Lecture 1: Linear Programming in Theoretical Computer Science

Rajat Mittal

IIT Kanpur

The primary aim of this lecture is to introduce you to linear programming. After a preliminary introduction to linear optimization and optimization in general, we will cover the basics of linear algebra needed for this course.

Exercise 1. What is linear algebra?

We would like to answer the following questions in the first part.

– What is linear programming?
– Why should we study linear programming?
– What content will be covered in this course?

Before we look at these questions, consider the following problem (this problem is taken from the lecture notes of Robert Sedgewick, Princeton University [2]). Suppose, we want to open a brewery which makes two kinds of beer, light and dark. The amount of ingredients needed for the two beers is listed below.

	$corn(kg)$	$hops(g)$	$barley(kg)$	$profit$
$dark(ltr)$	5	4	35	15
$light(ltr)$	15	4	20	20

We have already bought some material (which can't be returned); we have 480kg of corn, 160g of hop and 1190kg of barley. The question is: what quantity of each kind should we make to maximize profit?

If we make only light beer, then 32 liter can be made and corn will be the bottleneck. If only dark one is made, we get 34 liter with barley as the bottleneck. The profit in these cases is 640 and 510 respectively.

Trying out a few combinations, it is easy to see that we can possibly get more profit than 640. For instance, if we make 31 liter of light beer, allows us to make 3 more liters of dark beer, with better profit. This raises a very natural question, What is the best possible profit?

Let us formulate this problem mathematically. Suppose we make x liters of dark beer and y liters of light beer. Obviously, both x and y should be positive. We can also write constraints imposed by our inventory on x and y as equations.

In other words, x and y should satisfy the following constraints.

– $5x + 15y \leq 480$
– $4x + 4y \leq 160$
– $35x + 20y \leq 1190$
– $x, y \geq 0$

These are the only constraints we need to satisfy. So, our task reduces to finding x and y, which satisfy the constraints above, maximizing the profit. Notice that the profit can be written as $15x + 20y$.

Exercise 2. How can we solve this problem?

We are lucky, there are only two variables and hence we can plot our constraints on the X-Y plane. If we fix a profit, that equation can also be viewed as a line in this two dimensional space. Look at Fig. 1.

The set of x, y's which satisfy all the constraints are depicted by the black area. Convince yourself that any point is in this blackened area if and only if it satisfies all the constraints on x and y. So, we need to find the point/points in the blackened area which maximizes our profit.

Let us assume that the best profit is z, our profit line becomes $15x + 20y = z$. It can be viewed as a sliding line (for varying values of z). We need to find the maximum z, such that, our profit line intersects the blackened area with at least one point.

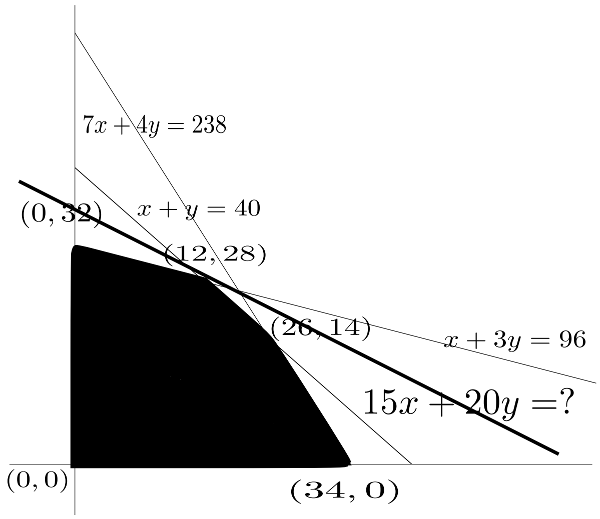

Fig. 1. Visual representation of beer making problem.

Exercise 3. Does the profit maximizing point need to be unique?

Now comes the main observation. Let us start with the line $15x + 20y = 0$ (zero profit). Moving the line upwards (increasing the profit), at some instance the complete line will fall outside the blackened area. Just before that, it should be hitting a vertex. In other words,
"There will be at least one vertex of black area which maximizes the profit."

Exercise 4. Can you finish the solution now?

Comparing the profit at all vertices, we see that $(12, 28)$ is the best possible solution for our beer problem. The total profit obtained is 740 Rs.

Congratulations, you have solved your first linear programming problem. It is not hard to see that such resource allocation problems appear a lot in our every-day life. Given a set of constraints over some variables, we need to find the setting of variables which maximizes/minimizes the profit/loss respectively.

The general class of problems dealing with maximization/minimization of quantities under constraints is studied in a subject called *mathematical optimization*.

1 Optimization

Optimization is a process of maximizing or minimizing a quantity under given constraints. Most of the problems in this world are optimization problems. You have to maximize (happiness/peace/money) or minimize (poverty, anxiety, grief, wars etc.). Unfortunately, we are not solving any of those problems.

On a smaller scale, there are many real world problems where we need to optimize quantities and constraints that are expressible as mathematical functions. To take some examples: optimizing time in the production cycle of an industry, optimizing tax in a tax-return, optimizing length in a tour are mathematical optimization problems we encounter in our daily life.

Formally, any problem of the form:

$$\begin{array}{ll} \min & f_0(x) \\ \text{s.t.} & f_i(x) \leq b_i \quad i = 1, 2, \cdots, m \end{array}$$

is called a mathematical optimization problem. Here,

- $x = (x_1, x_2, \cdots, x_n)$ is the set of variables, we need to find the values of these variables,
- $f_0(x)$ is the multivariate objective/optimization function and
- $f_i(x) \leq b_i$ (for any i) is called a constraint on variable x (in other words, constraint on the variables in x).

The task here is to find the max/min value of $f_0(x)$ such that x satisfies all the constraints. An x satisfying all the constraints is called a *feasible solution*. The set of all feasible x's (x's satisfying all the constraints) is called the *feasible region* (remember the black area in beer problem).

$$S = \{x : f_i(x) \leq b_i \quad \forall\, i \in [m]\}$$

A feasible solution x^* is called an *optimal* solution if it has the smallest objective value among all the feasible solutions. So, for any feasible z ($f_i(z) \leq b_i \quad \forall\, i \in [m]$), we know,

$$f_0(z) \geq f_0(x^*),$$

since it is a *minimization* problem.

Exercise 5. Can you give some examples of mathematical optimization problems?

It is quite evident from the previous discussion that general optimization problems seem to be really hard. Let us change the question: are there classes of optimization problems which can be solved easily and/or have specific properties? Generally, these different classes differ in the kind of constraints and objective functions that are allowed to be included in these problems. A natural question might be, what kind of classes should be studied? A class of problems is interesting if:

- Many real world problems can be modeled in that class.
- Problems in the class are *easily/efficiently* solved.
- Problems in the class have nice properties (e.g., Duality), which can give us more information about the structure of the problem (this will become clear later).

One of the prime example of such class is linear programming, the main focus for this course. It is the class of problems where both, objective function and constraints, are linear functions of the variables. Linear programming satisfies all the above properties and hence a natural candidate to be studied.

Using some standard manipulations (will be discussed later in the class), a linear program can always be written in the form

$$\begin{array}{lll} & \min \sum_j c_j x_j = c^T x & \\ \text{subject to} & a_i^T x = b_i & \forall i \in \{1, 2, \cdots, m\} \\ & x_j \geq 0 & \forall j \in \{1, 2, \cdots, n\} \end{array}$$

A more succinct representation can be obtained by arranging a_i's in a matrix A and b_i's in a vector b. The linear program becomes,

$$\begin{array}{rl} \min & c^T x \\ \text{subject to} & Ax = b \\ & x \geq 0 \end{array} \tag{1}$$

Note 1. Here, variable x can be thought of as a column vector with n entries.

So, a linear program is specified by three things: constraint matrix A, constraint vector b and objective vector c. The feasible region of this linear program is the solution set of $Ax = b$ intersected with positive orthant ($x \geq 0$).

This simple class of problems, linear programs, find a surprising large set of applications. To name a few,

– Finance: Portfolio management
– Management: Resource allocation
– Manufacturing: Production line optimization
– Telecommunications: Network design, routing
– Transportation: Traffic routing
– Computer science: Allocation of registers in compiler, and many more.

One way to realize the diversity of application is that the linear programming course is offered in departments as different as computer science, mathematics, optimization, industrial engineering, finance and social sciences.

Our objective will be to understand why linear programming can be solved efficiently, how to solve them and see some applications of them in the field of computer science. Even in computer science, linear programming finds its applications in not just algorithms but machine learning, algorithmic game theory and complexity theory. The focus of this course will be to view linear programs as a modelling tool for a set of diverse problems in computer science. We will start by covering the basics of linear programming techniques. In this process we will learn some linear algebra, definition and manipulation of linear programs and convexity theory.

After these basics, we will be ready for *duality theory* of linear programs, one of the most beautiful mathematical constructs in my humble opinion (given by John von Neumann). Duality theory provides us a deep insight in the world of linear programs and gives us information about the structure of quantities modeled by linear programs. In the process, we will look at some of the techniques to solve linear programs.

In the second half, our main focus will be to look at several applications of this class in theoretical computer science. Specifically, we will look at applications in algorithms, complexity theory and approximation theory. If time permits, we will look at one possible generalization of linear programs (called *semidefinite programs*).

Algorithms to solve linear programs:

You might already know that there are many known algorithms for solving linear programs; like simplex, ellipsoid and interior point method. Simplex method was one of the first methods to solve these programs. But almost all initial versions have examples which will take too long (exponential time) to solve. It is an open question if some version of simplex can run in polynomial time for all the instances. Since it is efficient in practice, it is used in many places.

The first polynomial time algorithm was Ellipsoid algorithm. It is not found to be very efficient in practice. Few years later, interior point method was developed and shown to be in polynomial time. Since it is efficient in practice and is provably fast, it is implemented in a lot of places.

Because of the abundance of algorithms to solve linear programs, researchers were really excited about this paradigm. There were many attempts to solve even NP hard problems (like traveling salesman problem) using linear programming. Notice that this will prove one of the most fundamental questions of complexity theory, P=NP. This is because we know that linear programs can be solved in polynomial time.

Recently there was a big result by Wolf et al., where they showed that most of these techniques are bound to fail. They showed that the traveling salesman polytope or its extension will require exponential number of constraints.

Convex optimization:

Convex optimization is a generalization of linear programming where the constraints and objective function are convex. It is interesting because most of the algorithms for linear programming can be generalized to convex optimization too. More importantly, many more problems can be expressed in this framework than linear programming. Many subclasses of convex optimization like semidefinite programming and least square problem are also widely used and have important applications in various fields. If time permits, we will cover basics of semidefinite programming.

2 Linear Programming

Linear programming is one of the well studied classes of optimization problem. We already discussed that a linear program is one which has linear objective and constraint functions. A linear constraint is a linear expression with equalities or inequalities.

Exercise 6. What is a linear expression?

A linear program looks like

$$\begin{aligned} \min \sum_j c_j x_j & \\ \text{subject to } a_i^T x \le b_i & \quad \forall i \in \{1, \cdots, m_1\} \\ a_i^T x \ge b_i & \ \forall i \in \{m_1+1, \cdots, m_2\} \\ a_i^T x = b_i & \ \ \forall i \in \{m_2+1, \cdots, m\} \end{aligned}$$

Here the vectors $c, a_1, \cdots, a_m \in \mathbb{R}^n$ and scalars $b_i \in \mathbb{R}$ are the problem parameters. Notice that $\sum_i c_i x_i$ can also be written as $c^T x$ in vector notation.

The task here is to find the minimum value of $c^T x$, s.t., x satisfies all the constraints. Like a general optimization problem, an x satisfying all the constraints is called a *feasible solution*. The set of all feasible x's, satisfying all the constraints, is called the *feasible region* (remember the black area in beer problem).

$$S = \{x : a_i^T x \le b_i \ \ \forall i \in [m_1], a_i^T x \ge b_i \ \ \forall i \in \{m_1+1, \cdots, m_2\}, a_i^T x = b_i \ \ \forall i \in \{m_2+1, \cdots, m\}\}$$

A feasible solution x^* will be called *optimal*, if it has the smallest objective value among all the feasible solutions. So for any feasible z, we know,

$$c^T z \ge c^T x^*.$$

Notice that the optimal solution need not be unique.

2.1 Examples

We have already seen one example, beer problem, in the introduction. Let us see a very similar problem and generalize it.

Suppose there is a manufacturing company which makes two kinds of laptop, Apple and Dell. Every Apple gives a profit of 10 Rs. and every Dell 5 Rs. It is clear that to maximize the profit the company should make as many Apple computers as possible (assuming they can sell everything they build).

Though, life is not so simple, every Apple computer takes 20 people to build, on the contrary Dell just takes 13. Similarly, an Apple needs 4 chips, but Dell needs only 1. At any particular day, the company has at most 95 people and 28 chips for their disposal. How many Apple's and Dell's should the company make? This problem is an instance of *resource allocation problem*.

From the mathematical point of view, the problem is quite clear,

$$\begin{aligned} \max \quad & 10x_1 + 5x_2 \\ \text{s.t.} \quad & 20x_1 + 13x_2 \le 95 \\ & 4x_1 + x_2 \le 28 \\ & x_1, x_2 \ge 0. \end{aligned}$$

Here, x_1 is the number of Apple's and x_2 is the number of Dell's. In a real scenario, we want these to be integers. Let's not worry about this constraint yet. Though, we will see that these kind of constraints, that variables should be integer, make certain problems really hard.

In any case, the above optimization approach can be generalized to the following resource allocation problem.

Suppose, a manufacturing unit wants to produce items $i = 1, \cdots, n$ using raw materials $j = 1, \cdots m$. The cost of raw material j is γ_j and the price of item i is ρ_i. There is only b_j amount of raw material j available. Suppose a single unit of item i requires a_{ij} amount of raw material j.

Perspective 1: The manager's job is,

$$\begin{array}{rc} & \max \quad \sum_i (\rho_i - \sum_j a_{ij}\gamma_j) x_j \\ \text{s.t.} & \forall\ j \quad \sum_i a_{ij} x_i \leq b_j \\ & \forall\ i \quad x_i \geq 0. \end{array}$$

Notice that $\rho_i - \sum_j a_{ij}\gamma_j$ can be thought of as the profit for item i, we call it c_i. Suppose c is the vector with co-ordinates c_i, x with co-ordinates x_i and $a_{(j)}$ is a vector with entries a_{ij}, then

$$\begin{array}{rc} & \max \quad c^T x \\ \text{s.t.} & \forall\ j \quad a_{(j)}^T x \leq b_j \\ & x \geq 0. \end{array}$$

Perspective 2: Let us look at the same resource allocation problem from another perspective. The total amount of profit can be also be thought as the *value of the inventory.* Suppose, the manager wants to assign some cost y_j to every raw material in the inventory, so that the cost of his inventory is minimized (for budget purposes). Though the catch is, he should be willing to sell the raw material at the same price to some other competitor manufacturing unit.

These constraint imply, his assigned cost should not be smaller than the market price, $y_j \geq \gamma_j$ (else the competitors can directly buy from him instead of market) and also

$$\forall\ i \quad \sum_j a_{ij} y_j \geq \rho_i$$

Otherwise, the competitor can buy the raw material from his unit and make the items cheaper than the market price. Hence, the problem becomes,

$$\begin{array}{rc} & \min \quad \sum_j b_j y_j \\ \text{s.t.} & \forall\ i \quad \sum_j a_{ij} y_j \geq \rho_i \\ & \forall\ j \quad y_j \geq \gamma_j. \end{array}$$

If we make a change of variable here $z_j = y_j - \gamma_j$, the life will be much simpler,

$$\begin{array}{rc} & \min \quad \sum_j b_j z_j \\ \text{s.t.} & \forall\ i \quad \sum_j a_{ij} z_j \geq c_i \\ & \forall\ j \quad z_j \geq 0. \end{array} \tag{2}$$

Notice that $\sum_j b_j \gamma_j$ is a constant and can be ignored in the optimization function. We again get a linear program. The two perspectives of the resource allocation problem seem very different. We will see later that they are two sides of the same coin! These two programs are called *dual* of each other. It is not just that there value is equal, lot more is known about the relationship between these programs. We will study *duality theory* in detail later.

Approximate degree linear program Let us take another example, where it is not straight forward to see the connection with linear programs. Mathematicians have always been interested in representing functions in terms of polynomials, it allows them to study and understand functions better.

Suppose there is a Boolean function $f : \{0,1\}^n \to \{0,1\}$. The function can be viewed as $f(x_1, x_2, \cdots, x_n)$, where x_i's are Boolean variables. We would like a representation of f in terms of a real polynomial. In other words, a polynomial $p(x_1, x_2, \cdots, x_n)$ represents/approximates f iff

$$|p(x_1, x_2, \cdots, x_n) - f(x_1, x_2, \cdots, x_n)| \leq 1/3 \;\; \forall x_1, x_2, \cdots, x_n \in \{0,1\}.$$

Notice a few things. First, even though f only takes input from $\{0,1\}^n$, p can take input from $\mathbb{R}^n$, but we are only interested in its values at $\{0,1\}^n$. Second, the constant on the right hand side is arbitrary $(1/3)$, we can put any constant smaller than $1/2$. Third, since we are only interested in Boolean values, $x_i^2 = x_i$; so we can assume that p has monomials where each individual degree of a variable is at most 1. That means, each monomials corresponds to a subset S of $[n]$, $\chi_S := \Pi_{i \in S} x_i$. Such polynomials are called multilinear polynomials.

Another thing, there can be multiple p's which represent f. We are interested in the one which has minimum degree. In other words, we are interested in

$$\widetilde{deg}(f) = \min_{p \text{ represents } f} \text{ degree of } p.$$

Approximate degree has been used in many places, e.g., quantum computing, learning theory and cryptography.

Can you write a linear program to find the approximate degree? Some thought will convince you (at least intuitively) that minimizing degree does not seem possible using linear programs (for one, it is an integer). Let us flip the question, how well can we approximate a function if we are restricted to degree d functions? Mathematically, is it possible to find

$$\epsilon_d = \min\{\epsilon : \exists \; p \text{ of degree } d \text{ such that } |p(x) - f(x)| \leq \epsilon \;\; \forall x \in \{0,1\}^n\}.$$

It turns out that this is an optimization program.

$$\begin{aligned} &\min \quad \epsilon \\ \text{s.t.} \;\; &|p(x) - f(x)| \leq \epsilon \;\; \forall x \in \{0,1\}^n \end{aligned}$$

Exercise 7. Do you think this is a linear program? What are the variables?

This is indeed a linear program with ϵ and coefficients of p as variables (notice that x is not a variable in this linear program). If we restrict p to have non-zero coefficients only when the monomial has degree less than d then this linear program gives us ϵ_d.

Note 2. We will not worry about the size of the linear program, it has exponential constraints in n. We are not planning to solve it. Just being able to write it as a linear program allows us to infer lot of properties of approximate degree. Though, that will be explained later.

It turns out that this linear program and its dual linear program is immensely useful, it is the only way known to lower bound approximate degree for non-symmetric functions. For a nice survey on approximate degree, check [1].

The dual of this linear program is,

$$
\begin{aligned}
\max \quad & \sum_{x\in\{0,1\}^n} f(x)(\phi_2(x)-\phi_1(x)) \\
\text{s.t.} \quad & \sum_{x\in\{0,1\}^n} \phi_1(x)+\phi_2(x) \leq 1 \\
\sum_{x\in\{0,1\}^n} (\phi_2(x)-\phi_1(x))\chi_S(x) = 0 \quad & \forall S \subseteq [n] : |S| \leq d \\
& \phi_1(x), \phi_2(x) \geq 0 \quad \forall x \in \{-1,1\}^n
\end{aligned}
$$

Remember that χ_S is the monomial corresponding to set S.

Exercise 8. Can you figure out the number of variables and constraints in this linear program?

2.2 Converting one LP into another

In the introduction, we defined a standard form of an LP and said that we can convert any LP into standard form. What does it mean to convert and LP into another?

Intuitively, it means that solving one LP gives us the solution for other LP too. What does it mean mathematically? Suppose we are given two LP's L_1 and L_2, when are they equivalent?

Two LP's (L_1 and L_2) are equivalent iff

- Any optimal solution of L_1 can be converted into a feasible solution of L_2 with *same* objective value.
- Any optimal solution of L_2 can be converted into a feasible solution of L_1 with *same* objective value.

Note 3. The solutions for two LP's having the *same* value can be defined in various ways, e.g., one could be a simple monotone function of another.

For an example, consider a sequence of sets $C_1, C_2, \cdots, C_m \subseteq \{0,1\}^n$. Define variables u_x, v_x for all $x \in \{0,1\}^n$. Consider the LP,

$$
\begin{aligned}
\max \quad & \textstyle\sum_x u_x + v_x \\
\text{s.t.} \ \ \forall\, i \in [m] \quad & \textstyle\sum_{x\in C_i} u_x - v_x \leq |C_i| \\
\forall\, x : \quad & u_x, v_x \in \mathbb{R}.
\end{aligned}
$$

Exercise 9. What is the optimal value of this LP?

Observe that by change of variables, $y_x = u_x + v_x$ and $z_x = u_x - v_x$, the LP converts to

$$
\begin{aligned}
\max \quad & \textstyle\sum_x y_x \\
\text{s.t.} \ \ \forall\, i \in [m] \quad & \textstyle\sum_{x\in C_i} z_x \leq |C_i| \\
\forall\, x : \quad & y_x, z_x \in \mathbb{R}.
\end{aligned}
$$

Now it is clear that value of z_x doesn't matter (we can set it to zero) and y_x can be raised as high as possible.

Exercise 10. Show that above two LP's are equivalent. What if in the first LP, we had constraint $u_x, v_x \geq 0$ for all x?

Let us take a non-trivial example of equivalence. Remember the dual of the approximate degree linear program.

$$\begin{aligned} \max \quad & \sum_{x \in \{0,1\}^n} f(x)(\phi_2(x) - \phi_1(x)) \\ \text{s.t.} \quad & \sum_{x \in \{0,1\}^n} \phi_1(x) + \phi_2(x) \leq 1 \\ \sum_{x \in \{0,1\}^n} (\phi_2(x) - \phi_1(x))\chi_S(x) &= 0 \;\; \forall S \subseteq [n] : |S| \leq d \\ \phi_1(x), \phi_2(x) &\geq 0 \;\; \forall x \in \{-1,1\}^n \end{aligned}$$

A notation will help our life a bit, given two functions $f, g : \{0,1\}^n \to \{0,1\}$, define $\langle f|g\rangle = \sum_{x \in \{0,1\}^n} f(x)g(x)$.

$$\begin{aligned} \max \quad & \langle f|\phi_2 - \phi_1\rangle \\ \text{s.t.} \quad & \sum_{x \in \{-1,1\}^n} \phi_1(x) + \phi_2(x) \leq 1 \\ \langle \phi_2 - \phi_1|\chi_S\rangle &= 0 \;\; \forall S \subseteq [n] : |S| \leq d \\ \phi_1(x), \phi_2(x) &\geq 0 \;\; \forall x \in \{-1,1\}^n \end{aligned}$$

A trick will simplify the linear program considerably. Suppose, we reduce $\phi_1(x)$ and $\phi_2(x)$ by the same quantity, keeping them positive. It remains a feasible solution (first and third constraint are still satisfied and second is not affected) and the objective value does not change.

For every x, we reduce both till at least one becomes zero. Then, we can replace them by a single variable $\phi(x)$, it is $-\phi_1(x)$ if $\phi_2(x)$ becomes zero first, and it is $\phi_2(x)$ if $\phi_1(x)$ becomes zero first. This gives the linear program,

$$\begin{aligned} \max \quad & \langle f|\phi\rangle \\ \text{s.t.} \quad & \sum_{x \in \{-1,1\}^n} |\phi(x)| \leq 1 \\ \langle \phi|\chi_S\rangle &= 0 \;\; \forall S \subseteq [n] : |S| \leq d \end{aligned}$$

Exercise 11. Show that we can renormalize ϕ, s.t., $\sum_x |\phi(x)| = 1$, without changing the objective value of the dual.

We get the final dual linear program,

$$\begin{aligned} \max \quad & \langle f|\phi\rangle \\ \text{s.t.} \quad & \|\phi\|_1 = 1 \\ \langle \phi|\chi_S\rangle &= 0 \;\; \forall S : |S| \leq d \end{aligned}$$

Converting to standard form The next set of moves show how to convert different kind of linear constraints into the standard form.

- inequality into equality: Use extra non-negative variables.
- Inequality in the opposite direction: A constraint like $d^T x \geq e$ can be converted to $(-d^T)x \leq (-e)$.

  Exercise 12. What if input variable is less than zero?

- No constraint on input variable: If x_i is unconstrained, then $x_i = y_i - z_i$, where $y_i, z_i \geq 0$.

Exercise 13. Show that the two LP's in this case would be equivalent in the sense described above.

- Strict inequalities: Not allowed in LP's. Instead we solve the approximate version with inequalities.
- We don't need to consider sup/inf and can only work with max/min. This can be justified using Fourier-Motzkin elimination.

Using these manipulations in the paragraph above, a linear program can be converted into an equivalent linear program of the form

$$\begin{aligned} \min \textstyle\sum_j c_j x_j = c^T x & \\ \text{subject to} \quad a_i^T x = b_i & \quad \forall i \in \{1, 2, \cdots, m\} \\ x_j \geq 0 & \quad \forall j \in \{1, 2, \cdots, n\} \end{aligned}$$

We will call this the *standard form* of the linear program. In the standard form, the vectors $c, a_1, \cdots, a_m \in \mathbb{R}^n$ and scalars $b_i \in \mathbb{R}$ are the problem parameters. In other words, given these parameters, you can write a complete linear program assuming it is in the standard form.

At this point, notice two things.

- All constraints and objective function are linear in variable x for the standard form.
- The beer problem almost looks like a program in standard form. Can you convert it into one?

Exercise 14. Convert all linear programs encountered till now into standard form.

A more succinct representation can be obtained by arranging a_i's in a matrix A and b_i's in a vector b. The linear program becomes,

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to } & Ax = b \\ & x \geq 0 \end{aligned}$$

Note 4. Here, variable x can be thought of as a column vector with n entries.

Given the program in this standard format, how should we solve and find the optimal solution? It seems like a difficult problem, let us simplify it. How about the case when there is no constraint $x \geq 0$.

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to } & Ax = b \end{aligned}$$

This should look familiar, actually an entire branch of mathematics is devoted to finding the solution of $Ax = b$ (and a few other basic questions). Our next step will be to learn a bit of linear algebra.

3 Linear algebra

Looking at the form of a linear program (Equation 1) the first obvious question is, do we even have a non-empty feasible region (setting of variables satisfying all constraints) in every case? The answer to this question depends on matrix A and vector b in the specification of the linear program.

First, we will go through the basics of linear algebra, it will not just help us answer the above question but also give us more information about the feasible set. This means, we will learn about vector spaces, linear independence, matrices and rank in this section.

The content below is mostly taken from Gilbert Strang's book, *Linear algebra and its applications* [3]. This brief introduction is supposed to be a reminder to the concepts of linear algebra, and is not a comprehensive introduction. For a detailed introduction to these concepts, please refer to Strang's book or any other elementary book on linear algebra.

Note 5. For the simplicity of notation, I will use capital letters for matrices and small letters for vectors.

3.1 System of linear equations

I believe that most of the areas in mathematics owe their existence to a problem. For linear algebra, one of the central problem is to solve equation $Ax = b$. Here, A is an $m \times n$ matrix, x represents variables (column matrix of dimension $n \times 1$) and b is a column vector with dimension $m \times 1$. This equation can also be viewed as m linear equations in n variables.

Exercise 15. What cases are easy to solve? What if A is a diagonal matrix?

Let us look at a very simple example,

$$\begin{aligned} 2x + y &= 4 \\ x + y &= 3 \end{aligned}$$

You can deduce that the solution is $x = 1, y = 2$. What about,

$$\begin{aligned} 2x + 2y &= 5 \\ x + y &= 3 \end{aligned}$$

You can again prove that these set of equations does not have a solution. So, before solving $Ax = b$, the first question we should ask is,

when does $Ax = b$ have a solution?

We will develop a theory for this question. You might wonder, why develop a theory when you can answer it just by inspection. The reason is, we want to answer this question when we have thousands of variable and equations. The algorithmic answer to these questions is used widely in many industries today.

The theory originates by looking at those two equations in a different manner. Instead of looking at the set of equations as two rows (linear equations), we will view them as column vectors and their combinations.

$$x \begin{bmatrix} 2 \\ 1 \end{bmatrix} + y \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

The system of linear equations question can be framed differently now. Does there exist a *linear* combination of two vectors, $\begin{bmatrix} 2 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$, which equals $\begin{bmatrix} 4 \\ 3 \end{bmatrix}$?

Vector spaces A vector space is a set of elements closed under addition and scalar multiplication (all linear combinations). In other words, V is a vector space iff

$$\forall\ x, y \in V,\ \alpha, \beta \in \mathbb{R} : \quad \alpha x + \beta y \in V.$$

In particular, it implies that for $x, y \in V$, $x + y$ and αx are members of the vector space.

Note 6. For reader with background in abstract algebra: we have defined the scalars (α, β) to be from real numbers. But the vector space can be defined over any field by taking the scalars from that field.

There is a more formal definition with axioms about the binary operations and identity element. But the definition above will provide enough intuition for us. The most common examples for a vector space are $\mathbb{R}^n$, $\mathbb{C}^n$, the space of all $m \times n$ matrices over a field and the space of all real-valued functions from a fixed domain. We will mostly be concerned with finite vector spaces over real number, $\mathbb{R}^n$, in this course.

A subspace is a subset of a vector space which is also a vector space and hence closed under addition and scalar multiplication. A *span* of a set of vectors S is the set of all possible linear combinations of vectors in S. It forms a subspace and is denoted by $Span(S)$.

Exercise 16. Give some examples of subspace of $\mathbb{R}^n$. Prove that a span is a subspace.

Note 7. We will be interested in vector space $\mathbb{R}^n$, but the following concepts are valid for general vector spaces.

Linear independence To understand the structure of a vector space, we need to understand how can all the elements of a vector space be generated. In particular, is there a set of vectors whose span is the given vector space? Using the definition of the vector space, the concept of linear dependence/independence comes out.

Given a set of vectors $v_1, \cdots, v_n \in V$, they are *linearly dependent* if and only if vector 0 can be expressed as a linear combination of these vectors.

$$\alpha_1 v_1 + \cdots + \alpha_n v_n = 0, \ \exists i: \ \alpha_i \neq 0.$$

This implies that at least some vector in the set can be represented as the linear combination of other elements. On the other hand, the set is called *linearly independent* iff

$$\alpha_1 v_1 + \cdots + \alpha_n v_n = 0 \Rightarrow \forall i, \alpha_i = 0$$

Intuitively, if we need to find generators of a vector space, a linearly dependent set is redundant. But a linearly independent set might not be able to generate all the elements of a vector space through linear combinations. This motivates the definition of basis, which is, in essence, a maximal linearly independent set of a vector space.

Definition 1. *Basis: A subset S of a vector space V is called a basis iff S is linearly independent and any vector in V can be represented as a linear combination of elements in S.*

Other way to put the same definition would be, a basis is the minimal set of vectors such that their span is the vector space itself.

Since any element in V can be represented as a linear combination of elements of S. This implies that adding any $v \in V \setminus S$ in S will make it linearly dependent (hence a basis is maximal linearly independent set).

One of the basic theorems of linear algebra says that the cardinality of all the basis sets is always the same and it is called the *dimension* of the vector space. Also given a linearly independent set of V, it can be extended to form a complete basis of V (hint: keep adding linearly independent vectors till a basis is obtained).

There is no mention about the uniqueness of the basis. There can be lot of basis sets for a given vector space.

The span of $k < n$ elements of a basis B_1 of V (dimension n) need not be contained in the span of some k' (even $n-1$) elements of B_2. Consider the standard basis $B = \{e_1, \cdots, e_n\}$ and vector $x = (1, 1, \cdots, 1)^T$. Now x or the space spanned by x is not contained in span of any $n-1$ vectors from B.

Inner product space All the examples we discussed above are not just vector spaces but inner product spaces. That means they have an associated inner product. Again we won't go into the formal definition. Intuitively, inner product (dot product for $\mathbb{R}^n$) allows us to introduce the concept of angles, lengths and orthogonality between elements of vector space. We will use $x^T y$ to denote the inner product between x and y.

Definition 2. *Orthogonality: Two elements x, y of vector space V are called* orthogonal *iff $x^T y = 0$.*

Definition 3. *Length: The length of a vector $x \in V$ is defined to be $\|x\| = \sqrt{x^T x}$.*

Using orthogonality we can come up with a simpler representation of a vector space. This requires the definition of *orthonormal basis.*

Definition 4. *A basis B of vector space V is orthonormal iff,*

- *For any two elements $x, y \in B$, $x^T y = 0$,*
- *For all elements $x \in B$, $\|x\| = 1$.*

In this orthonormal basis, every vector can be represented as a usual column vector ($n \times 1$ matrix) with respect to this orthonormal basis. It will have co-ordinates corresponding to every basis vector and operation between vectors like summation, scalar multiplication and inner product will make sense as the usual operation on the column vectors.

Given any basis of a vector space, it can be converted into an orthonormal basis. Start with a vector of the basis and normalize it (make it length 1). Take another vector, subtract the components in the direction of already chosen vectors. Normalize the remaining vector and keep repeating this process. This process always results in an orthonormal basis and is known as *Gram-Schmidt Process*.

3.2 Matrices

The next object of study (familiar to all of you) is called a *matrix*. An $m \times n$ matrix M over reals can be thought of as a collection of mn real numbers arranged in m rows and n columns. The matrix is represented as:

$$M = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

The i,j-th entry (entry in the i-th row and j-th column) is represented by a_{ij}.

A matrix can also be thought of as a collection of n column vectors. For a matrix A, A^T denotes the transpose of the matrix, where $A_{ij} = A^T_{ji}$. Let us look at some of the simple matrices used a lot in computer science.

- Zero matrix: The matrix with all the entries 0. It acts trivially on every element and takes them to the 0 vector.
- Identity matrix: The matrix with 1's on the diagonal and 0 otherwise. It takes $v \in V$ to v itself.
- All 1's matrix (J): All the entries of this matrix are 1.

There is another interpretation of matrices. Using the standard definition of a matrix multiplied by a vector, Mv, the $m \times n$ matrix M can be thought of as a function from $\mathbb{R}^n$ to $\mathbb{R}^m$.

Exercise 17. What is the action of matrix J?

In fact, the function associated with a matrix has very nice *linearity* property,

$$M(v+u) = Mv + Mu \quad \text{and} \quad M(\alpha u) = \alpha M u \ \forall \alpha \in \mathbb{R}.$$

3.3 Extra reading: linear operators

Given two vector spaces, V and W over $\mathbb{R}$, a *linear* operator $M : V \to W$ is defined as an operator satisfying the following properties.

- $M(x+y) = M(x) + M(y)$.
- $M(\alpha x) = \alpha M(x), \ \forall \alpha \in \mathbb{R}$.

For linear programming, the set of constraints can be thought of as linear operators on the variable vector. These conditions imply that the *zero* of the vector space V is mapped to the *zero* of the vector space W. Using the above two conditions,

$$M(\alpha_1 x_1 + \cdots + \alpha_k x_k) = \alpha_1 M(x_1) + \cdots + \alpha_k M(x_k)$$

Where $x_1, \cdots, x_k$ are elements of V and α_i's are in $\mathbb{R}$. Assuming the linearity of an operator, it is enough to specify the value of the linear operator on any basis of the vector space V. In other words, a linear operator is uniquely defined by the values it takes on any particular basis of V.

Let us define the addition of two linear operators as $(M+N)(u) = M(u) + N(u)$. Similarly, αM (scalar multiplication) is defined to be the operator $(\alpha M)(u) = \alpha M(u)$. The space of all linear operators from V to W (denoted $L(V, W)$) is a vector space in itself. The space of linear operators from V to V will be denoted by $L(V)$.

Exercise 18. Given the dimension of V and W, what is the dimension of the vector spaces $L(V, W)$?

Matrices as linear operators Given two vector spaces $V = \mathbb{R}^n, W = \mathbb{R}^m$ and a matrix M of dimension $m \times n$, the operation $x \in V \to Mx \in W$ is a linear operation. So, a matrix acts as a linear operator on the corresponding vector space.

To ask the converse, can any linear operator be specified by a matrix?

Let f be a linear operator from a vector space V (dimension n) to a vector space W (dimension m). Suppose $\{e_1, e_2, \cdots, e_n\}$ is a basis for the vector space V. Denote the images of this basis under f as $\{w_1 = f(e_1), w_2 = f(e_2) \cdots, w_n = f(e_n)\}$.

Exercise 19. What is the lower-bound/ upper-bound on the dimension of the vector space spanned by $\{w_1, w_2, \cdots, w_n\}$?

Define M_f to be the matrix with columns $w_1, w_2, \cdots, w_n$. Notice that M_f is a matrix of dimension $m \times n$. It is a simple exercise to verify that the action of the matrix M_f on a vector $v \in V$ is just $M_f v$. Here we assume that v is expressed in the chosen basis $\{e_1, e_2, \cdots, e_n\}$.

Exercise 20. Convince yourself that Mv is a linear combination of columns of M.

The easiest way to show that M_f acts similar to f is: notice that the matrix M_f and the operator f act exactly the same on the basis elements of V. Since both the operations are linear, they are exactly the same operation. This proves that any linear operation can be specified by a matrix.

The previous discussion does not depend upon the chosen basis. We can pick our favorite basis, and the linear operator can similarly be written in the new basis as another matrix (The columns of this matrix are images of the basis elements). In other words, given bases of V and W and a linear operator f, it has a unique matrix representation.

To compute the action of a linear operator, express $v \in V$ in the preferred basis and multiply it with the matrix representation. The output will be in the chosen basis of W. We will use the two terms, linear operator and matrix, interchangeably in future (bases will be clear from the context).

3.4 Kernel, image and rank

For a linear operator/matrix (viewed as an operator from V to W), the *kernel* is defined to be the set of vectors which map to 0.

$$ker(M) = \{x \in V : Mx = 0\}$$

Here 0 is a vector in space W.

Exercise 21. What is the kernel of the matrix J?

The *image* is the set of vectors which can be obtained through the action of the matrix on some element of the vector space V.

$$img(M) = \{x \in W : \exists y \in V,\ x = My\}$$

Exercise 22. Show that $img(M)$ and $ker(M)$ are subspaces.

Exercise 23. What is the image of J?

Notice that $ker(M)$ is a subset of V, but $img(M)$ is a subset of W. The dimension of $img(M)$ is known as the *rank* of M ($rank(M)$). The dimension of $ker(M)$ is known as the nullity of M ($nullity(M)$). For a matrix $M \in L(V, W)$, by the famous rank-nullity theorem,

$$rank(M) + nullity(M) = dim(V).$$

Here $dim(V)$ is the dimension of the vector space V.

Proof. Suppose $u_1, \cdots, u_k$ is the basis for $ker(M)$. We can extend it to the basis of V, $u_1, \cdots, u_k, v_{k+1}, \cdots, v_n$. We need to prove that the dimension of $img(M)$ is $n-k$. It can be proved by showing that the set $\{Mv_{k+1}, \cdots, Mv_n\}$ forms a basis of $img(M)$.

Exercise 24. Prove that any vector in the image of M can be expressed as linear combination of $Mv_{k+1}, \cdots, Mv_n$. Also any linear combination of $Mv_{k+1}, \cdots, Mv_n$ can't be zero vector.

□

Given a vector v and a matrix M, it is easy to see that the vector Mv is a linear combination of columns of M. To be more precise, $Mv = \sum_i M_i v_i$ where M_i is the ith column of M and v_i is the ith co-ordinate of v. This implies that any element in the image of M is a linear combination of its columns.

Exercise 25. Prove the rank of a matrix is equal to the dimension of the vector space spanned by its columns (column-space).

The dimension of the column space is sometimes referred to as the *column-rank*. We can similarly define the *row-rank*, the dimension of the space spanned by the rows of the matrix. Luckily, row-rank turns out to be equal to column-rank and we will call both of them as the rank of the matrix. This can be proved easily using *Gaussian elimination*. We will give a *visual* proof of the theorem.

Proof. Given an $m \times n$ matrix M, say $\{c_1, c_2, \cdots, c_k\}$ span the column space of M. Suppose, C be the $m \times k$ matrix with columns $\{c_1, c_2, \cdots, c_k\}$. Then, there exist an $k \times n$ matrix R, s.t., $CR = M$. If $\{d_1, d_2, \cdots, d_k\}$ are the columns of R, then the equation $CR = M$ can be viewed as,

$$\begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ c_1 & c_2 & \cdot & \cdot & c_k \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ d_1 & d_2 & \cdot & \cdot & d_n \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} = \begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ Cd_1 & Cd_2 & \cdot & \cdot & Cd_n \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Another way to view the same equation is,

$$C \begin{pmatrix} \cdots & r_1 & \cdots \\ \cdots & r_2 & \cdots \\ \cdots & \cdot & \cdots \\ \cdots & \cdot & \cdots \\ \cdots & r_k & \cdots \end{pmatrix} = \begin{pmatrix} \cdots & \sum_i C_{1i} r_i & \cdots \\ \cdots & \sum_i C_{2i} r_i & \cdots \\ \cdots & \cdot & \cdots \\ \cdots & \cdot & \cdots \\ \cdots & \sum_i C_{ki} r_i & \cdots \end{pmatrix}$$

This shows that the k columns of R span the row-space of M. Hence, row-rank is smaller than the column-rank.

Exercise 26. Show that column-rank is less than row-rank by a similar argument.

□

Note 8. The column-rank is equal to row-rank. It does not mean that the row-space is same as the column-space.

Using these characterizations of rank, it can be proved easily that $rank(A) = rank(A^T)$ and $rank(A + B) \leq rank(A) + rank(B)$.

3.5 Solutions for linear equations

We are now in a position to talk about the solution set of equation $Ax = b$. We will start with the easier case of $b = 0$. From the previous discussion, we are interested in the kernel of matrix A.

There are two cases, depending upon the rank of A.

- Non-singular: If A is full rank (the columns are linearly independent), implies the nullity of A is 0 by rank-nullity theorem. This means that the kernel has dimension 0. In other words, there is only a *trivial* solution $x = 0$ for $Ax = 0$.
- Singular: If A's rank is not full (the columns are linearly dependent), then dimension of kernel is more than 0. So, there are non-trivial solutions of $Ax = 0$ and they form a subspace of non-zero dimension.

To test whether the columns are linearly dependent or not (and solve linear equations), the established method in practice is called *Gaussian elimination.* I assume that you know what Gaussian elimination is, so we will describe it briefly.

The idea of Gaussian elimination is: it is easy to solve $Ax = 0$ if A is an upper-triangular matrix.

Exercise 27. Why is it easy to solve $Ax = 0$ when A is upper-triangular?

So, we reduce A to an upper triangular matrix, keeping its kernel the same. This is done using these two *elementary operations.*

- Exchange any two rows R_i and R_j.
- Replace a row R_j by $\alpha R_i + R_j$ for some $\alpha \in \mathbb{R}$.

Please convince yourself that these operations do not change the image or the kernel of the matrix A. These steps/operations can be used to convert A into an upper triangular matrix. Remember that a column of A represents a variable and a row represents an equation. The reduced matrix A looks like,

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & \cdots \\ 0 & a_{22} & a_{23} & a_{24} & \cdots \\ 0 & 0 & 0 & a_{34} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & a_{ij} & \cdots \end{pmatrix}$$

Notice that all diagonal entries need not be non-zero. You can achieve this by swapping the variables. The leading entries (column numbers) of every row correspond to leading variables. They can be used with elementary operations to make coefficients of leading variables 0 in every other equation. The reduced matrix will look like,

$$\begin{pmatrix} a_{11} & 0 & 0 & \cdots & 0 & \cdots \\ 0 & a_{22} & 0 & \cdots & 0 & \cdots \\ 0 & 0 & a_{33} & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & a_{kk} & \cdots \\ 0 & 0 & \cdots & 0 & \cdots & 0 \end{pmatrix}$$

Here, k is the rank of the matrix.

Exercise 28. Show that the number of leading variables is equal to rank of the matrix.

The remaining variables are called *free variables* and their number is equal to the nullity of the matrix A. A solution of $Ax = 0$ can be obtained by setting free variables as you wish; the value of leading variable is fixed by setting the free variables. This discussion shows that the kernel of A is a subspace with dimension equal to the number of free variables (why?). Since every variable is either leading or free, this proves the rank-nullity theorem in another way.

Given an $n \times n$ matrix, Gaussian elimination works in time $O(n^3)$. There are better algorithms known, but Gaussian elimination will be enough for our purposes.

3.6 Solution set of $Ax = b$

Do you remember our original motivation for learning linear algebra? The standard form of a linear program is,

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to } & Ax = b \\ & x \geq 0 \end{aligned}$$

We wanted to solve a simpler problem where the constraint $x \geq 0$ was absent. We are now in a position to describe the solution set of $Ax = b$. Next theorem relates the solution set of $Ax = b$ with solution set of $Ax = 0$ (we know this set from our discussion about linear algebra).

Theorem 1. *Let S_0 denote the solution set of $Ax = 0$ and the solution set of $Ax = b$ be not empty. Then, the solution set of $Ax = b$ can be represented as,*

$$x_0 + S_0 = \{x_0 + s : \ s \in S_0\}.$$

Here, x_0 is fixed to be a solution of $Ax = b$.

Proof. The theorem follows from these observations,

- if x_0 is a solution of $Ax = b$ and y is a solution of $Ax = 0$ then $x_0 + y$ is a solution of $Ax = b$,
- if x_0 is a solution of $Ax = b$ and x_1 is a solution of $Ax = b$ then $x_1 - x_0$ is a solution of $Ax = 0$,

$\square$

Given $Ax = b$, there are two cases, depending upon the rank of A.

- Non-singular: If A is full rank (the columns are linearly independent) then $Ax = 0$ has a unique trivial solution. Also, columns of A span the whole space. So, there is a unique solution for every b.
- Singular: If A's rank is not full (the columns are linearly dependent) then $Ax = 0$ has a non-trivial subspace as solution set. If b does not fall in the span of columns of A then there is no solution. Otherwise, pick any solution x_0 of $Ax = b$. The complete solution set is $x_0 + S_0$, where S_0 is the solution set of $Ax = 0$.

To find if b is in the span of A and what linear combination of columns of A will give b, run Gaussian elimination on the combined matrix $A' = [Ab]$. In case there is a row of reduced A' with 0's everywhere except the last entry, there is no solution.

The above discussion shows that either $Ax = b$ is in-feasible or its solution set is of the form $x_0 + S_0$, where S_0 is a subspace. The subspace S_0 can easily be determined using Gaussian elimination.

Given this solution set, it is easy to solve the *simpler* linear programming problem.

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to } & Ax = b \end{aligned}$$

Exercise 29. What happens if there is no solution of unique solution?

For the remaining case, suppose $v_1, v_2, \cdots, v_k$ is a basis of S_0. The feasible set is $\{x_0 + \sum_{i=1}^k \alpha_i v_i : \alpha_i \in \mathbb{R} \ \ \forall i\}$. This means we can get rid of constraints by substituting this form in the objective function. This will give us a linear optimization function in α_i's, $\sum_{i=1}^k c_i' \alpha_i + d$, without any constraints.

Exercise 30. Describe the solution for unconstrained optimization program

$$\min \quad \sum_{i=1}^{k} c_i' \alpha_i + d.$$

This shows that the removing constraint $x \geq 0$ will make the problem easy. We need to put this constraint back, though lessons learnt for this simpler problem will be useful in further discussions. Again, our strategy will be to study the new feasible set. Indeed, most (probably all) solvers depply rely on the structure of the feasible set for a linear program. Remember that we already know the solution set for $Ax = b$. Adding the $x \geq 0$ gives the feasible set a different structure. We will move to theory of convex sets next to study these feasible sets.

Exercise 31. What does the feasible set look like in 2 dimensions? How about 3 dimensions?

4 Assignment

Exercise 32. When A is singular, under what conditions will we have no solution for the system $Ax = b$?

Exercise 33. Read about Gaussian elimination.

Exercise 34. Prove that $rank(A) = rank(A^* A)$.

Hint: $rank(A) \geq rank(A^* A)$ is easy. For the other direction, reduce A to its reduced row echelon form.

Exercise 35. Show that $v^T A w = \sum_{ij} A_{ij} v_i w_j$, where A is a matrix and v, w are vectors.

Exercise 36. Prove that $Trace(AB) = Trace(BA)$.

Exercise 37. Show that $trace(A(v^T v)) = v^T A v$, where A is a matrix and v is a vector.

Exercise 38. What is the least square optimization problem? Read about it.

Exercise 39. Show that every linear program can be converted into this kind of standard form.

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0. \end{aligned}$$

Exercise 40. Consider a two player game with a matrix M (of dimension $n \times n$). The two players, call them row player and column player, have n strategies each. Row player gets an output M_{ij} when she plays strategy i and column player strategy j. We want to find probabilities $p_1, p_2, \cdots, p_n$ for row player which optimizes her output.

Show that this problem can be formulated as a linear program.

References

1. M. Bun and J. Thaler. Approximate degree in classical and quantum computing. `https://people.cs.georgetown.edu/jthaler/adegFnT.pdf`.
2. R. Sedgewick. Algorithms. `https://algs4.cs.princeton.edu/lectures/keynote/99LinearProgramming-2x2.pdf`.
3. Gilbert Strang. *Introduction to Linear Algebra.* Wellesley-Cambridge Press, 2009.