaced with $X - \Delta X$, $\mathbf{y}$ with $\mathbf{y}_0 - \Delta\mathbf{y}$, and Z with $Z_0 - \Delta Z$, the resulting system of equations will be:

$$\begin{bmatrix} \mathcal{A} & 0 & 0 \\ 0 & \mathcal{A}^T & I \\ E & 0 & F \end{bmatrix} \begin{pmatrix} \Delta X \\ \Delta\mathbf{y} \\ \Delta X \end{pmatrix} = \begin{pmatrix} R \\ H \\ S \end{pmatrix}, \tag{2.5}$$

where E and F depend on which final equation is chosen.

Case (1) (XZ method):

$$(X_0 - \Delta X)(Z_0 - \Delta Z) - \mu I = 0 X_0\Delta Z + \Delta X Z_0 = X_0 Z_0 - \mu I + \Delta X\Delta Z$$

Discarding the $\Delta X \Delta Z$, and recalling that $\mathbf{vec}((I)\Delta X Z_0) = (Z_0 \otimes I)\ \mathbf{vec}\ \Delta\ X$, we have

$$\begin{aligned} E &= Z_0 \otimes I \\ F &= I \otimes X_0 \end{aligned}$$

Case (4) ($XZ + ZX$ method):

$$(X_0 - \Delta X)(Z_0 - \Delta X) + (Z_0 - \Delta Z)(X_0 - \Delta X) - 2\mu I = 0$$

$$X_0 \Delta Z + \Delta Z X_0 + Z_0 \Delta X + \Delta X Z_0 = XZ + ZX - 2\mu I + \Delta X \Delta Z + \Delta Z \Delta X$$

Again discarding quadratic terms, we get:

$$\begin{aligned} E &= Z_o \otimes I + I \otimes Z_0 \\ F &= X_o \otimes I + I \otimes X_0, \end{aligned}$$

which are Kronecker sums.

case (2) Recall that $(I - X)^{-1} = I + X + X^2 + \ldots$.

$$\begin{aligned} (Z_0 - \Delta Z) &- \mu(X_0 - \Delta X)^{-1} = 0 \\ (Z_0 - \Delta Z) &- \mu[X_0^{\frac{1}{2}}(I - X_0^{-\frac{1}{2}} \Delta X X_0^{-\frac{1}{2}}) X_0^{\frac{1}{2}}]^{-1} \\ (Z_0 - \Delta Z) &- \mu[X_0^{-\frac{1}{2}}(I + X_0^{-\frac{1}{2}} \Delta X X_0^{-\frac{1}{2}}) X_0^{\frac{1}{2}} + \ldots] \\ (Z_0 - \Delta Z) &- \mu(X_0^{-1} - X_0^{-1} \Delta X X_0^{-1} + \ldots) \end{aligned}$$

Discarding the terms contained in "…", we get then that

$$\begin{aligned} E &= \mu X_0^{-1} \otimes X_0^{-1} \\ F &= I \end{aligned}$$

case (3) Similarly to case (2),

$$\begin{aligned} E &= I \\ F &= \mu Z_0^{-1} \otimes Z_0^{-1} \end{aligned}$$

Note that when using cases 2 and 3, as $\mu \to 0$, the big matrix in (2.5) approaches singularity, making algorithms that use these cases numerically unstable. Cases 1 and 4 do not have this difficulty, except in cases of degeneracy. And finally, note that while case 1 (XZ) need not be symmetric, case 4 ($XZ + ZX$) is.

In terms of actually solving the system of equations (2.5), note that the matrix is of dimension $(2n^2 + m) \times (2n^2 + m)$, so solving explicitly would be $O(n^6)$. The system can be solved analytically however, such that if m is small in relation to n (i.e. $m = o(n^2)$) we can solve the system more efficiently. Retaining our general notation for all four of the above mentioned cases, our linear system is as follows:

$$\begin{aligned} \mathcal{A}\mathbf{vec}\Delta X &= R && (2.6)\\ \mathcal{A}^T\Delta\mathbf{y} + \Delta Z &= H && (2.7)\\ E\mathbf{vec}\Delta X + F\mathbf{vec}\Delta Z &= S && (2.8) \end{aligned}$$

¿From (2.8) we can solve $\mathbf{vec}\Delta Z = F^{-1}S - F^{-1}E\mathbf{vec}\Delta X$. This we substitute into (2.7), and then left-multiply by $\mathcal{A}E^{-1}F$, so we can add (2.6) and cancel ΔX terms, leaving

$$(\mathcal{A}E^{-1}F\mathcal{A}^T)\Delta\mathbf{y} = (\mathcal{A}E^{-1}F)(H - F^{-1}S) + R \tag{2.9}$$

which is an $m \times m$ system. After solving for $\mathbf{y}$, ΔX and ΔZ can be easily found. But although it will take only $O(m^3)$ effort to solve for $\mathbf{y}$, we need to carefully consider the time necessary to calculate $\mathcal{A}E^{-1}F\mathcal{A}^T$. Consider, for instance, E and F in the XZ method:

$$\begin{aligned} E^{-1}F &= (Z_0 \otimes I)^{-1}(I \otimes X_0)\\ &= (Z_0^{-1} \otimes I)(I \otimes X_0)\\ &= Z_0^{-1} \otimes X_0 \end{aligned}$$

Recall also that the rows of $\mathcal{A}$ are $\mathbf{vec}$'s of the matrices A_i, so we can find $\mathcal{A}E^{-1}F$ with m operations of the form:

$$(\mathbf{vec}A_i)^T(Z_0^{-1} \otimes X_0) = \mathbf{vec}(X_0 A_i Z_0^{-1}),$$

which are each $O(n^3)$. This total computation of $O(mn^3)$ supersedes the subsequent time to calculate $(\mathcal{A}E^{-1}F)\mathcal{A}^T$ and the time to invert the resulting $m \times m$ matrix. We can see, then, that if, for instance, $m = O(n)$, with this method of calculating $\mathcal{A}E^{-1}F\mathcal{A}^T$ we can use (2.9) to solve (2.5) in only $O(n^4)$.

This is unfortunately not analagous to the LP case, where, when would be working with, the matrix $A\mathrm{Diag}(\frac{x_i}{z_i})A^T$. If AA^T is sparse, $A\mathrm{Diag}(\frac{x_i}{z_i})A^T$ has nonzero entries in exactly the same places as AA^T, so it is a simple matter to recalculate new matrices for new $\mathbf{x}$ and $\mathbf{z}$. While this is not the case for SDP, is it possible to take advantage of any special information about X_0 and Z_0 to help calculate $\mathcal{A}E^{-1}F\mathcal{A}^T$?

3 Possible Project

Implement the calculation of $\mathcal{A}E^{-1}F\mathcal{A}^T$ for each of the four cases of matrices E and F. Consider, for instance, the $XZ + ZX$ method, where we need the matrix

$$\mathcal{A}(Z_0 \otimes I + I \otimes Z_0)^{-1}(X_0 \otimes I + I \otimes X_0)\mathcal{A}^T.$$

Recalling, as before, that the rows of $\mathcal{A}$ are $\mathbf{vec}A_i$, $(X_0 \otimes I + I \otimes X_0)\mathcal{A}^T$ is easy to calculate (as above), but what about the rest? Inversion of $Z_0 \otimes I + I \otimes Z_0$, if done directly, would be very expensive, about $O(n^6)$. But what would be more useful are the vectors $(\mathbf{vec}A_i)^T(Z_0 \otimes I + I \otimes Z_0)^{-1}$. In general, we can see that

$$\mathbf{vec}U = (I \otimes Z_0 + Z_0 \otimes I)^{-1}\mathbf{vec}B$$

$$(I \otimes Z_0 + Z_0 \otimes I)\mathbf{vec}U = \mathbf{vec}B$$

$$\mathbf{vec}(Z_0U + UZ_0) = \mathbf{vec}B$$

$$Z_0U + UZ_0 = B.$$

This system is easy to solve for U if $Z = D$, a diagonal matrix. In that case, the entries of U can just be read off as $U_{ij} = B_{ij}/(d_i + d_j)$. Since we are dealing with symmetric matrices, we can transform the problem: factor $Z_0 = QDQ^T$. Then we can left- and right- multiply the entire equation $QDQ^TU + UQDQ^T = B$ by Q^T and Q, leaving a simple, transformed problem $D\overline{U} + \overline{U}D = \overline{B}$, where $\overline{X} = Q^TXQ$.

4 An Algorithm

In the remainder of this lecture, we will prepare for the introduction next lecture of an interior point algorithm for SDP with the following feature: every $\sqrt{n}$ iterations (each requiring the solution of an $m \times n$ system of linear equations) reduces the duality gap by a constant factor.

First we define some potential functions. Let X be feasible for a primal SDP ($\max C \cdot X st A_i \cdot X = b; X \succeq 0$), and $(\mathbf{y}, Z)$ feasible for it's dual, and let X and Z both be strictly positive definite. q is a scalar parameter.

Primal potential function: $\phi(X, \underline{z}) = q\ln(C \cdot X - \underline{z}) - \ln(\det X)$

Primal-Dual potential function: $\psi(X, Z) = q\ln(X \cdot Z) - \ln(\det XZ)$

THEOREM 4.1. *Suppose the following conditions hold:*

a) *The sequence $(X_0, \mathbf{y}_0, Z_0), (X_1, \mathbf{y}_1, Z_1), \ldots$ are all Primal-Dual feasible, and $X_i, Z_i \succ 0$*

b) $q = n + \lceil\sqrt{n}\rceil$

c) *There is a $\delta > 0$ so that for all i, $\psi(X_{i+1}, Z_{i+1}) \geq \psi(X_i, Z_i) - \delta$*

d) $\psi(X_0, Z_0) \leq E\sqrt{n}$ *for some constant E*

Then, for $k = O(\sqrt{n}|\log\epsilon|)$, $C \cdot X_k - \mathbf{b}^T\mathbf{y}_k < 2^E\epsilon$

Proof. Using c) and d) (and absorbing δ into the constant c),

$$\begin{aligned} \psi(X_k, Z_k) &\leq \psi(X_0, Z_0) - k\delta \\ &\leq E\sqrt{n} - c\sqrt{n}|\log\epsilon|\delta \\ &\leq \sqrt{n}(c\log 2^E\epsilon) \end{aligned}$$

Recall now the arithmetic-geometric mean inequality: $(\sum a_i)/n \geq (\prod a_i)^{\frac{1}{n}}$. Applying a logarithm and shifting things around yields,

$$n \ln(\sum a_i) - \ln(\prod a_i) \geq n \ln n$$

Let $a_i = \lambda_i(X_k Z_k)$. Note that these eigenvalues are all real, because, in general, for $A \succ 0, B \succeq 0$, the eigenvalues of AB are the same as the eigenvalues of $A^{\frac{1}{2}} B A^{\frac{1}{2}}$, which is symmetric. But $\sum \lambda_i(X_k Z_k) = \text{trace } (X_k Z_k) = X_k \cdot Z_k$, and $\prod \lambda_i(X_k Z_k) = \det(X_k Z_k)$, so

$$n \ln n + \sqrt{n} \ln(X_k \cdot Z_k) \leq \sqrt{n} \ln(2^E \epsilon)$$

Discarding an $n \ln n$, we get our desired result, $X_k Z_k \leq 2^E \epsilon$. ∎

5 Adapting LP's big M method

Consider the standard form SDP and it's dual

$$\begin{array}{ll} \min & C \cdot X \\ \text{s.t.} & \mathcal{A}\mathbf{vec}X = b \\ & X \succeq 0 \end{array} \qquad \begin{array}{ll} \max & \mathbf{b} \cdot \mathbf{y} \\ \text{s.t.} & \mathcal{A}^T \mathbf{y} + Z = C \\ & Z \succeq 0 \end{array}$$

This problem could be either infeasible or unbounded (dual infeasible). It would be useful to transform the problem so that it is guaranteed to be both primal and dual feasible. Suppose M, N are positive and big (to be specified later), and $X_0 \succ 0, Z_0 \succeq 0, \mathbf{y}_0$ are arbitrary. Consider the following SDP:

$$\begin{array}{ll} \min & C \cdot X + M x_1 \\ \text{s.t.} & \mathcal{A}\mathbf{vec}X + [\mathbf{b} - \mathcal{A}\mathbf{vec}X_0]\, x_1 = \mathbf{b} \\ & [\mathcal{A}^T \mathbf{y}_0 + Z_0 - C] \cdot X + x_2 = N \\ & X \succeq 0, x_1, x_2 \geq 0 \end{array} \tag{5.10}$$

And it's dual:

$$\begin{array}{ll} \max & \mathbf{b} \cdot \mathbf{y} - N z_1 \\ \text{s.t.} & \mathcal{A}^T \mathbf{y} + Z + [C - \mathcal{A}^T \mathbf{y}_0 - Z_0] z_1 = C \\ & [b - \mathcal{A}\mathbf{vec}X_0]\mathbf{y} + z_2 = M \\ & Z \succeq 0, z_1, z_2 \geq 0 \end{array} \tag{5.11}$$

The following solution is always feasible for the primal problem: $X = X_0$, $x_1 = 1$, $x_2 = N - [\mathcal{A}^T\ \mathbf{y}_0 + Z_0 - C] \cdot X$, provided N is big enough. Similarly , the following solution is always feasible for the dual problem: $\mathbf{y} = \mathbf{y}_0$, $Z = Z_0$, $z_1 = 1, z_2 = M - [b - \mathcal{A}\mathbf{vec}X_0]\mathbf{y} + z_2$, provided M is big enough. Just as in the big M method for LP, the trick is to solve the SDP, and if M is big enough, an optimial solution with nonzero x_1 indicates infeasibility.

In order to properly set this up, we need to choose N and M "big enough" beforehand. What we need, for instance (choosing arbitrarily $X_0 = Z_0 = I$, $\mathbf{y}_0 = 0$)

$$\begin{aligned} N &\geq [I - C] \cdot X^* \\ M &\geq \sum y_i^*(b_i - \text{trace } A_i) \end{aligned}$$

for all possible solutions $X^*, \mathbf{y}^*$. In LP, similar methods are successful because, for instance, the size of the input can give us a polynomial bound on the the number of bits in the answer. But in SDP, no such a priori bound exists. Consider, for instance the following quadratic program:

$$\begin{array}{ll} \min & x_n \\ \text{s.t.} & x_1 = 2 \\ & x_i \geq x_{i-1}^2 \end{array}$$

This is equivalent to a SDP (for instance, the quadratic constraint is equivalent to $\begin{pmatrix} x_i & x_{i-1} \\ x_{i-1} & 1 \end{pmatrix} \succeq 0$). It is clear that the optimum value is 2^{2^n}, so there is no polynomial size representation of the SDP's (5.10) and (5.11) we wish to solve, because N and M would have to have an exponential number of bits. It appears the only way around this is to require that a problem instance comes with an upper bound on the number of bits in the answer. Then an appropriate N and M can be calculated, and the augmented SDPs (5.10) and (5.11) solved. If the solution has either $x_1 \neq 0$ or $z_1 \neq 0$, there is no solution of the SDP *within the given bounds.* The SDP could be infeasible, or the feasible solutions could all lie outside of the given bounds.