CS787: Advanced Algorithms

Scribe: David Malec and Xiaoyong Chai — **Lecturer:** Shuchi Chawla

Topic: Minimax Theorem and Semi-Definite Programming — **Date:** October 22 2007

In this lecture, we first conclude our discussion of LP-duality by applying it to prove the Minimax theorem. Next we introduce vector programming and semi-definite programming using the Max-Cut problem as a motivating example.

16.1 L.P. Duality Applied to the Minimax Theorem

The Minimax Theorem relates to the outcome of two player zero-sum games. Assume we have a payoff matrix A for the game, where columns and rows represent moves that each of the players can employ (we refer to the players hereafter as the Column Player and the Row Player respectively), and the entries in the matrix represent the payoff for the Row Player. Note that we only need the one matrix to describe the outcome, since in a zero-sum game the payoff to the Column Player and the payoff to the Row Player must sum to zero. Now, if we call the strategy employed by the Row Player x and the strategy employed by the Column Player y, we can express the optimal payoff to the Row player as

$$\max_x \min_y x^T A y \qquad \text{if the Row Player goes first} \qquad (16.1.1)$$

$$\min_y \max_x x^T A y \qquad \text{if the Column Player goes first} \qquad (16.1.2)$$

The Minimax Theorem says that if we allow the Row Player and Column Player to select probability distributions over the rows and columns of A, respectively, for their strategies, then (16.1.1) is equal to (16.1.2). We can show this by considering a Linear Program for finding (16.1.1), determining its Dual, and then recognizing that the Dual will find (16.1.2). The Minimax Theorem then follows immediately from the Strong LP-Duality Theorem. Consider the Linear Programs

Primal:

$$\begin{array}{llll}
\max t \text{ subject to} & & & \\
t - \sum_j A_{ij} y_j & \le 0 & \forall i & : x_i \\
\sum_j y_j & \le 1 & & : s \\
y_j & \ge 0 & \forall j &
\end{array}$$

Dual:

$$\begin{array}{llll}
\min s \text{ subject to} & & & \\
s - \sum_i A_{ij} x_i & \geq 0 & \forall j & : y_j \\
\sum_i x_i & \geq 1 & & : t \\
x_i & \geq 0 & \forall i &
\end{array}$$

Note that while technically we require that $\sum_j y_j = \sum_i x_i = 1$, since both are probability distributions, the relaxations of this conditions seen in the Primal and Dual Linear Programs are sufficient, since $\sum_j y_j < 1$ and $\sum_i x_i > 1$ correspond to suboptimal values for their respective objective functions.

The preceding demonstrates the Minimax Theorem to be true, since with a little thought it can be seen that maximizing t in the Primal Linear Program corresponds exactly to finding the value of (16.1.1), and minimizing s in the Dual Linear Program corresponds exactly to finding the value of (16.1.2); hence, the Strong LP-Duality Theorem gives us that these values are in fact equal.

16.2 Max-Cut Problem

Some problems can not be reasonably approximated using Linear Programming, because there is no Linear Programming relaxation that both is of small size and has a good integrality gap. We can sometimes use the technique of Semi-Definite Programming to find approximations to such problems. We will consider the Max-Cut problem as an example of this type of problem. Formally, we can state the Max-Cut Problem as follows:

Given: a graph $G = (V, E)$, and a cost function on the edges c_e.

Goal: Form a partition (V_1, V_2) of V such that $\sum\limits_{e \in E'} c_e$ is maximized, where $E' = E \cap (V_1 \times V_2)$.

We can form a program for this problem by considering making a variable for each $v \in V$ to represent which of the sets V_1, V_2 contains it, say $x_v \in \{0, 1\}$, where a value of 0 indicates it is in V_1 and a value of 1 indicates it is in V_2. Then an edge (u, v) crosses the partition if and only if $x_u \neq x_v$, which is equivalent to the condition $|x_u - x_v| = 1$, by the definition of our variables. Thus, we can translate our original goal into an objective function to get the program:

$$\begin{array}{ll}
\max \sum_{(u,v) \in E} |x_u - x_v| c_{(u,v)} \text{ subject to} & \\
x_v \in \{0, 1\} & \forall v \in V
\end{array}$$

Looking more closely at this program, however, we can see that there is a fundamental issue with it — specifically, the objective function makes use of the absolute value function, which isn't linear. Previously, we have seen cases where we were able to convert an objective function containing an absolute value into a linear objective function by adding variables and constraints, so it might seem like we could do so here. The problem with that idea is that our objective function is to be maximized in this case. Before, we rewrote

$$\min|t|$$

as

$$\begin{aligned} &\min t' \text{ subject to} \\ &t' \geq t \\ &t' \geq -t \end{aligned}$$

This worked because we could think in terms of minimizing an upper bound on both t and $-t$. If we try the same approach with a maximization problem, however, we would be proposing to rewrite

$$\max|t|$$

as

$$\begin{aligned} &\min t' \text{ subject to} \\ &t' \geq t \\ &t' \leq -t \end{aligned}$$

This doesn't work, because the two constraints on t' conflict at a very basic level. Thus, if we want to solve this problem, we need to find another way to deal with the issue of nonlinearity in our program instead of trying to make the proposed program linear. So instead of focusing on this, we will instead focus on how to make it easier to relax the program, since this is necessary to finding an approximation regardless of our approach. We rewrite our program as an equivalent Quadratic Program, specifically

$$\begin{aligned} &\max \sum_{(u,v)\in E} \frac{1 - x_u x_v}{2} c_{(u,v)} \text{ subject to} \\ &x_v \in \{-1, 1\} \qquad \forall v \in V \end{aligned}$$

This is preferable to our previous program, since we can relax the integrality constraint $x_v \in \{-1, 1\} \qquad \forall v \in V$ to the constraint $x_v^2 = 1 \qquad \forall v \in V$.

Now, since the Max-Cut Problem is NP-Hard, we can see that this implies that Quadratic Programs, unlike Linear Programs, must be NP-Hard. There are certain types that we can find decent approximations to in a reasonable amount of time. We consider one such type next.

16.3 Vector Programs

A Vector Program is a program over vectors $y_1, y_2, \ldots, y_n$ (in general, these vectors are high-dimensional), where we write the program in the form

$$\min / \max \sum_{i,j} c_{ij} y_i \cdot y_j \text{ subject to}$$
$$\sum_{i,j} A_{ij} y_i \cdot y_j \geq b$$

In the above expressions, $(\cdot)$ represents taking a dot product. These can be thought of as Linear Programs in the dot products of the vectors they are over. While we can solve such programs, it is important to note that we can't constrain the dimension of the y_i — they could end up being very high-dimensional.

While we could reinterpret our original program for the Max-Cut Problem as a Vector Program, there are some very basic issues with this. Specifically, in order to be able to give a meaningful interpretation of the values of the x_v, we would need to require that they be essentially one-dimensional; however, as we just noted, we can place no such constraints on the number of dimensions the vectors in such a program have.

16.4 Semi-Definite Programming

One limitation of linear programming is that it cannot handle nonlinear constraints. Semi-definite programming, as a generalization of linear programming, enables us to specify in addition to a set of linear constraints a "semi-definite" constraint, a special form of nonlinear constraints. In this section, we introduce the basic concept of semi-definite programming.

16.4.1 Definitions

First Let us define a positive semi-definite (*p.s.d.*) matrix A, which we denote as $A \succeq 0$.

Definition 16.4.1 *A matrix $A \in \mathbb{R}^{n \times n}$ is positive semi-definite if and only if (1) A is symmetric, and (2) for all $x \in \mathbb{R}^n$, $x^T A x = \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} x_i x_j \geq 0$.*

Then we define a semi-definite program (SDP) as follows.

Definition 16.4.2 *An SDP is a mathematical program with four components:*

- *a set of variables x_{ij};*
- *a linear objective function to minimize/maximize;*
- *a set of linear constraints over x_{ij};*
- *a semi-definite constraint $X = [x_{ij}] \succeq 0$.*

The semi-definite constraint is what differentiates SDPs from LPs. Interestingly, the constraint can be interpreted as an infinite class of linear constraints. This is because by Definition 16.4.1, if

$X \succeq 0$, then for all $v \in \mathbb{R}^n$, $v^T X v \geq 0$. Each possible real vector v gives us one linear constraint. Altogether, we have an infinite number of linear constraints. We will use this interpretation later in this section.

16.4.2 A Different View of Semi-Definite Programming

In fact, semi-definite programming is equivalent to vector programming. To show this, we first present the following theorem.

Theorem 16.4.3 *For a symmetric matrix A, the following statements are equivalent:*

1. $A \succeq 0$.
2. *All eigenvalues of A are non-negative.*
3. $A = C^T C$, *where* $C \in \mathbb{R}^{m \times n}$.

Proof: We prove the theorem by showing statement 1 implies statement 2, statement 2 implies statement 3, and statement 3 implies statement 1.

$1 \Rightarrow 2$: If $A \succeq 0$, then all eigenvalues of A are non-negative.

Recall the concept of eigenvectors and eigenvalues. The set of eigenvectors ω for A is defined as those vectors that satisfy $A\omega = \lambda\omega$, for some scalar λ. The corresponding λ values are called eigenvalues. Moreover, when A is a symmetric matrix, all the eigenvalues are real-valued. Now let us look at each eigenvalue λ. First of all, we have $A\omega = \lambda\omega$, where ω is the corresponding eigenvector. Multiplying both sides by ω^T, then we have:

$$\omega^T A \omega = \lambda \omega^T \omega \tag{16.4.3}$$

The LHS of Equation 16.4.3 is non-negative by the definition of positive semi-definite matrix A. On the RHS of the equation, $\omega^T \omega \geq 0$. Thus $\lambda \geq 0$.

$2 \Rightarrow 3$: If all eigenvalues of A are non-negative, then $A = C^T C$ for some real matrix C.

Let Λ denote the diagonal matrix with all of A's eigenvalues:

$$\begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix}.$$

Let W denote the matrix of the corresponding eigenvectors $[\omega_1 \; \omega_2 \; \cdots \; \omega_n]$. Since eigenvectors are orthogonal to one another (*i.e.*, $\omega_i^T \omega_j = 1$ if $i = j$; 0, otherwise), W is an orthogonal matrix. Given Λ and W, we obtain the matrix representation of eigenvalues: $AW = W\Lambda$. Multiplying both side by W^T, we have:

$$AWW^T = W\Lambda W^T. \tag{16.4.4}$$

Since W is an orthogonal matrix, $WW^T = I$ and thus the LHS of Equation 16.4.4 is equal to A. Since all the eigenvalues are non-negative, we can decompose Λ into $\Lambda^{\frac{1}{2}}(\Lambda^{\frac{1}{2}})^T$, where $\Lambda^{\frac{1}{2}}$ is defined as:

$$\begin{bmatrix} (\lambda_1)^{\frac{1}{2}} & & 0 \\ & \ddots & \\ 0 & & (\lambda_n)^{\frac{1}{2}} \end{bmatrix}.$$

Thus $W\Lambda W^T = W\Lambda^{\frac{1}{2}}(\Lambda^{\frac{1}{2}})^T W^T = W\Lambda^{\frac{1}{2}}(W\Lambda^{\frac{1}{2}})^T$. Let $C^T = W\Lambda^{\frac{1}{2}}$. Equation 16.4.4 is equivalent to $A = C^T C$.

$3 \Rightarrow 1$: If $A = C^T C$ for some real matrix C, then $A \succeq 0$.

We prove it from the definition of A being *p.s.d.*:

$$\begin{aligned} A &= C^T C \\ \Leftrightarrow x^T A x &= x^T C^T C x, \quad \text{where } x \in \mathbb{R}^n \\ \Leftrightarrow x^T A x &= y^T y, \qquad \text{where } y = Cx \\ \Rightarrow x^T A x &\geq 0, \qquad \text{since } y^T y \geq 0 \end{aligned}$$

■

Note that by statement 3, a positive semi-definite matrix can be decomposed into $C^T C$. Let $C = [C_1, C_2 \cdots C_n]$, where $C_i \in \mathbb{R}^m$. Thus A_{ij} is the dot product of C_i and C_j. This gives us the following corollary.

Corollary 16.4.4 *SDP is equivalent to vector programming.*

16.4.3 Feasible Region of an SDP

Similar to an LP, a linear constraint in an SDP produces a hyper-plane (or a flat face) that restricts the feasible region of the SDP. The semi-definite constraint, which is nonlinear, produces a non-flat face. In fact, as we discussed at the end of Section 16.4.1, this nonlinear constraint can be interpreted as an infinite number of linear constraints. As an example, the feasible region of an SDP can be visualized as in Figure 16.4.3. In the figure, C_1, C_2 and C_3 are three linear constraints, and C_4 is the nonlinear constraint. Constraints C_a and C_b are two instances among the infinite number of linear constraints corresponding to C_4. These infinite linear constraints produces the non-flat face of C_4.

The optimal solution to an SDP can lie on the non-flat face of the feasible region and thus can be irrational. Below we give a simple example of an irrational optimum to illustrate this point.

Example: Minimize x subject to the constraint:

$$\begin{bmatrix} x & \sqrt{2} \\ \sqrt{2} & x \end{bmatrix} \succeq 0.$$

For positive semi-definite matrices, all the leading principal minors are non-negative. The leading principal minors of an $n \times n$ matrix are the determinants of the submatrices obtained by deleting

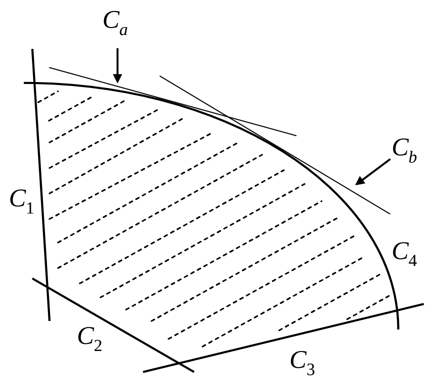

Figure 16.4.1: The feasible region of an SDP

the last k rows and the last k columns, where $k = n-1, n-2, \ldots, 0$. In the example, the matrix has two leading principal minors:

$$\begin{aligned} m_1 &= \left|[x]\right| = x, \quad \text{and} \\ m_2 &= \left| \begin{bmatrix} x & \sqrt{2} \\ \sqrt{2} & x \end{bmatrix} \right| = x^2 - 2. \end{aligned}$$

Thus we have $x \geq 0$ and $x^2 - 2 \geq 0$. It immediately follows that the minimum value of x is $\sqrt{2}$. ∎

Finally we state without proving the following theorem on SDP approximation.

Theorem 16.4.5 *We can achieve a (1+ϵ)-approximation to an SDP in time polynomial to n and $1/\epsilon$, where n is the size of the program.*