CSC2411 - Linear Programming and Combinatorial Optimization[*] Lecture 7: von Neumann minimax theorem, Yao's minimax Principle, Ellipsoid Algorithm

Notes taken by Xuming He

March 4, 2005

Summary: In this lecture, we prove the von Neumann's minimax theorem. We relate it to questions about the performance of randomized algorithms, and prove Yao's minimax principle. In the second part of lecture, we start to discuss the ellipsoid algorithm and show that it is a polynomial algorithm for linear programming problem.

1 von Neumann's Minimax theorem Cont'd

Recall the setting of the zero-sum game with 2 players we described last week. We have an $n \times m$ payoff matrix $A = (a_{ij})$. The entry a_{ij} represents the payoff to the row player when she picks the i^{th} strategy and column player picks his j^{th} strategy. At the same time, the payoff to the column player is $-a_{ij}$. Let y_i and x_j represent the probabilities of the row player and column player picking their i^{th} and j^{th} strategies, respectively (so the vector x,y correspond to *mixed strategies*). The resulting expected payoff is given by yAx. What happens if one of the players exposes her mixed strategy and let the other player choose the strategy of his liking? And if we reverse the order of the players? This question is answered by the celebrated theorem due to von Neumann, essentially saying the the order does not change the value of the game.

Theorem 1.1. *(von Neumann minmax theorem)*

$$\max_{x \in \Delta_m} \min_{y \in \Delta_n} yAx = \min_{y \in \Delta_n} \max_{x \in \Delta_m} yAx$$

Proof. Notice that for a chosen strategy x, the payoff $\min_{y \in \Delta_n} yAx$ is a simple linear programming problem with the constraint $\{y \geq 0, \sum_j y_j = 1\}$. Since the vertices of this polytope are the vectors $\{e_i\}_{i=1}^n$, where e_i is a vector with 1 at the ith location and

[*] Lecture Notes for a course given by Avner Magen, Dept. of Computer Sciecne, University of Toronto.

0 otherwise. In other words, we may always assume he second player always chooses a pure strategy to achieve the best payoff for herself. We have

$$\max_{x\in\Delta_m}\min_{y\in\Delta_n} yAx = \max_{x\in\Delta_m}\min_i (Ax)_i$$

Similarly,

$$\min_{y\in\Delta_n}\max_{x\in\Delta_m} yAx = \min_{y\in\Delta_n}\max_j (yA)_j$$

Thus, we only need to show the following to prove the theorem:

$$\max_{x\in\Delta_m}\min_i (Ax)_i = \min_{y\in\Delta_n}\max_j (yA)_j \tag{1}$$

We formulate LHS of equation (1) as a linear programming problem:

$$\begin{aligned}
\text{(Primal)}: \quad & \max \quad t \\
\text{s.t.} \quad & \sum_j a_{ij}x_j \ge t \\
& \sum_j x_j = 1 \\
& x \ge 0 \\
& t \gtrless 0
\end{aligned}$$

And its dual problem can be written as

$$\begin{aligned}
\text{(Dual)}: \quad & \min \quad w \\
\text{s.t.} \quad & y \ge 0 \\
& w \gtrless 0 \\
& w - \sum_i y_i a_{ij} \ge 0 \\
& \sum_i y_i = 1
\end{aligned}$$

It is easy to see that the primal is feasible (any $x \in \Delta_m$ and $t = \min_{ij} a_{ij}$ is feasible) and bounded (by $\max_{ij} a_{ij}$). Hence by duality theorem, the primal and dual problem have optimal solutions and at optimality, with identical optimal values. We rewrite the dual as

$$\begin{aligned}
\text{(Dual)}: \quad & \min \quad w \\
\text{s.t.} \quad & \sum_i y_i a_{ij} \le w \\
& y \in \Delta_m \\
& w \gtrless 0
\end{aligned}$$

which is $\min_{y\in\Delta_m}\max_j \sum_i y_i a_{ij}$, namely the RHS of equation (1). Consequently, the minimax theorem is proved. $\square$

Let x^*, y^* be optimal solutions to RHS and LHS of equation (1), y^*Ax^* is the common value of (1). From the above proof, we have $t^* = w^* = y^*Ax^*$. Applying the complementary slackness principle to the optimal solution, we have, for each i and j,

$$\sum_j a_{ij}x_j^* > t^* \rightarrow y_i^* = 0$$

$$\sum_i y_i^* a_{ij} < w^* \rightarrow x_j^* = 0$$

So if x^* is optimal, and I is the set of indices for the inequalities $t^* - \sum_j a_{ij}x_j^* \leq 0$ that are strict, then $y_i^* = 0$ for all $i \in I$. This is the same as saying that the y player cannot give any positive probability to a strategy that is not optimal against x^* and the suboptimum of i^{th} strategy implies $y_i^* = 0$. This holds similarly for the x player. The other direction holds as well, that is, if x, y satisfy those relations, they are both optimal.

2 Yao's minimax Principle

Consider a problem Π, assume that $\mathcal{I}$ is the set of all possible inputs of a given size and $\mathcal{I}$ is finite. Let $\mathcal{A}$ be the set of algorithm to Π, which is also assumed to be finite. Let the size of $\mathcal{A}$ be n and the size of $\mathcal{I}$ be m. Denote the time that algorithm A runs on I as $T(A, I)$. The expression

$$\min_{A \in \mathcal{A}} \max_{I \in \mathcal{I}} T(A, I)$$

is the worst case time for the best deterministic algorithm.

Now let's consider all the 'Las Vegas' randomized algorithms for the problem. These are algorithms which are always correct, and their running time is some random variable. Each such algorithm can be thought of as a distribution over $A \in \mathcal{A}$. Specifically, for a distribution q over $\mathcal{A}$, we let A_q be the randomized algorithm induced by q (i.e., A_{q_i} with probability q_i). Now $E[T(A_q, I)] = \sum q_i T(A_{q_i}, I)$, and by linearity of expectation,

$$\min_{q \in \Delta_n} \max_{I \in \mathcal{I}} E[T(A_q, I)],$$

is exactly the (expected) performance of the best randomized algorithm A_q.

Motivated by this expression and the setting in von Neumann minimax principle, we next consider the meaning of

$$\max_{p \in \Delta_m} \min_{A \in \mathcal{A}} E[T(A, I_p)].$$

This is picking a distribution p over input $\mathcal{I}$, denoted by I_p, and 'hoping' to get a bad expected runtime for any possible deterministic algorithm.

Notice that

$$E[T(A_q, I_p)] = \sum_{ij} p_i t_{ij} q_j = pTq,$$

where $t_{ij} = T(A_j, I_i)$. Viewing the algorithm as the column player, the input as the row player and the running time as the payoff, we have the following conclusion according to the minimax theorem:

$$\min_q \max_p E[T(A_q, I_p)] = \max_p \min_q E[T(A_q, I_p)]$$

Based on this, we can derive the following corollary, called *Yao's Minimax Principle*. It says the expected running time of the optimal deterministic algorithm for an arbitrarily chosen input distribution is a lower bound on the expected running time of the every randomized algorithm against all inputs.

Corollary 2.1. *(Yao's Minimax Principle) For all $p \in \Delta_m$ and $q \in \Delta_n$,*

$$\min_{A \in \mathcal{A}} E[T(A, I_p)] \leq \max_{I \in \mathcal{I}} E[T(A_q, I)]$$

Proof. This is the easy side of duality theorem:

$$\begin{aligned} \min_{A \in \mathcal{A}} E[T(A, I_p)] &\leq \max_p \min_{A \in \mathcal{A}} E[T(A, I_p)] = \max_p \min_q E[T(A_q, I_p)] \\ \text{(by minimax principle)} &= \min_q \max_p E[T(A_q, I_p)] = \min_q \max_{I \in \mathcal{I}} E[T(A_q, I)] \\ &\leq \max_{I \in \mathcal{I}} E[T(A_q, I)] \end{aligned}$$

□

Applying this principle, we can choose any distribution and produce a lower bound on the expected running time of all random algorithms on the worst input. The reduction to a lower bound on deterministic algorithms means we can analyze the optimal deterministic algorithm rather than the randomized one. Also, while this is indeed the easy side of duality theorem, the strong duality theorem actually tells us that there is a distribution over the input on which the best deterministic algorithm will perform exactly as the best randomized algorithm does against all inputs.

3 The Ellipsoid Algorithm

Until now, we know LP $\in$ NP $\bigcap$ coNP,[1] but we don't know whether LP $\in$ P. In 1979, Khachian proved the linear programming is polynomial based on a certain algorithm, called *Ellipsoid algorithm*, which will be discussed here.

The new algorithm is not combinatorial and is very different from the approach of the simplex algorithm. It is interesting mainly for theoretical reason, as it is not as good as the simplex algorithm on most inputs.

Since we can convert the LP in general form into standard form, we only need to consider the complexity of following problem:

$$\begin{aligned} \min \quad & (x, c) \\ & Ax = b, \quad x \geq 0 \end{aligned} \tag{2}$$

[1]As a decision problem, LP $= \{(A, b, c, \lambda) | \exists x, s.t., Ax = b, x \geq 0, (x, c) \leq \lambda.\}$. Clearly, LP $\in$ NP. LP $\in$ coNP since an optimal dual solution is a witness for $(A, b, c, \lambda) \notin$ LP

Under the assumption that all the coefficients are rationals, it can be shown that, as far as the existence of polynomial-time algorithm is concerned, the following decision problem (3)is as hard as the LP in (2) (cf. Assignment 1):

$$\{x|Ax = b, x \geq 0\} \overset{?}{=} \phi, \tag{3}$$

or equivalently,

$$\{x|Ax \leq b\} \overset{?}{=} \phi. \tag{4}$$

That is, we can use a subroutine which gives a solution to the above decision problem (3) or (4) to solve the optimality problem of LP in (2). We will show the above decision problem can be converted further to the following feasibility problem:

$$\{x|Ax < b\} \overset{?}{=} \phi. \tag{5}$$

Assume the LP problem has m constraints and n variables, and let L be the input size. The following lemma shows the relation between problem (4) and (5):

Lemma 3.1. *Let $P = \{x|Ax \leq b\}$, and $P_\epsilon = \{x|Ax < b + \epsilon \cdot \mathbb{1}\}$, where $\mathbb{1} = (1, ..., 1)^T$.*

(i) If $P_\epsilon = \phi$, then $P = \phi$;

(ii) If $P = \phi$, then $P_\epsilon = \phi$ for $\epsilon = 2^{-2L}$.

Proof. (i) is trivial since $P_\epsilon \supset P$.

(ii) If $P = \phi$, then according to Farkas's lemma, there is a vector y such that

$$yA = 0, \quad 0 > yb = -1, \quad y \geq 0.$$

Notice that the BFS of this feasible problem will be of size 2^L. Now the same y will satisfy $(y, (b + \epsilon \cdot \mathbb{1})) = yb + \epsilon \sum_i y_i < 0$, as long as $\epsilon < \frac{1}{n}2^{-L}$. So by Farkas's lemma, when $\epsilon_0 = \frac{1}{2n}2^{-L}$, the problem $P' = \{x|Ax \leq b + \epsilon_0 \cdot \mathbb{1}\}$ is infeasible. And since $\epsilon = 2^{-2L} < \epsilon_0$, $P' = \phi$ implies $P_\epsilon = \phi$. □

In the following discussion, we denote $P = \{x|Ax < b\}$. we need the conversion from (4) to (5) so that if $P \neq \phi$, it is full dimensional and we won't get a nonempty feasible region with zero volume. Also, we choose $\epsilon = 2^{-2L}$, so the size of the problem is about the same as the original input. From the lemma 3.1, if we can solve the problem in (5) polynomially, we can solve the original LP problem (2) in polynomial time.

How to solve $P = \{x|Ax < b\} \overset{?}{=} \phi$ polynomially? The basic idea is similar to the binary search: we start with a candidate region and cut the search space into two parts. We could reduce the search space by removing the part that doesn't intersect with the feasible region and continue the above procedure. If we can lower-bound the size of the feasible region (the volume of polytope) assuming it is nonempty, we will stop if either we find feasible solutions or the remaining region is smaller than that lower bound, which implies the feasible region does not exist. The whole procedure is similar to hunting a lion in the Sahara desert: we can always find the lion by the same procedure provided the lion doesn't move and isn't infinitely small.

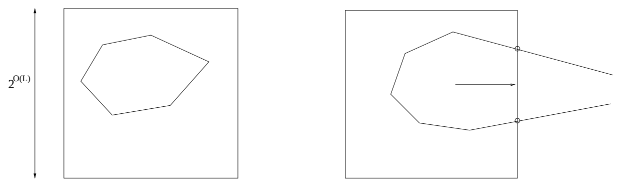

Figure 1: Left: The BFS's are bounded by a cube in the bounded feasible region case. Right: The BFS's and the bounding cube in the unbounded feasible region case.

First, how big is the initial search space we need to look at? If the feasible region is bounded, we can bound the BFS's by a cube with size $2^{O(L)}$, which also bounds the feasible region. If the feasible region is unbounded, it will intersect with that cube and the local optimum within the box achieves at the intersections. For the feasibility problem, we only need to find a region including all the bounded BFS's. So we may restrict the search to $\{x|Ax \leq b\} \bigcap [-2^{O(L)}, 2^{O(L)}]^n$ (cf. Figure 1).

Second, how do we get a lower bound to the volume of P, provided it is nonempty? By the fact that P is full dimensional, if it is nonempty, we know that there are $n+1$ BFS's in P that are affinely independent. Let $S = \{V_0, ..., V_n\}$ be $n+1$ affinely independent BFS's of P, the volume of the convex hull spanned by them is

$$\text{Vol}(\text{conv}(S)) = \frac{1}{n!} \left| \det \begin{pmatrix} 1 & \cdots & 1 \\ V_0 & \cdots & V_n \end{pmatrix} \right|$$

Since the determinant here is not 0 and its numerator is an integer, its absolute value can be lower-bounded by the inverse of the maximal denominator achievable. Notice that the denominators of the V_i are bounded by $2^{O(L)}$, and the determinant is a sum-product of its elements, so the lower bound of the volume is

$$\frac{1}{n!} 2^{-O(L)n} = 2^{-(O(L)n + \text{poly}(n))}.$$

Based the answers to the above two questions, we can describe the global view of the algorithm for the feasibility problem as follows: We start from a sphere (or ellipsoid) bounding the feasible region and select the center of the sphere (or ellipsoid). If it is feasible, we report that the problem is feasible and stop. Otherwise, that point is not in the feasible region and at least one inequality in the system is violated. We can use this inequality to divide the ball into two halves. Notice that one of them, say the first half, doesn't intersect with the feasible region so we can concentrate on the second half. Then we use a smaller ellipsoid to bound the second half of the ball and continue the above procedure. At all the time we maintain an ellipsoid which contains the feasible region, if it exists. At each iteration, we replace the current ellipsoid with a smaller one. After enough iterations, either we must discover a solution, or the ellipsoid has become too small to contain the feasible region (cf. Figure 2).

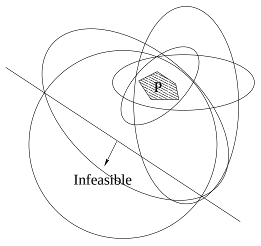

Figure 2: The general procedure of the ellipsoid algorithm.