AI2615 算法设计与分析 2020-2021 春季学期

Lecture 11 – Linear Programming

2021 年 5 月 7 日

Lecturer: 张驰豪 *Scribe:* 陶表帅

Suppose a factory can produce two kinds of products. The profit for producing one unit of the first kind of products is 1, and the profit for producing one unit of the second kind of products is 6. Suppose a factory has a limited amount of resources such that 1) the first kind of products cannot exceed 200 units, 2) the second kind of products cannot exceed 300 units, and 3) the overall amount of products cannot exceed 400 units. Given these, you are to decide how many units of the first kind of products and how many units of the second kind of products to produce respectively, so that the overall profit is maximized. This problem can be formulated as follows:

$$\begin{aligned} \max \quad & x_1 + 6x_2 \\ \text{subject to} \quad & x_1 \le 200 \\ & x_2 \le 300 \\ & x_1 + x_2 \le 400 \\ & x_1, x_2 \ge 0 \end{aligned} \tag{1}$$

This is a typical *linear program* (LP). In a linear program, we have a linear objective that needs to be maximized (or minimized), and we have a bundle of linear constraints.

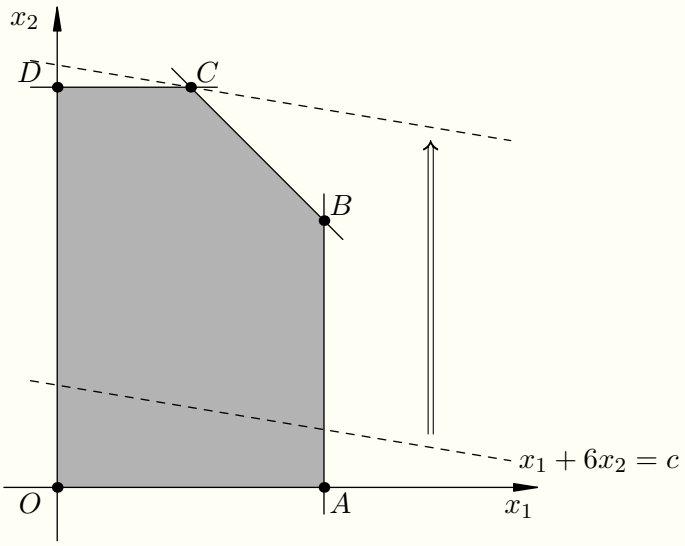

Figure 1: The feasible region for linear program (1).

The shadow area in Fig. 1 illustrates the feasible region for this linear program. The two axises correspond

to the constraints $x_1 \geq 0$ and $x_2 \geq 0$ respectively. Line AB corresponds to the constraint $x_1 \leq 200$, Line CD corresponds to the constraint $x_2 \leq 300$, and Line BC corresponds to the constraint $x_1 + x_2 \leq 400$. If we write $c = x_1 + 6x_2$, this corresponds to the dashed line with gradient $-\frac{1}{6}$. Our objective is to maximize c while making sure (x_1, x_2) is still in the feasible region. In Fig. 1, we would like to shift the line upward by as much as possible, while making sure this line still intersects the shadow area. This will shift the line to a position that intersects the point C, with coordinate $(100, 300)$. Therefore, the optimal cost to linear program (1) is 1900, with $x_1 = 100$ and $x_2 = 300$.

Let us see another linear program example with three variables.

$$\begin{aligned} \max \quad & x_1 + 6x_2 + 13x_3 \\ \text{subject to} \quad & x_1 \leq 200 \\ & x_2 \leq 300 \\ & x_1 + x_2 + x_3 \leq 400 \\ & x_2 + 3x_3 \leq 600 \\ & x_1, x_2, x_3 \geq 0 \end{aligned} \tag{2}$$

The feasible region for this linear program is the polytope shown in Fig. 2.

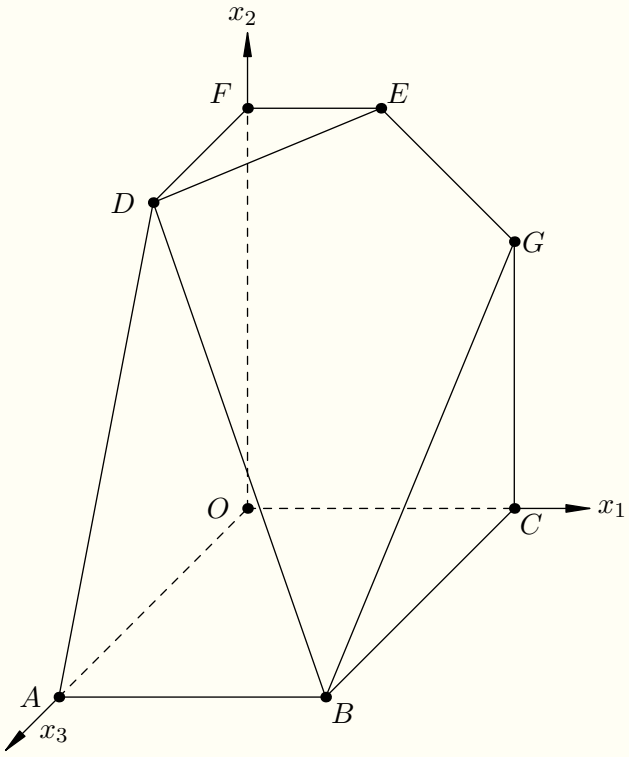

Figure 2: The feasible region for linear program (2).

If we write $c = x_1 + 6x_2 + 13x_3$ and try to maximize c, we will need to shift the plane $c = x_1 + 6x_2 + 13x_3$ upward by as much as possible, while making sure the plane still intersects the polytope.

In general, we can write a linear program by

$$\begin{aligned} \max \quad & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} \le \mathbf{b} \\ & \mathbf{x} \ge \mathbf{0} \end{aligned} \tag{3}$$

where $\mathbf{c}, \mathbf{x} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, and A is an $m \times n$ matrix.

1 Simplex Method

We have seen that the optimal solution for a linear program can be obtained by a line shifting (for two variables), a plane shifting (for three variables), or a hyper-plane shifting (for more than three variables). This method is based on geometric observations, and it is unclear how it can be described by an *algorithm*, especially when the number of variables becomes large. In this section, we will introduce the *simplex method*, which is an algorithm, or a class of algorithms, that is closely related to the above-mentioned geometric observations. This method is most commonly used in practice when solving a linear program.

Before introducing the simple method, we will first state some geometrically intuitive observations without proofs. First of all, the optimal solution $(x_1, \ldots, x_n)$ always corresponds to a *vertex* of the polytope describing the feasible region. In particular, it will not correspond to any interior point. For example, in Fig. 1, the optimal solution corresponds to vertex C; in Fig. 2, the optimal solution corresponds to vertex D (verify this!). Intuitively, if we try to shift the hyper-plane corresponding to the objective upward by as much as possible, we will reach a position where the hyper-plane just barely intersects the polytope by a single vertex. On the other hand, if an interior point is in the intersection between the hyper-plane and the polytope, we can still shift the hyper-plane upward, and this interior point cannot correspond to an optimal solution.

Secondly, if we consider the function that maps a point in the polytope to the objective value of the linear program, this function has only one local maximum, which is the global maximum (which is the cost of the optimal solution). This is because both the feasible region and the objective expression are *convex*.

The simple method makes use of these two observations. It starts from the origin, and it iteratively moves to an adjacent vertex with a higher cost, until it reaches a vertex with a cost higher than any of its neighbors (i.e., a local optimum). The two observations imply the correctness of this method: the optimal solution is always on a vertex, and the local optimum is just the global optimum. In the example in Fig. 1, the simplex method may go along the path $O \to A \to B \to C$; in the example in Fig. 2, the simplex method may go along the path $O \to C \to E \to B \to D$.

Before going further, let us first clarify what do we mean by saying two vertices are adjacent. This is clear in Fig. 1 and Fig. 2 with only two or three variables. When we have n variables, the polytope is in $\mathbb{R}^n$. A vertex is the intersection point of n hyper-planes corresponding to a set of n constraints. If two vertices

are adjacent, we mean that the line passing through these two vertices are the intersection of $n-1$ hyperplanes corresponding to a set of $n-1$ constraints. In other words, if two vertices are adjacent, there are $n-1$ common constraints between the n constraints defining one vertex and the n constraints defining the other.

One natural problem arises. When we are deciding the next vertex to move, there may be more than one adjacent vertex that has a higher cost. Which vertex are we going to? Since the number of vertices are finite, regardless of what we choose at each step, we will always end up to the vertex corresponding to optimal solution. Therefore, the choices here do not affect the correctness of the method. However, it affects the running time! Especially, we have exponentially many vertices for a polytope in $\mathbb{R}^n$: since n constraints define a vertex, we can have up to $\binom{m}{n}$ vertices if we have m constraints in total.

In fact, it is a central open problem if there is an implementation of the simplex method that always runs in polynomial time. We may have some natural choices when deciding the next vertex to move to in the simplex method. For example, we can choose an adjacent vertex with the highest cost, or, we can choose an adjacent vertex randomly. However, for each of those implementations of the simplex method that researchers have thought about, there are examples that makes the implementation runs in exponential time.

Fortunately, those examples are rare: in practice, the simplex method is very efficient. Theoretical computer scientists have been spending efforts on explaining this phenomenon. There is an entire field called *smoothed analysis*, introduced by Teng & Spielman, that deals with this. Most notably, Teng & Spielman showed that, if we add a random Gaussian noise to either A or $\mathbf{b}$ in (3), simplex method runs in polynomial time in the average case.

We also remark that, although the simplex method does not provably run in polynomial time (yet), there exist other methods for solving linear programs that provably run in polynomial time. We will discuss this more in the last section of this lecture. Therefore, whenever a problem can be formulated by a linear program, it is polynomial time solvable.

2 Vertex Cover

Definition 1. Given an undirected graph $G = (V, E)$, a subset of vertices $S \subseteq V$ is a *vertex cover* if $S \cap e \neq \emptyset$ for any edge $e = \{i, j\} \in E$.

In words, a vertex cover S "covers" every edge by including one or both endpoints of it.

Problem 2 ((Minimum) Vertex Cover). Given an undirected graph $G = (V, E)$, find a vertex cover with minimum number of vertices.

Exercise 3. Show that the vertex cover problem can be viewed as a special case of the set cover problem.

Exercise 4. Prove that $V \setminus S$ is an independent set of $G = (V, E)$ if S is a vertex cover.

For each vertex $v \in V$, we use $x_v \in \{0, 1\}$ to described if v is selected in the vertex cover. That is, $x_v = 0$ if $v \notin V$, and $x_v = 1$ if $v \in V$. The vertex cover problem can be formulated as follows.

$$\begin{aligned} \min \quad & \textstyle\sum_{v \in V} x_v \\ \text{subject to} \quad & \forall \{i, j\} \in E: \quad x_i + x_j \geq 1 \\ & \forall v \in V: \quad x_v \in \{0, 1\} \end{aligned} \tag{4}$$

This is an *integer program* (IP). The difference between an integer program and a linear program is that each variable in an integer program is restricted to take integer values, typically from $\{0, 1\}$. This difference is significant. We have see that a linear program can be solved in polynomial time, while solving an integer program is NP-hard. In fact, vertex cover is a well-known NP-hard problem.

In this section, we will see a common technique for designing an approximation algorithm: *linear relaxation*. The approximation algorithms we have seen in the previous lectures are all greedy algorithms. The success of a greedy-based approximation algorithm heavily relies on the structures of the problems. Linear relaxation, on the other hand, is a more generic technique.

Typically, to apply this technique, we first formulate the problem by an integer program (like (4) for vertex cover), and then relax the constraint $x_v \in \{0, 1\}$ to $0 \leq x_v \leq 1$ to obtain a linear program. The integer program (4) is then relaxed to the following linear program.

$$\begin{aligned} \min \quad & \textstyle\sum_{v \in V} x_v \\ \text{subject to} \quad & \forall \{i, j\} \in E: \quad x_i + x_j \geq 1 \\ & \forall v \in V: \quad 0 \leq x_v \leq 1 \end{aligned} \tag{5}$$

Notice that the linear program (5) can be easily convert to the standard form in (3). The objective can be rewrite to a maximization by $\max \sum_{v \in V} (-x_v)$ (the solution maximizing $\sum_{v \in V} (-x_v)$ is exactly the solution minimizing $\sum_{v \in V} x_v$. For the constraint $x_i + x_j \geq 1$ with the opposite direction of inequality, we can write it as $-x_i - x_j \leq -1$.

Let $OPT(IP)$ be the optimal cost for the integer program (4) and $OPT(LP)$ be the optimal cost for the linear program (5). Our objective is to find a solution to the integer program (4) with cost not far from $OPT(IP)$.

We first have the following simple observation.

Proposition 5. $OPT(IP) \geq OPT(LP)$.

Proof. Notice that the feasible region for (4) is a subset of the feasible region for (5). The optimal solution over a larger feasible region is clearly better. □

For the next step, we solve the linear program (5) and obtain the optimal solution $\{x_v^*\}$ with cost $OPT(LP)$, and we need to use this to find a solution for the vertex cover problem that gives a reasonably close

approximation to $OPT(IP)$. If x_v^* is an integer, it is either 0 or 1, which is also a valid value for x_v in the integer program (4). If x_v^* is a fractional value in the open interval $(0,1)$, a natural way to convert it to an integer value is by *rounding*. That is, we assign $x_v = 1$ if and only if $x_v^* \geq \frac{1}{2}$. Putting together, we have the following algorithm for vertex cover:

1. Formulate the problem to the integer program (4) and relax it to the linear program (5);
2. Solve the linear program (5) and obtain $\{x_v^*\}$;
3. Return $S = \{v \mid x_v^* \geq \frac{1}{2}\}$.

The following two lemmas show that this is a 2-approximation algorithm.

Lemma 6. *S returned by the algorithm is a vertex cover.*

Proof. For any edge $\{i, j\} \in E$, since $\{x_v^*\}$ is a valid solution to the linear program (5), we have $x_i^* + x_j^* \geq 1$. This implies either $x_i^* \geq \frac{1}{2}$ or $x_j^* \geq \frac{1}{2}$, or both are true. By our algorithm, S contains at least one of i and j. □

Lemma 7. $|S| \leq 2 \cdot OPT(IP)$.

Proof. Since we have $OPT(IP) \geq OPT(LP)$ by Proposition 5, it suffices to prove $|S| \leq 2 \cdot OPT(LP)$. We have

$$OPT(LP) = \sum_{v \in V} x_v^* = \sum_{v : x_v^* < \frac{1}{2}} x_v^* + \sum_{v : x_v^* \geq \frac{1}{2}} x_v^* \geq \sum_{v : x_v^* < \frac{1}{2}} 0 + \sum_{v : x_v^* \geq \frac{1}{2}} \frac{1}{2} = \frac{1}{2}|S|,$$

which implies $|S| \leq 2 \cdot OPT(LP)$. □

This approximation algorithm contains a typical idea behind general approximation algorithm design. For a minimization problem, when we want to prove that the value of the solution output by an algorithm is at most α times the optimum, it is usually hard to directly compare the value to the optimum, as finding the optimum is normally NP-hard. Instead, we find a *lower bound* to the optimum, and show that the value of the solution found by the algorithm is at most α times this lower bound. In our example here, the lower bound is $OPT(LT)$, and we show that the value of the solution we find is at most 2 times $OPT(LT)$.

3 LP Duality Theorem

We have seen that the optimal solution for the linear program (1) is $(x_1, x_2) = (100, 300)$, with cost 1900. We have used both an geometric argument and an argument based on simplex method to show that this is optimal. Can we prove it by some simple observations from the linear program itself? Let us denote the three constraints $x_1 \leq 200$, $x_2 \leq 300$ and $x_1 + x_2 \leq 400$ by (i), (ii) and (iii) respectively. If we add (i) to 6 times (ii), we obtain $x_1 + 6x_2 \leq 200 + 6 \times 300 = 2000$. This indicates that the optimal solution has cost at most 2000. Can we find a tighter upper bound for the cost of the optimal solution by a linear combination

of the constraints like what we have done just now? Yes, we can. In fact, if we multiple (ii) by 5 and add (iii), we obtain $x_1 + 6x_2 \le 5 \times 300 + 400 = 1900$. Since we have found a solution $(x_1, x_2) = (100, 300)$ with cost 1900, this proves that the solution is optimal.

Let us try this on the linear program (2). Suppose we multiple the first constraint $x_1 \le 200$ by y_1, the second constraint $x_2 \le 300$ by y_2, the third constraint $x_1 + x_2 + x_3 \le 400$ by y_3, and the last constraint $x_2 + 3x_3 \le 600$ by y_4. We ensure that $y_1, y_2, y_3, y_4 \ge 0$, and we obtain

$$(y_1 + y_3)x_1 + (y_2 + y_3 + y_4)x_2 + (y_3 + 3y_4)x_3 \le 200y_1 + 300y_2 + 400y_3 + 600y_4.$$

To find an upper bound to the objective $x_1 + 6x_2 + 13x_3$, we need to make sure that

$$x_1 + 6x_2 + 13x_3 \le (y_1 + y_3)x_1 + (y_2 + y_3 + y_4)x_2 + (y_3 + 3y_4)x_3$$

always holds. Since $x_1, x_2, x_3 \ge 0$, the only way to make this always true is to make sure, for each x_i ($i = 1, 2, 3$), the coefficient of x_i on the right-hand side of the inequality is at least as large as the coefficient of x_i on the left-hand side. Therefore, we have

$$\begin{aligned} y_1 + y_3 &\ge 1 \\ y_2 + y_3 + y_4 &\ge 6 \\ y_3 + 3y_4 &\ge 13 \end{aligned}$$

By ensuring these, $200y_1 + 300y_2 + 400y_3 + 600y_4$ becomes an upper bound to the objective $x_1 + 6x_2 + 13x_3$ of the linear program (2). Now we need to find an upper bound that is as tight as possible. In other words, we want to minimize $200y_1 + 300y_2 + 400y_3 + 600y_4$. Putting together, we have another linear program:

$$\begin{aligned} \min \quad & 200y_1 + 300y_2 + 400y_3 + 600y_4 \\ \text{subject to} \quad & y_1 + y_3 \le 1 \\ & y_2 + y_3 + y_4 \le 6 \\ & y_3 + 3y_4 \le 13 \\ & y_1, y_2, y_3, y_4 \ge 0 \end{aligned} \tag{6}$$

This linear program is called the *dual* of the linear program (2). Correspondingly, we call (2) the *primal*. Similarly, the dual of the linear program (1) is

$$\begin{aligned} \min \quad & 200y_1 + 300y_2 + 400y_3 \\ \text{subject to} \quad & y_1 + y_3 \ge 1 \\ & y_2 + y_3 \ge 6 \\ & y_1, y_2, y_3 \ge 0 \end{aligned} \tag{7}$$

For the general form linear program (3), its dual is

$$\begin{aligned} \min \quad & \mathbf{b}^\top \mathbf{y} \\ \text{subject to} \quad & \mathbf{y}^\top A \ge \mathbf{c}^\top \\ & \mathbf{y} \ge \mathbf{0} \end{aligned} \tag{8}$$

You may have learned the *Lagrange multiplier* for a convex program in another course. A linear program is a special case of a convex program. In fact, for a linear program, if we write the objective and the constraints by the Lagrange multiplier method, this is exactly the dual.
From our motivation that the dual program finds an upper bound for the objective of the primal program, the cost of a valid solution in the dual program is always weakly larger than the cost of a valid solution in the primal program. This is exactly *the weak duality theorem.*

Theorem 8 (Weak Duality Theorem). *If $\hat{\mathbf{x}}$ is a feasible solution for (3) and $\hat{\mathbf{y}}$ is a feasible solution for (8), we have $\mathbf{c}^\top \hat{\mathbf{x}} \le \mathbf{b}^\top \hat{\mathbf{y}}$.*

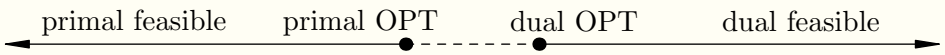

Figure 3: The weak duality theorem

In fact, if we solve the dual program (8), its optimal cost is a *tight* upper bound for the primal program (3). This is *the strong duality theorem.*

Theorem 9 (Strong Duality Theorem). *Let $\mathbf{x}^*$ be the optimal solution for (3) and $\mathbf{y}^*$ be the optimal solution for (8). We have $\mathbf{c}^\top \mathbf{x}^* = \mathbf{b}^\top \mathbf{y}^*$.*

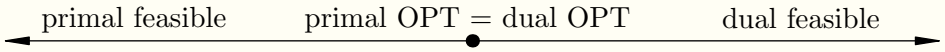

Figure 4: The strong duality theorem

3.1 Proof of Strong Duality Theorem

To prove the strong duality theorem, we will need *Farkas Lemma.*

Theorem 10 (Farkas Lemma). *Let $A \in \mathbb{R}^{m \times n}$ be a $m \times n$ real matrix and $\mathbf{b} \in \mathbb{R}^m$ be a m-dimensional real vector. Exactly one of the following is true:*

1. *there exists $\mathbf{x} \in \mathbb{R}^n$ with $\mathbf{x} \ge \mathbf{0}$ such that $A\mathbf{x} = \mathbf{b}$;*
2. *there exists $\mathbf{y} \in \mathbb{R}^m$ such that $A^\top \mathbf{y} \ge \mathbf{0}$ and $\mathbf{b}^\top \mathbf{y} < 0$.*

We will not give a formal proof for Farkas Lemma. Instead, we will give a geometric intuition.
Let us assume $m = n = 2$. Write $A = [\mathbf{c}_1 \quad \mathbf{c}_2]$, where $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^2$ are the two columns of the matrix A. The subset $\{A\mathbf{x} \mid \mathbf{x} \ge \mathbf{0}\} \subseteq \mathbb{R}^2$ is exactly the *cone* inscribed by the two vectors $\mathbf{c}_1, \mathbf{c}_2$, shown by the grey area in Fig. 5.
Suppose $\mathbf{b}$ is outside this grey area. We know that 1 of Farkas Lemma is false. To show that 2 is true, notice that we can find a plane (or a line in the case of $\mathbb{R}^2$) that "separates" the grey area and the vector $\mathbf{b}$,

such that the grey area and the vector $\mathbf{b}$ are on the opposite side of the plane. We choose a normal vector $\mathbf{y}$ of this plane such that $\mathbf{y}$ is on the same side of $\mathbf{c}_1, \mathbf{c}_2$, and is on the opposite side of $\mathbf{b}$. This vector then satisfies $A^\top \mathbf{y} \geq \mathbf{0}$ (since $\mathbf{c}_1^\top \mathbf{y} \geq 0$ and $\mathbf{c}_2^\top \mathbf{y} \geq 0$) and $\mathbf{b}^\top \mathbf{y} < 0$, so 2 in Farkas Lemma is true. See Fig. 5 for an illustration.

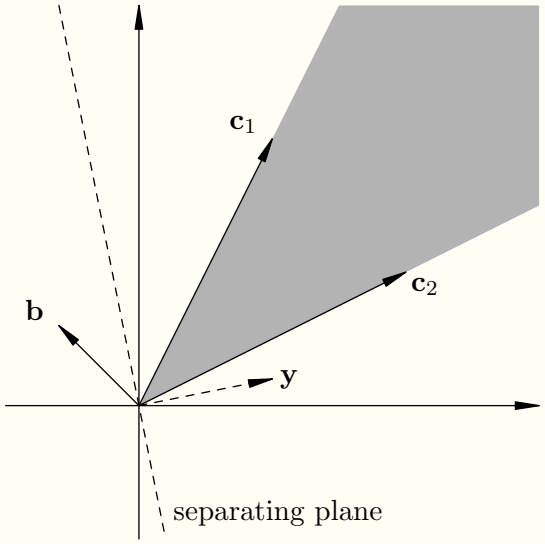

Figure 5: Geometric intuition behind Farkas Lemma.

Suppose $\mathbf{b}$ is inside the grey area. We know that 1 of Farkas Lemma is true. To see that 2 is false, 2 essentially says that we can find a separating plane that leaves $\mathbf{b}$ and the grey area on the opposite sides. This is clearly impossible if $\mathbf{b}$ is inside the grey area.
To prove the strong duality theorem, we will need the following corollary to Farkas Lemma.

Corollary 11 (A Corollary to Farkas Lemma). *Let $A \in \mathbb{R}^{m \times n}$ be a $m \times n$ real matrix and $\mathbf{b} \in \mathbb{R}^m$ be a m-dimensional real vector. Exactly one of the following is true:*

1. *there exists $\mathbf{x} \in \mathbb{R}^n$ with $\mathbf{x} \geq \mathbf{0}$ such that $A\mathbf{x} \geq \mathbf{b}$;*
2. *there exists $\mathbf{y} \in \mathbb{R}^n$ with $\mathbf{y} \leq \mathbf{0}$ such that $A^\top \mathbf{y} \geq \mathbf{0}$ and $\mathbf{b}^\top \mathbf{y} < 0$.*

Proof. Let A' be a $m \times (n+m)$ matrix defined by $A' = [A \quad -I]$, where I is the $m \times m$ identity matrix. We apply Farkas Lemma on A' and $\mathbf{b}$. Let P1 and P2 be statement 1 and 2 in Farkas Lemma for A' and $\mathbf{b}$ respectively. Let Q1 and Q2 be statement 1 and 2 in this corollary for A and $\mathbf{b}$ respectively. It suffices to show that P1 is equivalent to Q1 and P2 is equivalent to Q2. Then, exactly one of P1 and P2 being true implies exactly one of Q1 and Q2 being true.
P1 says that there exists $\mathbf{x}' \in \mathbb{R}^{n+m}$ with $\mathbf{x}' \geq \mathbf{0}$ such that $A'\mathbf{x}' = \mathbf{b}$. By writing $\mathbf{x}' = \begin{bmatrix} \mathbf{x} \\ \bar{\mathbf{x}} \end{bmatrix}$, this breaks down

to

$$[A \quad -I] \begin{bmatrix} \mathbf{x} \\ \bar{\mathbf{x}} \end{bmatrix} = \mathbf{b},$$

which is equivalent to $A\mathbf{x} = \mathbf{b} + \bar{\mathbf{x}}$, which is just equivalent to $A\mathbf{x} \geq \mathbf{b}$ since $\bar{\mathbf{x}}$ is a positive vector. We have proved P1 is equivalent to Q1.

P2 says that there exists $\mathbf{y} \in \mathbb{R}^m$ such that $A'^{\top}\mathbf{y} \geq \mathbf{0}$ and $\mathbf{b}^{\top}\mathbf{y} < 0$, which break down to

$$\begin{bmatrix} A^{\top} \\ -I \end{bmatrix} \mathbf{y} \geq \mathbf{0} \qquad \text{and} \qquad \mathbf{b}^{\top}\mathbf{y} < 0,$$

which further break down to $A^{\top}\mathbf{y} \geq \mathbf{0}$, $-\mathbf{y} \geq \mathbf{0}$, and $\mathbf{b}^{\top}\mathbf{y} < 0$, which is just Q2. □

There is also a geometric intuition behind this corollary. Suppose $m = n = 2$ and let $A = [\mathbf{c}_1 \quad \mathbf{c}_2]$ again. Like before, $A\mathbf{x}$ with $\mathbf{x} \geq \mathbf{0}$ describes the cone inscribed by vectors $\mathbf{c}_1$ and $\mathbf{c}_2$. On the other hand, all the vectors that are (coordinate-wise) weakly larger than $\mathbf{b}$ is the rectangular area that is to the upper-right of $\mathbf{b}$. In Fig. 6, the dark grey area is the cone inscribed by vectors $\mathbf{c}_1$ and $\mathbf{c}_2$, and the light grey area is the set of points that are coordinate-wise weakly larger than $\mathbf{b}$.

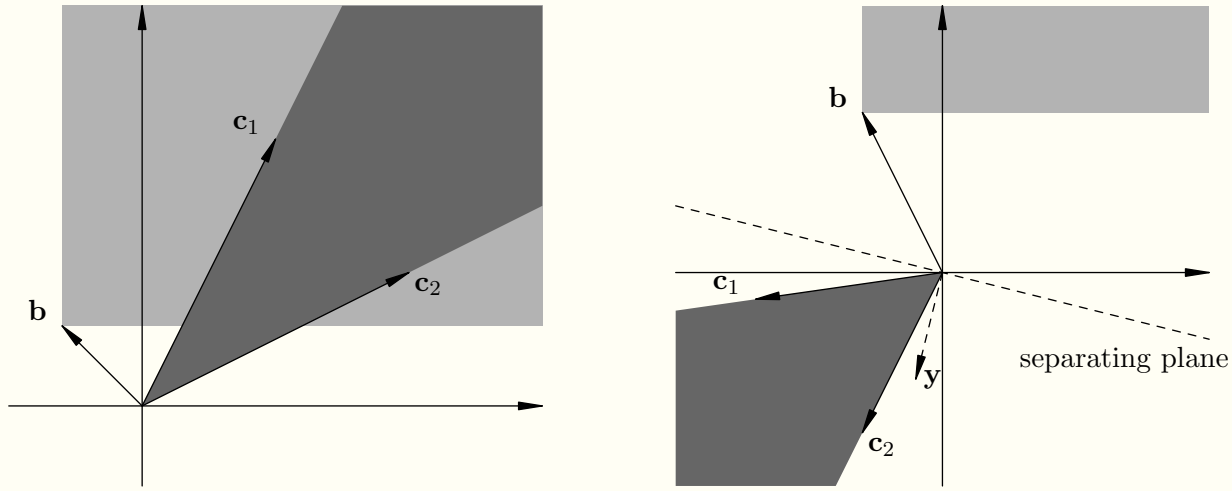

Figure 6: Geometric intuition behind the corollary of Farkas Lemma.

If 1 in the corollary is true, the two areas intersect. This means that we can have a point with coordinate $A\mathbf{x}$ that is coordinate-wise weakly larger than $\mathbf{b}$. See the figure on the left-hand side of Fig. 6.

If 1 in the corollary is false, the two areas do not intersect. This corresponds to the figure on the right-hand side of Fig. 6. We are able to find a separating plane that separates those two areas. We choose the normal vector $\mathbf{y}$ of this plane such that $\mathbf{y}$ is on the same side of $\mathbf{c}_1, \mathbf{c}_2$ and on the opposite side of $\mathbf{b}$. In particular, $\mathbf{y}$ must be in the third quadrant (i.e., $\mathbf{y} \leq \mathbf{0}$). For otherwise, the separating plane will pass through the first

quadrant, which will eventually intersect the light grey area. In general $\mathbb{R}^n$, if $\mathbf{y} \leq \mathbf{0}$ is not true, we can find a vector $\mathbf{z}$ with $\mathbf{z} < \mathbf{0}$ that is on the opposite side of $\mathbf{y}$. This means $-\mathbf{z}$ is on the same side with $\mathbf{y}$, which is on the same side with $\mathbf{c}_1, \mathbf{c}_2$. Since $-\mathbf{z} > \mathbf{0}$, we can rescale $-\mathbf{z}$ by a very large positive scalar α so that $-\alpha\mathbf{z} > \mathbf{b}$ (i.e., $-\alpha\mathbf{z}$ is in the light grey area). We have found a point in the light grey area that is on the same side of $\mathbf{c}_1$ and $\mathbf{c}_2$, so the plane fails to separate the two areas. Thus, we must have $\mathbf{y} \leq \mathbf{0}$.
Now we are ready to prove the strong duality theorem.

Proof of Theorem 9. The weak duality theorem indicates $\mathbf{c}^\top \mathbf{x} \leq \mathbf{b}^\top \mathbf{y}^*$ for any $\mathbf{x}$ that is feasible for (3). Suppose the strong duality does not hold. We must have $\mathbf{c}^\top \mathbf{x} < \mathbf{b}^\top \mathbf{y}^*$ for any $\mathbf{x}$ that is feasible for (3). In other words, there does not exist $\mathbf{x} \geq 0$ satisfying both $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{c}^\top \mathbf{x} \geq \mathbf{b}^\top \mathbf{y}^*$. To write this in the matrix form, we have that the set of constraints

$$\begin{bmatrix} -A \\ \mathbf{c}^\top \end{bmatrix} \mathbf{x} \geq \begin{bmatrix} -\mathbf{b} \\ \mathbf{b}^\top \mathbf{y}^* \end{bmatrix} \qquad \text{and} \qquad \mathbf{x} \geq \mathbf{0}$$

is infeasible.
By applying the corollary of Farkas Lemma on matrix $\begin{bmatrix} -A \\ \mathbf{c}^\top \end{bmatrix}$ and vector $\begin{bmatrix} -\mathbf{b} \\ \mathbf{b}^\top \mathbf{y}^* \end{bmatrix}$, since 1 in the corollary is false, 2 must be true, and there must exist $\mathbf{y} \in \mathbb{R}^m$ and $w \in \mathbb{R}$ such that

$$\begin{bmatrix} -A^\top & \mathbf{c} \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ w \end{bmatrix} \geq \mathbf{0}, \qquad \begin{bmatrix} -\mathbf{b}^\top & \mathbf{b}^\top \mathbf{y}^* \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ w \end{bmatrix} < 0 \qquad \text{and} \qquad \begin{bmatrix} \mathbf{y} \\ w \end{bmatrix} \leq \mathbf{0}.$$

After matrix multiplications, we have

$$\begin{cases} -A^\top \mathbf{y} + w\mathbf{c} \geq \mathbf{0} \\ -\mathbf{b}^\top \mathbf{y} + w\mathbf{b}^\top \mathbf{y}^* < 0 \\ \mathbf{y} \leq \mathbf{0} \\ w \leq 0 \end{cases}. \tag{9}$$

Now we discuss two cases: $w = 0$ and $w < 0$.
If $w = 0$, we have $A^\top(-\mathbf{y}) \geq \mathbf{0}$, $\mathbf{b}^\top(-\mathbf{y}) < 0$ and $-\mathbf{y} \geq 0$. Consider $\mathbf{z} = \mathbf{y}^* - \mathbf{y}$. We have $A^\top \mathbf{z} = A^\top \mathbf{y}^* + A^\top(-\mathbf{y}) \geq \mathbf{c}$. Therefore, $\mathbf{z}$ is a feasible solution to the dual program. In addition, we have $\mathbf{b}^\top \mathbf{z} = \mathbf{b}^\top \mathbf{y}^* + \mathbf{b}^\top(-\mathbf{y}) < \mathbf{b}^\top \mathbf{y}^*$. This contradicts to that $\mathbf{y}^*$ is dual optimal.
Now suppose $w < 0$. By dividing w on both side of the inequalities in (9), we have,

$$\begin{cases} -A^\top \left(\frac{\mathbf{y}}{w}\right) + \mathbf{c} \leq \mathbf{0} \\ -\mathbf{b}^\top \left(\frac{\mathbf{y}}{w}\right) + \mathbf{b}^\top \mathbf{y}^* > 0 \\ \frac{\mathbf{y}}{w} \geq 0 \end{cases}.$$

This implies $\hat{\mathbf{y}} := \frac{\mathbf{y}}{w}$ is a feasible solution to the dual program (8) such that $\mathbf{b}^\top \hat{\mathbf{y}} < \mathbf{b}^\top \mathbf{y}^*$, contradicting to that $\mathbf{y}^*$ is optimal. □

4 Zero-Sum Games and Minimax Theorem

In this section, we show that the strong duality theorem can be used to prove *minimax theorem*, a central theorem in game theory that is even considered as the starting point of game theory.

Consider the *rock-scissors-paper* game between two players, the *row player* and the *column player*. Each player can play one of the three moves: R (rock), S (scissors), or P (paper). Two players play simultaneously. The *payoff matrix* of this game is given in Table 1, where each entry corresponds to the gain for the row player, which is also the loss of the column player. For example, if the row player plays R and the column player plays S, the row player receives payoff 1, and the column player receives payoff -1 (the column player's loss is 1). In this game, the sum of all the players payoffs always equals to 0. Games satisfy this property are called *zero-sum games*.

		Column		
		R	S	P
Row	R	0	1	-1
	S	-1	0	1
	P	1	-1	0

Table 1: The payoff matrix for the rock-scissors-paper game.

A *strategy* of a player is a probability distribution over $\{R, S, P\}$. If the strategy specifies that one of the three moves is played with probability 1, this strategy is called a *pure strategy*. Otherwise, it is a *mixed strategy*. Fix one player's strategy, the *best response* for the other player is a strategy that maximizes this player's payoff. For example, if the column player plays the strategy $(1, 0, 0)$ (i.e., the column player plays R), the best response for the row player is $(0, 0, 1)$ (i.e., the row player plays P), which gives the row player payoff 1. If the column player plays $(\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$, the best response for the row player is again $(0, 0, 1)$. To see this, the expected payoff for the row player for playing R is

$$0 \times \frac{1}{2} + 1 \times \frac{1}{4} + (-1) \times \frac{1}{4} = 0,$$

the expected payoff for the row player for playing S is

$$(-1) \times \frac{1}{2} + 0 \times \frac{1}{4} + 1 \times \frac{1}{4} = -\frac{1}{4},$$

and the expected payoff for the row player for playing P is

$$1 \times \frac{1}{2} + (-1) \times \frac{1}{4} + 0 \times \frac{1}{4} = \frac{1}{4}.$$

Clearly, playing P gives the row player best payoff. In addition, since playing P gives the row player strictly higher payoff than playing R or playing S, playing any mixed strategy can only reduce the payoff of the

row player. In general, there may be more than one best response, and there is always a best response that only uses pure strategy. For example, if the column player plays $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, then any strategy of the row player is a best response, which always gives the row player payoff 0 (check this!).

In general, suppose each of the two players can play one of the n moves. Let $\mathbf{x} = (x_1, \ldots, x_n)$ be the strategy of the row player and $\mathbf{y} = (y_1, \ldots, y_n)$ be the strategy of the column player. Let $G \in \mathbb{R}^{n \times n}$ be the payoff matrix, where $G(i, j)$ is the payoff for the row player if the row player plays i and the column player plays j. The expected payoff for the row player is

$$\sum_{1 \le i, j \le n} G(i, j) x_i y_j.$$

Suppose the row player chooses a strategy first and assuming the column player always plays a best response. Given that the column player always plays a best response, the row player would like to choose a strategy that maximizes his own expected payoff. For the rock-scissors-paper game, the row player will play $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ and receive an expected payoff of 0. If the row player mixes the three moves unevenly, it is easy to see that he will receive a negative expected payoff when the column player plays a best response. On the other hand, if the column player chooses a strategy first and assuming the row player always plays a best response, the column player will also play $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ and receive an expected payoff of 0. In the rock-scissors-paper game, it does not matter which player chooses a strategy first, the expected payoff when two players act optimally is always the same.

You may believe that this property is true only for the rock-scissors-paper game because this game is highly symmetric. In fact, you may believe that the player choosing a strategy first has a disadvantage, as the other player can always play against this strategy in a best way. However, the *minimax theorem* says that this property is indeed always true for any zero-sum game.

To formally state this theorem, if the row player chooses a strategy first, he will choose $\mathbf{x}$ such that $\min_{\mathbf{y}} \sum_{1 \le i, j \le n} G(i, j) x_i y_j$ is maximized. In particular, for fixed $\mathbf{x}$, the column player's best response is a strategy $\mathbf{y}$ that minimizes the row player's payoff $\sum_{1 \le i, j \le n} G(i, j) x_i y_j$. Thus, the row player's payoff for playing strategy $\mathbf{x}$ is then $\min_{\mathbf{y}} \sum_{1 \le i, j \le n} G(i, j) x_i y_j$, and he wants to maximizes this value. Therefore, the row player's payoff for the optimal choice of strategy, given that the column player always plays a best response, is

$$\max_{\mathbf{x}} \min_{\mathbf{y}} \sum_{1 \le i, j \le n} G(i, j) x_i y_j.$$

Similarly, if the column player chooses a strategy first, his payoff for the optimal choice of strategy, given that the row player always plays a best response, is

$$\min_{\mathbf{y}} \max_{\mathbf{x}} \sum_{1 \le i, j \le n} G(i, j) x_i y_j.$$

The minimax theorem states that these two values are equal.

Theorem 12 (Minimax Theorem). *Given a zero-sum game with payoff matrix $G \in \mathbb{R}^{n\times n}$, we have*

$$\max_{\mathbf{x}} \min_{\mathbf{y}} \sum_{1\le i,j\le n} G(i,j)x_i y_j = \min_{\mathbf{y}} \max_{\mathbf{x}} \sum_{1\le i,j\le n} G(i,j)x_i y_j.$$

Let us see another example. Suppose each of the row player and the column player can play two moves, and the payoff matrix is given by

$$G = \begin{pmatrix} 3 & -1 \\ -2 & 1 \end{pmatrix}.$$

Suppose the row player chooses a strategy (x_1, x_2) first. The expected payoff for the column player for playing the first move is $3x_1 - 2x_2$, and it is $-x_1 + x_2$ for playing the second move. Fix the row player's strategy (x_1, x_2), the expected payoff for the column player for playing a best response is $\min\{3x_1 - 2x_2, -x_1 + x_2\}$. The expected payoff for the row player for an optimally chosen strategy is

$$\max_{(x_1,x_2)} \min\{3x_1 - 2x_2, -x_1 + 2x_2\}.$$

To solve x_1 and x_2, we can formulate it as a linear program:

$$\begin{aligned} \max \quad & z \\ \text{subject to} \quad & 3x_1 - 2x_2 \ge z \\ & -x_1 + x_2 \ge z \\ & x_1 + x_2 = 1 \\ & x_1, x_2 \ge 0 \end{aligned} \tag{10}$$

Suppose the column player chooses a strategy (y_1, y_2) first. By the same analysis, when the column player plays an optimal strategy and assuming the row player always plays a best response, the expected payoff for the column player is

$$\min_{(y_1,y_2)} \max\{3y_1 - y_2, -2y_1 + y_2\},$$

and (y_1, y_2) can be solved by the linear program

$$\begin{aligned} \min \quad & w \\ \text{subject to} \quad & 3y_1 - y_2 \le w \\ & -2y_1 + y_2 \le w \\ & y_1 + y_2 = 1 \\ & y_1, y_2 \ge 0 \end{aligned} \tag{11}$$

By writing the two linear programs to the standard form, we can see that (11) is the dual of (10). By the strong duality theorem, the two linear programs have the same optimum. This proves the minimax theorem by an example. Notice that the general case can be proved similarly by writing down the two linear programs and showing one is the dual of the other.

5 Polynomial Time Algorithms to Solve Linear Programs

We have mentioned that the simplex method can solve a linear program quickly in practice, but there is currently no guarantee that it can be done in polynomial time in the worst case. In this section, we will describe, only in high level ideas, two other algorithms for solving a linear program.

5.1 Ellipsoid Method

Firstly, notice that, if we are able to decide whether the feasible region of a linear program is a non-empty set, we can solve the linear program by a binary search on the objective value. The *ellipsoid method* is used to decide if the feasible region is non-empty.

It starts by finding a large ellipsoid in $\mathbb{R}^n$ that is guaranteed to enclose the feasible region (if non-empty). Then it finds a point inside this ellipsoid, and see if this point is inside the feasible region. If it is inside the feasible region, we are done. Otherwise, we can find a constraint which this point fails to satisfy. This constraint corresponds to a hyper-plane in $\mathbb{R}^n$ that intersect the ellipsoid, and this hyper-plane separates the point and the feasible region (see the dashed line in Fig. 7). We know that the feasible region is then in the half of the ellipsoid separated by the hyper-plane (the grey area in Fig. 7). In the next iteration, the ellipsoid method finds a smaller ellipsoid that enclose the grey area. The ellipsoid method iteratively "shrinks" the size of the ellipsoid and eventually locates the feasible region.

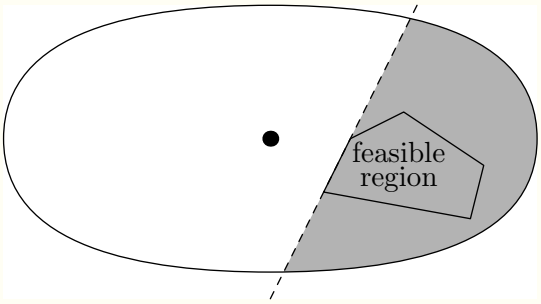

Figure 7: Ellipsoid method.

In fact, the ellipsoid method works even when there are exponentially many constraints, as long as there is a *separation oracle* that, given a point $\mathbf{x}$,

- output that $\mathbf{x}$ is in the feasible region if so, or
- find a constraint that $\mathbf{x}$ violates if $\mathbf{x}$ is not in the feasible region.

However, the ellipsoid method does not run in *strong polynomial time*. Its running time is polynomial in terms of n (the number of variables) and the numerical values in the linear program.

5.2 Interior Point Method

The *interior point method*, on the other hand, runs in strong polynomial time. However, it requires that the number of constraints m is bounded by a polynomial of the number of variables n.
Similar to the simplex method, the interior point method move a step in each iteration. Unlike the simplex method, the interior point method always moves *inside* the feasible region, instead of on the boundary, until it reaches the vertex representing the optimal solution. The closer the algorithm moves to the boundary, there is a larger "force" that repels it from the boundary.

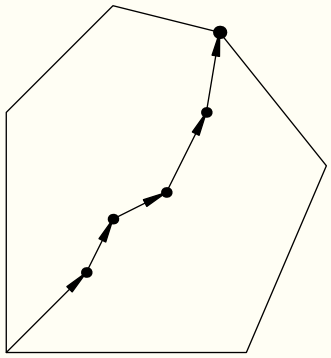

Figure 8: Interior point method.