DRAFT

Formulation and Analysis of Linear Programs

Benjamin Van Roy and Kahn Mason

September 5, 2016

Contents

Chapter 1

Introduction

Optimization is the process of selecting the best among a set of alternatives. An optimization problem is characterized by a set of feasible solutions and an objective function, which assigns a value to each feasible solution. A simple approach to optimization involves listing the feasible solutions, applying the objective function to each, and choosing one that attains the optimum, which could be the maximum or minimum, depending on what we are looking for. However, this approach does not work for most interesting problems. The reason is that there are usually too many feasible solutions, as we illustrate with the following example.

Example 1.0.1. (Task Assignment) *Suppose we are managing a group of* 20 *employees and have* 20 *tasks that we would like to complete. Each employee is qualified to carry out only a subset of the tasks. Furthermore, each employee can only be assigned one task. We need to decide on which task to assign to each employee. The set of feasible solutions here is the set of* 20! *possible assignments of the* 20 *tasks to the* 20 *employees.*

Suppose our objective is to maximize the number of tasks assigned to qualified employees. One way of doing this is to enumerate the 20! *possible assignments, calculate the number of qualifying tasks in each case, and choose an assignment that maximizes this quantity. However,* 20! *is a huge number – its greater than a billion billion. Even if we used a computer that could assess* 1 *billion assignments per second, iterating through all of them would take more than* 75 *years!*

Fortunately, solving an optimization problem does not always require iterating through and assessing every feasible solution. This book is about linear programs – a class of optimization problems that can be solved very quickly by numerical algorithms. In a linear program, solutions are encoded in terms of real-valued decision variables. The objective function is linear

in the decision variables, and the feasible set is constrained by linear inequalities. Many important practical problems can be formulated as linear programs. For example, the task-assignment problem of Example 1.0.1 can be formulated as a linear program and solved in a small fraction of a second by a linear programming algorithm running on just about any modern personal computer.

This book represents a first course on linear programs, intended to serve engineers, economists, managers, and other professionals who think about design, resource allocation, strategy, logistics, or other decisions. The aim is to develop two intellectual assets. The first is an ability to formulate a wide variety of practical problems as linear programs. The second is a geometric intuition for linear programs. The latter helps in interpreting solutions to linear programs and often also leads to useful insights about problems formulated as linear programs. Many other books focus on linear programming algorithms. We discuss such algorithms in our final chapter, but they do not represent the focus of this book. The following sections offer previews of what is to come in subsequent chapters.

1.1 Linear Algebra

Linear algebra is about systems of linear equations. It forms a foundation for linear programming, which deals with both linear equations and inequalities. Chapter 2 presents a few concepts from linear algebra that are essential to developments in the remainder of the book. The chapter certainly does not provide comprehensive coverage of important topics in linear algebra; rather, the emphasis is on two geometric interpretations of systems of linear equations.

The first associates each linear equation in N unknowns with an $(N-1)$-dimensional plane consisting of points that satisfy the equation. The set of solutions to a system of M linear equations in N unknowns is then the intersection of M planes. A second perspective views solving a system of linear equations in terms of finding a linear combination of N vectors that generates another desired vector.

Though these geometric interpretations of linear systems of equations are simple and intuitive, they are remarkably useful for developing insights about real problems. In Chapter 2, we will illustrate the utility of these concepts through an example involving contingent claims analysis. In this context, the few ideas we discuss about linear algebra lead to an understanding of asset replication, market completeness, and state prices.

1.2 Linear Programs

In Chapter 3, we introduce linear programs. In a linear program, a solution is defined by N real numbers called *decision variables*. The set of feasible solutions is a subset of the N-dimensional space, characterized by linear inequalities. Each linear inequality is satisfied by points in the N-dimensional space that are on one side of an $(N-1)$-dimensional plane, forming a set called a *half space*. Given M inequalities, the set of feasible solutions is the intersection of M half-spaces. Such a set is called a *polytope*. A linear program involves optimization (maximization or minimization) of a linear function of the decision variables over the polytope of feasible solutions. Let us bring this to life with an example.

Example 1.2.1. (Petroleum Production) *Crude petroleum extracted from a well contains a complex mixture of component hydrocarbons, each with a different boiling point. A refinery separates these component hydrocarbons using a distillation column. The resulting components are then used to manufacture consumer products such as low, medium, and high octane gasoline, diesel fuel, aviation fuel, and heating oil.*

Suppose we are managing a company that manufactures N petroleum products and have to decide on the number of liters x_j, $j = 1, \ldots, N$ of each product to manufacture next month. Since the amount we produce must be positive, we constrain the decision variables by linear inequalities $x_j \geq 0$, $j = 1, \ldots, N$. Additional constraints arise from the fact that we have limited resources. We have M resources in the form of component hydrocarbons. Let b_i, $i = 1, \ldots, M$, denote the quantity in liters of the ith resource to be available to us next month. Our process for manufacturing the jth product consumes a_{ij} liters of the ith resource. Hence, production of quantities $x_1, \ldots, x_N$, requires $\sum_{j=1}^{N} a_{ij} x_j$ liters of resource i. Because we only have b_i liters of resource i available, our solution must satisfy

$$\sum_{j=1}^{N} a_{ij} x_j \leq b_i.$$

We define as our objective maximization of next month's profit. Given that the jth product garners c_j Dollars per liter, production of quantities $x_1, \ldots, x_N$ would generate $\sum_{j=1}^{N} c_j x_j$ Dollars in profit. Assembling constraints and the objective, we have the following linear program:

$$\begin{array}{lll} \text{maximize} & \sum_{j=1}^{N} c_j x_j & \\ \text{subject to} & \sum_{j=1}^{N} a_{ij} x_j \leq b_i, & i = 1, \ldots, M \\ & x_j \geq 0 & j = 1, \ldots, N. \end{array}$$

The above example illustrates the process of formulating a linear program. Linear inequalities characterize the polytope of feasible solutions and a linear function defines our objective. Resource constraints limit the objective.

1.3 Duality

Associated with every linear program is another called the *dual*. In order to distinguish a linear program from its dual, the former is called the *primal*. As an example of a *dual* linear program, we describe that associated with the petroleum production problem, for which the primal was formulated in Example 1.2.1.

Example 1.3.1. (Dual of Petroleum Production) *The dual linear program comes from a thought experiment about prices at which we would be willing to sell all our resources. Consider a set of resource prices $y_1, \ldots, y_M$, expressed in terms of dollars per liter. To sell a resource, we would clearly require that the price is nonnegative: $y_i \geq 0$. Furthermore, to sell a bundle of resources that could be used to manufacture a liter of product j we would require that the proceeds $\sum_{i=1}^{M} y_i a_{ij}$ at least equal the revenue c_j that we would receive for the finished product. Aggregating constraints, we have a characterization of the set of prices at which we would be willing to sell our entire stock of resources rather than manufacture products:*

$$\begin{array}{ll} \sum_{i=1}^{M} y_i a_{ij} \geq c_j, & j = 1, \ldots, N \\ y_i \geq 0 & i = 1, \ldots, M. \end{array}$$

The dual linear program finds our minimum net proceeds among such situations:

$$\begin{array}{lll} \text{minimize} & \sum_{i=1}^{M} y_i b_i & \\ \text{subject to} & \sum_{i=1}^{M} y_i a_{ij} \geq c_j, & j = 1, \ldots, N \\ & y_i \geq 0 & i = 1, \ldots, M. \end{array}$$

Duality theory studies the relationship between primal and dual linear programs. A central result – the duality theorem – states that optimal objective values from the primal and dual are equal, if both are finite. In our petroleum production problem, this means that, in the worst case among prices that would induce us to sell all our resources rather than manufacture, our proceeds would equal the revenue we could earn from manufacturing. It is obvious that the minimum cash amount for which we are willing to part with our resources should be equal to the maximum revenue we can generate through their use. The duality theorem makes a stronger statement that the

sale of aggregate resources for this cash amount can be induced by fixed per unit resource prices.

In Chapter 4, we develop duality theory. This theory extends geometric concepts of linear algebra, and like linear algebra, duality leads to useful insights about many problems. For example, duality is central to understanding situations where adversaries compete.

An example arises in two-player zero-sum games, which are situations where one agent's gain is the other's loss. Examples include two companies competing for a set of customers, or games of chess, football, or rock-paper-scissors. Such games can be described as follows. Each agent picks an action. If agent 1 picks action i and agent 2 picks action j then agent 1 receives r_{ij} units of benefit and agent 2 receives $-r_{ij}$ units of benefit. The game is said to be *zero-sum* because the benefits received by the two players always sum to zero, since one player receives r_{ij} and the other $-r_{ij}$.

Agent 1 wants to maximize r_{ij}, while agent 2 wants to maximize $-r_{ij}$ (or, equivalently, minimize r_{ij}). What makes games like this tricky is that agent 1 must take into account the likely actions of agent 2, who must in turn take into account the likely actions of agent 1. For example, consider a game of rock-paper-scissors, and suppose that agent 1 always chooses rock. Agent 2 would quickly learn this, and choose a strategy that will always beat agent 1, namely, pick paper. Agent 1 will then take this into account, and decide to always pick scissors, and so on.

The circularity that appears in our discussion of rock-paper-scissors is associated with the use of *deterministic strategies* – strategies that selects fixed actions. If an agent can learn his opponent's strategy, he can design a strategy that will beat it. One way around this problem is to consider randomized strategies. For example, agent 1 could randomly choose between rock, paper, and scissors each time. This would prevent agent 2 from predicting the action of agent 1, and therefore, from designing an effective counter-strategy.

With randomized strategies, the payoff will still be r_{ij} to agent 1, and $-r_{ij}$ to player 2, but now i and j are randomly (and independently) chosen. Agent 1 now tries to maximize the expected value of r_{ij}, while player 2 tries to maximize the expected value of $-r_{ij}$.

If we allow for randomized strategies, the problem of finding a strategy that maximizes an agent's expected payoff against the best possible counter-strategy can be formulated as a linear program. Interestingly, each agent's linear program is the dual of his opponent's. We will see that duality guarantees existence of an equilibrium. In other words, there exists a pair of strategies such that agent 1 can not improve his strategy after learning the strategy of agent 2, and vice-versa.

1.4 Network Flows

Network flow problems are a class of optimization problems that arise in many application domains, including the analysis of communication, transportation, and logistic networks. Let us provide an example.

Example 1.4.1. (Transportation Problem) *Suppose we are selling a single product and have inventories of $s_1, \ldots, s_M$ in stock at M different warehouse locations. We plan to ship our entire inventory to N customers, in quantities of $d_1, \ldots, d_N$. Shipping from location i to customer j costs c_{ij} Dollars. We need to decide on how to route shipments.*

Our decision variables x_{ij} represent the number of units to be shipped from location i to customer j. The set of feasible solutions is characterized by three sets of constraints. First, the amount shipped from each location to each customer is not negative: $x_{ij} \geq 0$. Second, all inventory at location i is shipped: $\sum_{j=1}^{N} x_{ij} = s_i$. Finally, the number of units sold to customer j are shipped to him: $\sum_{i=1}^{M} x_{ij} = d_j$. Note that the last two constraints imply that the quantity shipped is equal to that received: $\sum_{i=1}^{M} s_i = \sum_{j=1}^{N} d_j$.

With an objective of minimizing total shipping costs, we arrive at the following linear program:

$$\begin{array}{lll} \text{minimize} & \sum_{i=1}^{M} \sum_{j=1}^{n} c_{ij} x_{ij} & \\ \text{subject to} & \sum_{j=1}^{N} x_{ij} = s_i, & i = 1, \ldots, M \\ & \sum_{i=1}^{M} x_{ij} = d_j, & j = 1, \ldots, N \\ & x_{ij} \geq 0 & i = 1, \ldots, M; j = 1, \ldots, N. \end{array}$$

The set of feasible solutions to the transportation problem allows for shipment of fractional quantities. This seems problematic if units of the product are indivisible. We don't want to send a customer two halves of a car from two different warehouses! Surprisingly, we never run into this problem because shipping costs in the linear program are guaranteed to be minimized by integral shipments when inventories $s_1, \ldots, s_M$ and customer sales $d_1, \ldots, d_N$ are integers.

The situation generalizes to network flow problems of which the transportation problem is a special case – when certain problem parameters are integers, the associated linear program is optimized by an integral solution. This enables use of linear programming – a formulation involving continuous decision variables – to solve a variety of optimization problems with discrete decision variables. An example is the task-assignment problem of Example 1.0.1. Though the problem is inherently discrete – either we assign a task to an employee or we don't – it can be solved efficiently via linear programming. In Chapter 5, we discuss a variety of network flow problems and the integrality of optimal solutions.

1.5 Linear Programming Algorithms

As we have discussed, many important problems can be formulated as linear programs. Another piece of good news is that there are algorithms that can solve linear programs very quickly, even with thousands and sometimes millions of variables and constraints. The history of linear programming algorithms is a long winding road marked with fabulous ideas. Amazingly, over the course of their development over the past six decades, linear programming algorithms have become thousands of times faster and state-of-the-art computers have become thousands of times faster – together this makes a factor of millions. A linear program that would have required one year to solve when the first linear programming algorithms were deployed now takes under half a minute! In Chapter 6, we discuss some of this history and present some of the main algorithmic ideas in linear programming.

Chapter 2

Linear Algebra

Linear algebra is about linear systems of equations and their solutions. As a simple example, consider the following system of two linear equations with two unknowns:

$$\begin{aligned} 2x_1 - x_2 &= 1 \\ x_1 + x_2 &= 5. \end{aligned}$$

There is a unique solution, given by $x_1 = 2$ and $x_2 = 3$. One way to see this is by drawing a graph, as shown in Figure 2.1. One line in the figure represents the set of pairs (x_1, x_2) for which $2x_1 - x_2 = 1$, while the other represents the set of pairs for which $x_1 + x_2 = 5$. The two lines intersect only at $(2, 3)$, so this is the only point that satisfies both equations.

The above example illustrates one possible outcome with two equations and two unknowns. Two other possibilities are to have no solutions or an infinite number of solutions, as illustrated in Figure 2.2. The lines in Figure 2.2(a) are given by the equations

$$\begin{aligned} x_1 + x_2 &= 2 \\ x_1 + x_2 &= 5. \end{aligned}$$

Since they are parallel, they never intersect. The system of equations therefore does not have a solution. In Figure 2.2(b), a single line is given by both of the following equations

$$\begin{aligned} 2x_1 - x_2 &= 1 \\ -4x_1 + 2x_2 &= -2. \end{aligned}$$

Every point on this line solves the system of equations. Hence, there are an infinite number of solutions.

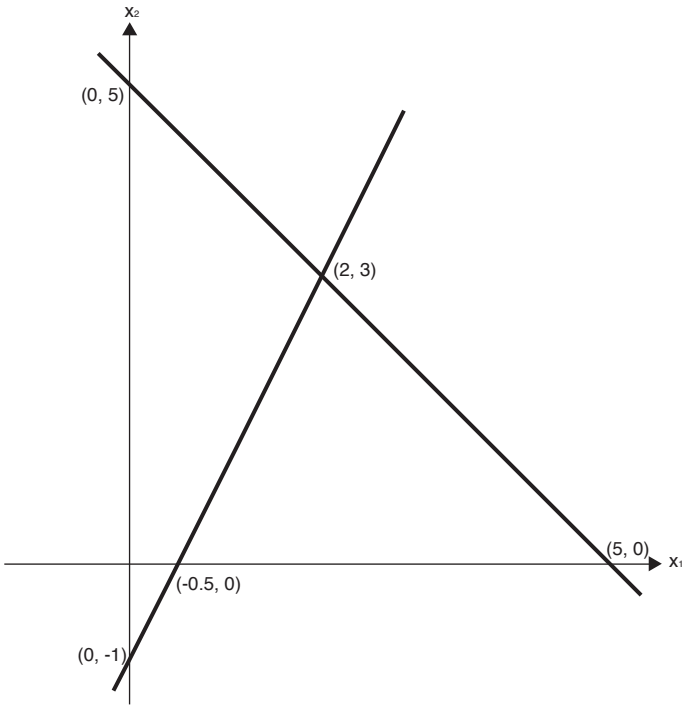
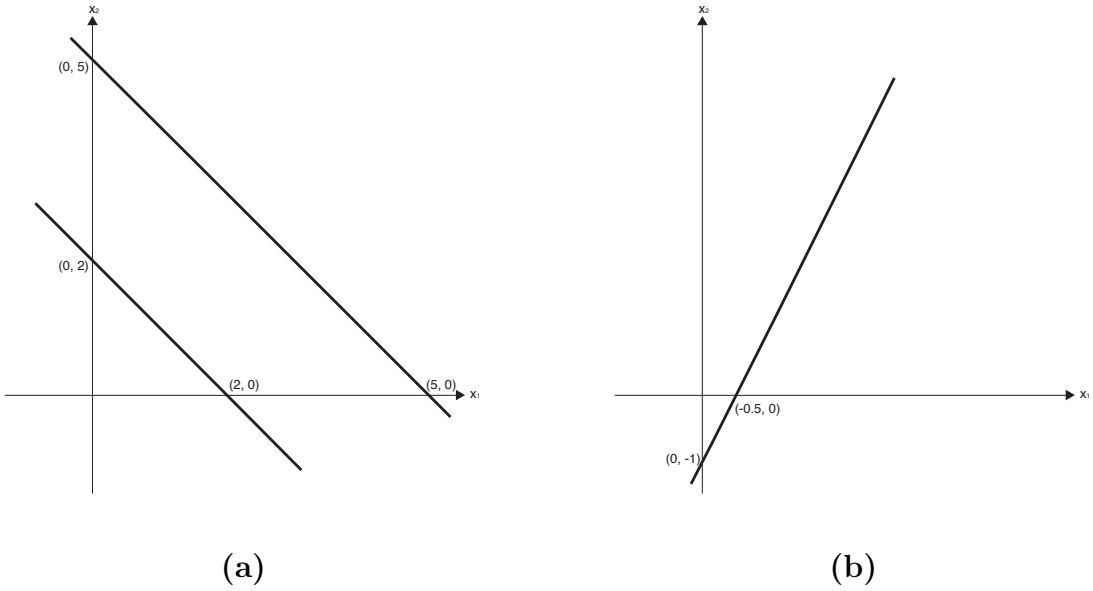


Figure 2.1: A system of two linear equations with two unknowns, with a unique solution.

There is no way to draw two straight lines on a plane so that they only intersect twice. In fact, there are only three possibilities for two lines on a plane: they never intersect, intersect once, or intersect an infinite number of times. Hence, a system of two equations with two unknowns can have either no solution, a unique solution, or an infinite number of solutions.

Linear algebra is about such systems of linear equations, but with many more equations and unknowns. Many practical problems benefit from insights offered by linear algebra. Later in this chapter, as an example of how linear algebra arises in real problems, we explore the analysis of contingent claims. Our primary motivation for studying linear algebra, however, is to develop a foundation for linear programming, which is the main topic of this book. Our coverage of linear algebra in this chapter is neither self-contained nor comprehensive. A couple of results are stated without any form of proof. The concepts presented are chosen based on their relevance to the rest of the book.



(a) **(b)**

Figure 2.2: (a) A case with no solution. (b) A case with an infinite number of solutions.

2.1 Matrices

We assume that the reader is familiar with matrices and their arithmetic operations. This section provides a brief review and also serves to define notation that we will be using in the rest of the book. We will denote the set of matrices of real numbers with M rows and N columns by $\Re^{M \times N}$. Given a matrix $A \in \Re^{M \times N}$, we will use the notation A_{ij} to refer to the entry in the ith row and jth column, so that

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1N} \\ A_{21} & A_{22} & \cdots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{M1} & A_{M2} & \cdots & A_{MN} \end{bmatrix}.$$

2.1.1 Vectors

A column vector is a matrix that has only one column, and a row vector is a matrix that has only one row. A matrix with one row and one column is a number. For shorthand, instead of using $\Re^{1 \times N}$ and $\Re^{N \times 1}$ to refer the N-dimensional row and column vectors, we will generally use $\Re^N$ for both. When using this shorthand notation and the distinction matters, it should

be clear from context whether we are referring to column or row vectors.

Given a vector $x \in \Re^M$, we denote the ith component by x_i. We will use A_{i*} to denote the row vector consisting of entries of the ith row. Similarly, A_{*j} will denote the column vector made up of the jth column of the matrix.

2.1.2 Addition

Two matrices can be added if they have the same number of rows and the same number of columns. Given $A, B \in \Re^{M \times N}$, entries of the matrix sum $A + B$ are given by the sums of entries $(A + B)_{ij} = A_{ij} + B_{ij}$. For example,

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}.$$

Multiplying a matrix by a scalar $\alpha \in \Re$ involves scaling each entry by α. In particular, given a scalar $\alpha \in \Re$ and a matrix $A \in \Re^{M \times N}$, we have $(\alpha A)_{ij} = (A\alpha)_{ij} = \alpha A_{ij}$. As an example,

$$2\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}.$$

Note that matrix addition is commutative (so, $A+B = B+A$) and associative ($(A + B) + C = A + (B + C) =: A + B + C$) just like normal addition. This means we can manipulate addition-only equations with matrices in the same ways we can manipulate normal addition-only equations. We denote by 0 any matrix for which all entries are 0.

2.1.3 Transposition

We denote the *transpose* of a matrix A by A^T. If A is an element of $\Re^{M \times N}$, A^T is in $\Re^{N \times M}$, and each entry is given by $(A^T)_{ij} = A_{ji}$. For example,

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}.$$

Clearly, transposition of a transpose gives the original matrix: $((A^T)^T)_{ij} = (A^T)_{ji} = A_{ij}$. Note that the transpose of a column vector is a row vector and vice-versa. More generally, the columns of a matrix become rows of its transpose and its rows become columns of the transpose. In other words, $(A^T)_{i*} = (A_{*i})^T$ and $(A^T)_{*j} = (A_{j*})^T$.

2.1.4 Multiplication

A row vector and a column vector can be multiplied if each has the same number of components. If $x, y \in \Re^N$, then the product $x^T y$ of the row vector x^T and column vector y is a scalar, given by $x^T y = \sum_{i=1}^{N} x_i y_i$. For example,

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} = 1 \times 4 + 2 \times 5 + 3 \times 6 = 32.$$

Note that $x^T x = \sum_{i=1}^{N} x_i^2$ is positive unless $x = 0$ in which case it is 0. Its square root $(x^T x)^{1/2}$ is called the *norm* of x. In three-dimensional space, the norm of a vector is just its Euclidean length. For example, the norm of $[4\ 5\ 6]^T$ is $\sqrt{77}$.

More generally, two matrices A and B can be multiplied if B has exactly as many rows as A has columns. The product is a matrix $C = AB$ that has as many rows as A and columns as B. If $A \in \Re^{M \times N}$ and $B \in \Re^{N \times K}$ then the product is a matrix $C \in \Re^{M \times K}$, whose (i, j)th entry is given by the product of A_{i*}, the ith row of A and B_{*j}, the jth column of B; in other words,

$$C_{ij} = A_{i*} B_{*j} = \sum_{k=1}^{N} A_{ik} B_{kj}.$$

For example,

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} = \begin{bmatrix} 32 \\ 77 \end{bmatrix}$$

Like scalar multiplication, matrix multiplication is associative (i.e., $(AB)C = A(BC)$) and distributive (i.e., $A(B + C) = AB + AC$ and $(A + B)C = AC + BC$), but unlike scalar multiplication, it is not commutative (i.e., AB is generally not equal to BA). The first two facts mean that much of our intuition about the multiplication of numbers is still valid when applied to matrices. The last fact means that not all of it is – the order of multiplication is important. For example,

$$\begin{bmatrix} 1 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ 4 \end{bmatrix} = 11, \quad \text{but} \quad \begin{bmatrix} 3 \\ 4 \end{bmatrix} \begin{bmatrix} 1 & 2 \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 4 & 8 \end{bmatrix}.$$

A matrix is *square* if the number of rows equals the number of columns. The *identity* matrix is a square matrix with diagonal entries equal to 1 and all other entries equal to 0. We denote this matrix by I (for any number of rows/columns). Note that for any matrix A, $IA = A$ and $AI = A$, given

identity matrices of appropriate dimension in each case. The identity matrix I plays a role in matrix multiplication analogous to that played by 1 in number multiplication.

A special case of matrix multiplication that arises frequently involves the second matrix being a column vector. Given a matrix $A \in \Re^{M \times N}$ and a vector $x \in \Re^N$, the product Ax is a vector in $\Re^M$. Each ith element of the product is the product of the vectors A_{i*} and x. Another useful way to view the product y is as a sum of columns of A, each multiplied by a component of x: $y = \sum_{j=1}^{N} x_j A_{*j}$.

2.1.5 Linear Systems of Equations

We will use matrix notation regularly to represent systems of linear equations. In particular, consider a system of M linear equations with N unknowns $x_1, \ldots, x_N$:

$$\begin{aligned} A_{11}x_1 + A_{12}x_2 + \cdots + A_{1N}x_N &= b_1 \\ A_{21}x_1 + A_{22}x_2 + \cdots + A_{2N}x_N &= b_2 \\ &\vdots \\ A_{M1}x_1 + A_{M2}x_2 + \cdots + A_{MN}x_N &= b_M. \end{aligned}$$

The ith equation can be rewritten as $A_{i*}x = b_i$. Furthermore, the entire system can be written in a compact form: $Ax = b$.

2.1.6 Partitioning of Matrices

It is sometimes convenient to view a matrix as being made up of smaller matrices. For example, suppose we have two matrix equations: $A^1x = b^1$ and $A^2x = b^2$, with $A^1 \in \Re^{M \times N}$, $A^2 \in \Re^{K \times N}$, $b^1 \in \Re^M$, and $b^2 \in \Re^K$. We can represent them as a single matrix equation if we define

$$A = \begin{bmatrix} A^1 \\ A^2 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} b^1 \\ b^2 \end{bmatrix}.$$

The two equations become one:

$$Ax = \begin{bmatrix} A^1 \\ A^2 \end{bmatrix} x = \begin{bmatrix} A^1x \\ A^2x \end{bmatrix} = \begin{bmatrix} b^1 \\ b^2 \end{bmatrix} = b. \tag{2.1}$$

Here, the first M rows of A come from A^1 and the last K rows come from A^2, so $A \in \Re^{(M+K) \times N}$. Similarly, the first M components of b come from b^1 and

the last K components come from b^2, and $b \in \Re^{M+K}$. The representation of a matrix in terms of smaller matrices is called a *partition*.

We can also partition a matrix by concatenating two matrices with the same number of rows. Suppose we had two linear equations: $Ax^1 = b^1$ and $Ax^2 = b^2$. Note that the vectors x^1 and x^2 can take on different values, while the two equations share a common matrix A. We can concatenate x^1 and x^2 because they have the same number of components (they must to be multiplied by A). Similarly, we can concatenate b^1 and b^2. If we write $X = [x^1\ x^2]$ and $B = [b^1\ b^2]$ then we again have $AX = B$. Note that now X and B are not vectors; they each have 2 columns. Writing out the matrices in terms of partitions, we have

$$A \begin{bmatrix} x^1 & x^2 \end{bmatrix} = \begin{bmatrix} b^1 & b^2 \end{bmatrix}. \tag{2.2}$$

Given the same equations, we could alternatively combine them to form a single equation

$$\begin{bmatrix} A & 0 \\ 0 & A \end{bmatrix} \begin{bmatrix} x^1 \\ x^2 \end{bmatrix} = \begin{bmatrix} Ax^1 + 0x^2 \\ 0x^1 + Ax^2 \end{bmatrix} = \begin{bmatrix} Ax^1 \\ Ax^2 \end{bmatrix} = \begin{bmatrix} b^1 \\ b^2 \end{bmatrix},$$

where the 0 matrix has as many rows and columns as A.

Note that, for any partition, the numbers of rows and columns of component matrices must be compatible. For example, in Equation (2.1), the matrices A^1 and A^2 had to have the same number of columns in order for the definition of A to make sense. Similarly, in Equation (2.2), x^1 and x^2 must have the same number of components, as do b^1 and b^2. More generally, we can partition a matrix by separating rows, columns, or both. All that is required is that each component matrix has the same number of columns as any other component matrix it adjoins vertically, and the same number of rows as any other component matrix it adjoins horizontally.

2.2 Vector Spaces

The study of vector spaces comprises a wealth of ideas. There are a number of complexities that arise when one deals with *infinite-dimensional* vector spaces, and we will not address them. Any vector space we refer to in this book is implicitly assumed to be *finite-dimensional*.

Given a collection of vectors $a^1, \ldots, a^N \in \Re^M$, the term *linear combination* refers to a sum of the form

$$\sum_{j=1}^{N} x_j a^j,$$

for some real-valued coefficients $x_1, \ldots, x_N$. A finite-dimensional vector space is the set S of linear combinations of a prescribed collection of vectors. In particular, a collection of vectors $a^1, \ldots, a^N \in \Re^M$ generates a vector space

$$S = \left\{ \sum_{j=1}^{N} x_j a^j \;\middle|\; x \in \Re^M \right\}.$$

This vector space is referred to as the *span* of the vectors $a^1, \ldots, a^N$ used to generate it. As a convention, we will take the span of an empty set of vectors to be the 0-vector; i.e., the vector with all components equal to 0.

If a vector x is in the span of $a^1, \ldots, a^N$, we say x is *linearly dependent* on $a^1, \ldots, a^N$. On the other hand, if x is not in the span of $a^1, \ldots, a^N$, we say x is linearly independent of $a^1, \ldots, a^N$. A collection of vectors, $a^1, \ldots, a^N$ is called *linearly independent* if none of the vectors in the collection is linearly dependent on the others. Note that this excludes the 0-vector from any set of linearly independent vectors. The following lemma provides an equivalent definition of linear independence.

Lemma 2.2.1. *A collection of vectors $a^1, \ldots, a^N$ is linearly independent if $\sum_{j=1}^{N} \alpha_j a^j = 0$ implies that $\alpha_1 = \alpha_2 = \cdots = \alpha_N = 0$.*

One example of a vector space is the space $\Re^M$ itself. This vector space is generated by a collection of M vectors $e^1, \ldots, e^M \in \Re^M$. Each e^i denotes the M-dimensional vector with all components equal to 0, except for the ith component, which is equal to 1. Clearly, any element of $\Re^M$ is in the span of $e^1, \ldots, e^M$. In particular, for any $y \in \Re^M$,

$$y = \sum_{i=1}^{M} y_i e^i,$$

so it is a linear combination of $e^1, \ldots, e^M$.

More interesting examples of vector spaces involve nontrivial subsets of $\Re^M$. Consider the case of $M = 2$. Figure 2.3 illustrates the vector space generated by the vector $a^1 = [1\ 2]^T$. Suppose we are given another vector a^2. If a^2 is a multiple of a^1 (i.e., $a^2 = \alpha a^1$ for number α), the vector space spanned by the two vectors is no different from that spanned by a^1 alone. If a^2 is not a multiple of a^1 then any vector in $y \in \Re^2$ can be written as a linear combination

$$y = x_1 a^1 + x_2 a^2,$$

and therefore, the two vectors span $\Re^2$. In this case, incorporating a third vector a^3 cannot make any difference to the span, since the first two vectors already span the entire space.

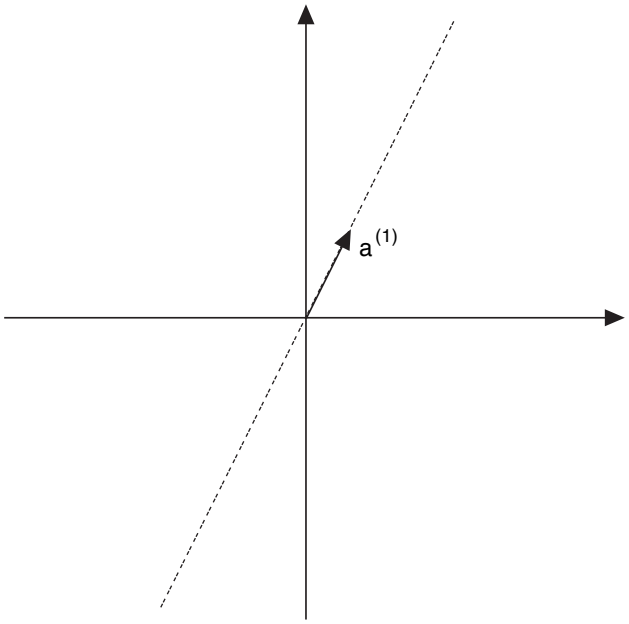


Figure 2.3: The vector space spanned by $a^1 = [1\ 2]^T$.

The range of possibilities increases when $M = 3$. As illustrated in Figure 2.4(a), a single vector spans a line. If a second vector is not a multiple of the first, the two span a plane, as shown in 2.4(b). If a third vector is a linear combination of the first two, it does not increase the span. Otherwise, the three vectors together span all of $\Re^3$.

Sometimes one vector space is a subset of another. When this is the case, the former is said to be a *subspace* of the latter. For example, the vector space of Figure 2.4(a) is a subspace of the one in Figure 2.4(b). Both of these vector spaces are subspaces of the vector space $\Re^3$. Any vector space is a subset and therefore a subspace of itself.

2.2.1 Bases and Dimension

For any given vector space S, there are many collections of vectors that span S. Of particular importance are those with the special property that they are also linearly independent collections. Such a collection is called a *basis*.

To find a basis for a space S, one can take any spanning set, and repeatedly remove vectors that are linear combinations of the remaining vectors.

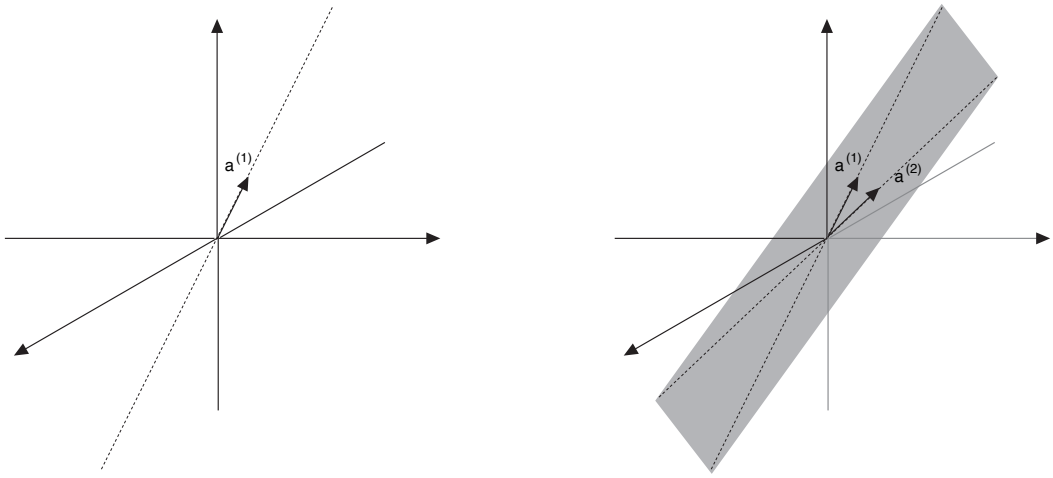


Figure 2.4: (a) The vector space spanned by a single vector $a^1 \in \Re^3$. (b) The vector space spanned by a^1 and $a^2 \in \Re^3$ where a^2 is not a multiple of a^1.

At each stage the set of vectors remaining will span S, and the process will conclude when the set of vectors remaining is linearly independent. That is, with a basis.

Starting with different spanning sets will result in different bases. They all however, have the following property, which is important enough to state as a theorem.

Theorem 2.2.1. *Any two bases of a vector space S have the same number of vectors.*

We prove this theorem after we establish the following helpful lemma:

Lemma 2.2.2. *If $A = \{a^1, a^2, \ldots, a^N\}$ is a basis for S, and if $b = \alpha_1 a^1 + \alpha_2 a^2 + \ldots + \alpha_N a^N$ with $\alpha_1 \neq 0$, then $\{b, a^2, \ldots, a^N\}$ is a basis for S.*

Proof of lemma: We need to show that $\{b, a^2, \ldots, a^N\}$ is linearly independent and spans S. Note that $a^1 = \frac{1}{\alpha_1} b - \frac{\alpha_2}{\alpha_1} a^2 - \ldots + \frac{\alpha_N a^N}{\alpha_1}$.

To show linear independence, note that the a^i's are linearly independent, so the only possibility for linear dependence is if b is a linear combination of $a^2, \ldots, a^N$. But a_1 is a linear combination of $b, a^2, \ldots, a^N$, so this would make a^1 a linear combination of $a^2, \ldots, a^N$, which contradicts the fact that

$a^1, \ldots, a^N$ are linearly independent. Thus b must be linearly independent of $a^2, \ldots, a^N$, and so the set is linearly independent.

To show that the set spans S, we just need to show that it spans a basis of S. It obviously spans $a^2, \ldots, a^n$, and because a^1 is a linear combination of $b, a^2, \ldots, a^n$ we know the set spans S. □

Proof of theorem: Suppose $A = \{a^1, a^2, \ldots, a^N\}$ and $B = \{b^1, b^2, \ldots, b^M\}$ are two bases for S with $N > M$. Because A is a basis, we know that $b^1 = \alpha_1 a^1 + \alpha_2 a^2 + \ldots + \alpha_N a^N$ for some $\alpha_1, \alpha_2, \ldots, \alpha_N$ where not all the α_i's are equal to 0. Assume without loss of generality that $\alpha_1 \neq 0$. Then, by Lemma 2.2.2, $A^1 = \{b^1, a^2, \ldots, a^N\}$ is a basis for S.

Now, because A^1 is a basis for S, we know that $b^2 = \beta_1 b^1 + \alpha_2 a^2 + \ldots + \alpha_N a^N$ for some $\beta_1, \alpha_2, \ldots, \alpha_N$ where we know that not all of $\alpha_2, \ldots, \alpha_N$ are equal to 0 (otherwise b^2 would be linearly dependent on b^1, and so B would not be a basis). Note the α_i's here are not the same as those for b^1 in terms of $a^1, a^2, \ldots, a^N$. Assume that $\alpha_2 \neq 0$. The lemma now says that $A^2 = \{b^1, b^2, a^3 \ldots, a^N\}$ is a basis for S.

We continue in this manner, substituting b^i into A^{i-1} and using the lemma to show that A^i is a basis for S, until we arrive at $A^M = \{b^1, \ldots, b^M, a^{M+1}, \ldots, a^N\}$ this cannot be a basis for S because a^{M+1} is a linear combination of $\{b^1, \ldots, b^M\}$. This contradiction means that N cannot be greater than M. Thus any two bases must have the same number of vectors. □

Let us motivate Theorem 2.2.1 with an example. Consider a plane in three dimensions that cuts through the origin. It is easy to see that such a plane is spanned by any two vectors that it contains, so long as they are linearly independent. This means that any additional vectors cannot be linearly independent. Since the first two were arbitrary linearly independent vectors, this indicates that any basis of the plane has two vectors.

Because the size of a basis depends only on S and not on the choice of basis, it is a fundamental property of S. It is called the dimension of S. This definition corresponds perfectly with the standard use of the word dimension. For example, $\dim(\Re^M) = M$ because $e^1, \ldots, e^M$ is a basis for $\Re^M$.

Theorem 2.2.1 states that any basis of a vector space S has $\dim(S)$ vectors. However, one might ask whether any collection of $\dim(S)$ linearly independent vectors in S spans the vector space. The answer is provided by the following theorem:

Theorem 2.2.2. *For any vector space $S \subseteq \Re^M$, each linearly independent collection of vectors $a^1, \ldots, a^{\dim(S)} \in S$ is a basis of S.*

Proof: Let $b^1, \ldots, b^K$ be a collection of vectors that span S. Certainly the set

$\{a^1, \ldots, a^{\dim(S)}, b^1, \ldots, b^K\}$ spans S. Consider repeatedly removing vectors b^i that are linearly dependent on remaining vectors in the set. This will leave us with a linearly independent collection of vectors comprised of $a^1, \ldots, a^{\dim(S)}$ plus any remaining b^i's. This set of vectors still spans S, and therefore, they form a basis of S. From Theorem 2.2.1 this means it has $\dim(S)$ vectors. It follows that the remaining collection of vectors cannot include any b^i's. Hence, $a^1, \ldots, a^{\dim(S)}$ is a basis for S. □

We began this section describing how a basis can be constructed by repeatedly removing vectors from a collection B that spans the vector space. Let us close describing how a basis can be constructed by repeatedly appending vectors to a collection. We start with a collection of vectors in a vector space S. If the collection does not span S, we append an element of $x \in S$ that is not a linear combination of vectors already in the collection. Each vector appended maintains the linear independence of the collection. By Theorem 2.2.2, the collection spans S once it includes $\dim(S)$ vectors. Hence, we end up with a basis.

Note that we could apply the procedure we just described even if S were not a vector space but rather an arbitrary subset of $\Re^M$. Since there can not be more than M linearly independent vectors in S, the procedure must terminate with a collection of no more than M vectors. The set S would be a subset of the span of these vectors. In the event that S is equal to the span it is a vector space, otherwise it is not. This observation leads to the following theorem:

Theorem 2.2.3. *A set $S \subseteq \Re^M$ is a vector space if and only if any linear combination of vectors in S is in S.*

Note that this theorem provides an alternative definition of a *vector space*.

2.2.2 Orthogonality

Two vectors $x, y \in \Re^M$ are said to be *orthogonal* (or *perpendicular*) if $x^T y = 0$. Figure 2.5 presents four vectors in $\Re^2$. The vectors $[1\ 0]^T$ and $[0\ 1]^T$ are orthogonal, as are $[1\ 1]^T$ and $[-1\ 1]^T$. On the other hand, $[1\ 0]^T$ and $[1\ 1]^T$ are not orthogonal.

Any collection of nonzero vectors $a^1, \ldots, a^N$ that are orthogonal to one another are linearly independent. To establish this, suppose that they are orthogonal, and that

$$\sum_{j=1}^{N} \alpha_j a^j = 0.$$

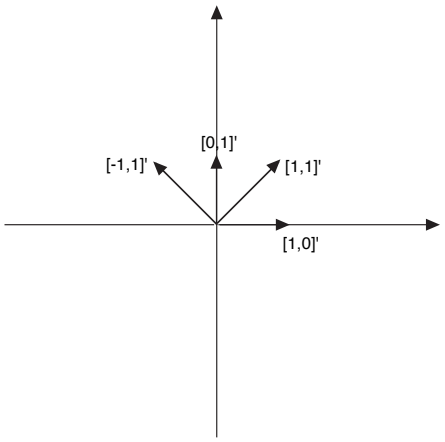


Figure 2.5: The vectors $[1\ 0]^T$ and $[0\ 1]^T$ are orthogonal, as are $[1\ 1]^T$ and $[-1\ 1]^T$. On the other hand, $[1\ 0]^T$ and $[1\ 1]^T$ are not orthogonal.

Multiplying both sides by $(a^k)^T$, we obtain

$$0 = (a^k)^T \sum_{j=1}^{N} \alpha_j a^j = \alpha_k (a^k)^T a^k.$$

Because $(a^k)^T a^k > 0$, it must be that $\alpha_k = 0$. This is true for all k, and so the a^i's are linearly independent. An *orthogonal basis* is one in which all vectors in the basis are orthogonal to one another.

2.2.3 Orthogonal Subspaces

Two subspaces S and T of $\Re^M$ are said to be orthogonal if $x^T y = 0$ for all $x \in S$ and $y \in T$. To establish that two subspaces S and T of $\Re^M$ are orthogonal, it suffices to show that a collection of vectors $u^1, \ldots, u^N$ that span S are orthogonal to a collection $v^1, \ldots, v^K$ that span T. To see why, consider arbitrary vectors $x \in S$ and $y \in T$. For some numbers $c_1, \ldots, c_N$ and $d_1, \ldots, d_K$, we have

$$x^T y = \left(\sum_{j=1}^{N} c_j u^j \right)^T \left(\sum_{k=1}^{K} d_k v^k \right) = \sum_{j=1}^{N} \sum_{k=1}^{K} c_j d_k (u^j)^T v^k = 0,$$

since each u^j is orthogonal to each v^k. We capture this in terms of a theorem.

Theorem 2.2.4. *Let S and T be the subspaces spanned by $u^1, \ldots, u^N \in \Re^M$ and $v^1, \ldots, v^K \in \Re^M$, respectively. S and T are orthogonal subspaces if and only if $(u^i)^T v^j = 0$ for all $i \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, K\}$.*

The notion of orthogonality offers an alternative way to characterize a subspace. Given a vector space S, we can define its *orthogonal complement*, which is the set $S^\perp \subseteq \Re^M$ of vectors that are orthogonal to all vectors in S. But is $S^\perp$ a vector space? We will now establish that it is.

From Theorem 2.2.3, it suffices to show that any linear combination of M vectors in $S^\perp$ is in $S^\perp$. Suppose $t^1, \ldots t^M$ are vectors in $S^\perp$ and that $x = \sum_{i=1}^M \alpha_i t^i$ for some numbers $\alpha_1, \ldots, \alpha_M$. Then, for any $s \in S$

$$s^T x = \sum \alpha_i s^T t^i = \sum \alpha_i \cdot 0 = 0.$$

This establishes the following theorem.

Theorem 2.2.5. *The orthogonal complement $S^\perp \subseteq \Re^M$ of a vector space $S \subseteq \Re^M$ is also a vector space.*

The following theorem establishes an interesting property of orthogonal subspaces.

Theorem 2.2.6. *The dimensions of a vector space $S \subseteq \Re^M$ and its orthogonal complement $S^\perp \subseteq \Re^M$ satisfy $\dim(S) + \dim(S^\perp) = M$.*

Proof: Suppose $s^1, \ldots, s^N$ form a basis for S, and $t^1, \ldots, t^L$ form a basis for $S^\perp$. It is certainly the case that $N + L \leq M$ because there can not be more than M linearly independent vectors in $\Re^M$.

We will assume that $N + L < M$, and derive a contradiction. If $N + L < M$, there must be a vector $x \in \Re^M$ that is not a linear combination of vectors in S and $S^\perp$. Let $\alpha_1^*, \ldots, \alpha_N^*$ be a set of numbers that minimizes the function

$$f(\alpha_1, \ldots, \alpha_N) = \left(x - \sum_{i=1}^N \alpha_i s^i\right)^T \left(x - \sum_{i=1}^N \alpha_i s^i\right).$$

Such numbers exist because the function is quadratic with a nonnegative range. Let $y = x - \sum_{i=1}^N \alpha_i^* s^i$. Note that $y \notin S^\perp$, because if $y \in S^\perp$ then x would be a linear combination of elements of S and $S^\perp$. It follows that there is a vector $u \in S$ such that $y^T u \neq 0$. Let $\beta = -\frac{u^T y}{u^T u}$. Then,

$$\begin{aligned}(y + \beta u)^T (y + \beta u) &= y^T y + 2\beta u^T y + \beta^2 u^T u \\ &= y^T y - 2\frac{u^T y}{u^T u} u^T y + \left(-\frac{u^T y}{u^T u}\right)^2 u^T u\end{aligned}$$

$$\begin{aligned} &= y^T y - 2\frac{(u^T y)^2}{u^T u} + \frac{(u^T y)^2}{u^T u} \\ &= y^T y - \frac{(u^T y)^2}{u^T u} \\ &< y^T y \end{aligned}$$

which contradicts that fact that $y^T y$ is the minimum of $f(\alpha_1, \ldots, \alpha_N)$. It follows that $N + L = M$. □

2.2.4 Vector Spaces Associated with Matrices

Given a matrix $A \in \Re^{M \times N}$, there are several subspaces of $\Re^M$ and $\Re^N$ worth studying. An understanding of these subspaces facilitates intuition about linear systems of equations. The first of these is the subspace of $\Re^M$ spanned by the columns of A. This subspace is called the *column space* of A, and we denote it by $\mathcal{C}(A)$. Note that, for any vector $x \in \Re^N$, the product Ax is in $\mathcal{C}(A)$, since it is a linear combination of columns of A:

$$Ax = \sum_{j=1}^{N} x_j A_{*j}.$$

The converse is also true. If $b \in \Re^M$ is in $\mathcal{C}(A)$ then there is a vector $x \in \Re^N$ such that $Ax = b$.

The *row space* of A is the subspace of $\Re^N$ spanned by the rows of A. It is equivalent to the column space $\mathcal{C}(A^T)$ of A^T. If a vector $c \in \Re^N$ is in the row space of A then there is a vector $y \in \Re^M$ such that $A^T y = c$ or, written in another way, $y^T A = c^T$.

Another interesting subspace of $\Re^N$ is the *null space*. This is the set of vectors $x \in \Re^N$ such that $Ax = 0$. Note that the null space is the set of vectors that are orthogonal to every row of A. Hence, by Theorem 2.2.4, the null space is the orthogonal complement of the row space. We denote the null space of a matrix A by $\mathcal{N}(A)$.

The *left null space* is analogously defined as the set of vectors $y \in \Re^M$ such that $y^T A = 0$, or written in another way, $A^T y = 0$. Again, by Theorem 2.2.4, the left null space is the set of vectors that are orthogonal to every column of A and is equivalent to $\mathcal{N}(A^T)$, the null space of A^T. The following theorem summarizes our observations relating null spaces to row and column spaces.

Theorem 2.2.7. *The left null space is the orthogonal complement of the column space. The null space is the orthogonal complement of the row space.*

Applying Theorems 2.2.6 and 2.2.7, we deduce the following relationships among the dimensionality of spaces associated with a matrix.

Theorem 2.2.8. *For any matrix* $A \in \Re^{M \times N}$, $\dim(\mathcal{C}(A))+\dim(\mathcal{N}(A^T)) = M$ *and* $\dim(\mathcal{C}(A^T)) + \dim(\mathcal{N}(A)) = N$.

Recall from Section 2.2.1 that a basis for a vector space can be obtained by starting with a collection of vectors that span the space and repeatedly removing vectors that are linear combinations of others until all remaining columns are linearly independent. When the dimension $\dim(\mathcal{C}(A))$ of the column space of a matrix $A \in \Re^{M \times N}$ is less than N, the columns are linearly dependent. We can therefore repeatedly remove columns of a matrix A until we arrive at a matrix $B \in \Re^{M \times K}$ with K linearly independent columns that span $\mathcal{C}(A)$. Hence, $\dim(\mathcal{C}(A)) = \dim(\mathcal{C}(B)) = K \leq N$. Perhaps more surprising is the fact that the process of removing columns that are linear combinations of others does not change the dimension of the row space. In particular, $\dim(\mathcal{C}(B^T)) = \dim(\mathcal{C}(A^T))$, as we will now establish.

Theorem 2.2.9. *Let* A *be a matrix. If we remove a column of* A *that is a linear combination of the remaining columns, the row space of the resulting matrix has the same dimension as the row space of* A.

Proof: Suppose that the last column of A is a linear combination of the other columns. Partition A as follows:

$$A = \begin{bmatrix} B & c \\ r^T & \alpha \end{bmatrix}$$

where $B \in \Re^{(M-1)\times(N-1)}, c \in \Re^{M-1}, r^T \in \Re^{N-1}$, and $\alpha \in \Re$. Note that $[c^T \ \alpha]^T = A_{*N}^T$ and $[r^T \ \alpha] = A_{M*}$.

Because the last column of A is a linear combination of the other columns, there is a vector $x \in \Re^{N-1}$ such that

$$\begin{bmatrix} c \\ \alpha \end{bmatrix} = \begin{bmatrix} B \\ r^T \end{bmatrix} x.$$

In other words, $\alpha = r^T x$ and $c = Bx$.

We will show that the last row of A is a linear combination of the other rows of A if and only if the last row of

$$\begin{bmatrix} B \\ r^T \end{bmatrix}$$

is a linear combination of the previous rows. That is, if and only if r^T is a linear combination of the rows of B.

Suppose the last row of A is a linear combination of the previous rows of A. Then $[r^T\ \alpha] = y^T[B\ c]$ for some $y \in \Re^{M-1}$. This means $r^T = y^T B$, and therefore, that r^T is a linear combination of the rows of B.

Conversely, suppose r^T is a linear combination of the rows of B. This means for some $y \in \Re^{M-1}$ that $r^T = y^T B$. But then $\alpha = r^T x = y^T Bx = y^T c$. Thus $[r^T\ \alpha] = y^T[B\ c]$, and therefore, the last row of A is a linear combination of the previous rows of A.

The same argument can be applied to any column that is a linear combination of other columns and any row. It follows that a row is linearly independent of other rows after removing a column that is linearly dependent on other columns if and only if it was linearly independent of other rows prior to removing the column. This implies the result that we set out to prove. □

The procedure for removing columns can also be applied to eliminating rows until all rows are linearly independent. This will change neither the row space nor the dimension of the column space. If we apply the procedure to columns and then to rows, the resulting matrix will have linearly independent rows and linearly independent columns.

Suppose we start with a matrix $A \in \Re^{M \times N}$ and eliminate columns and rows until we arrive at a matrix $C \in \Re^{K \times L}$ with linearly independent columns and linearly independent rows. Because the columns are in $\Re^K$ and linearly independent, there can be no more than K of them. This means that $L \leq K$. Similarly, since the rows are in $\Re^L$ and linearly independent, there can be no more than L of them. Therefore, $K \leq L$. This gives us the following theorem.

Theorem 2.2.10. *For any matrix A,* $\dim(\mathcal{C}(A)) = \dim(\mathcal{C}(A^T))$.

The dimension of the column space, or equivalently, the row space, of $A \in \Re^{M \times N}$ is referred to as the *rank* of A. We say that a matrix has *full column rank* if $\dim(\mathcal{C}(A)) = N$ and *full row rank* if $\dim(\mathcal{C}(A)) = M$. A matrix is said to have *full rank* if it has either *full column rank* or *full row rank*.

2.3 Linear Systems of Equations

Not all lines on a plane go through the origin. In fact, most do not. They are however all translations of lines that do go through the origin. Using the terminology of linear algebra we say that lines going through the origin are subspaces, while the translated lines are affine subspaces. An *affine subspace*

is a translated subspace. That is, an affine subspace $D \subseteq \Re^N$ is described by a subspace $S \subseteq \Re^N$ and a vector $d \in \Re^N$, with $D = \{x | x = d + s, s \in S\}$. We define $\dim(D) = \dim(S)$.

Note that if $d \in S$ then $D = S$. An example of this would be translating a line in the direction of the line. On the other hand, if $d \notin S$ then $D \neq S$; in fact, the intersection of D and S is empty in this case. There are also a number of different vectors d that can be used to describe a given affine subspace D. Also, the difference between any two vectors in D is in S, in fact $S = \{x - y | x, y \in D\}$.

Of particular interest to us is the case when the dimension of $D \subseteq \Re^N$ is $N - 1$, in which case D is called a *hyperplane*. If $N = 2$, a hyperplane D is a line. If $N = 3$, a hyperplane D is a plane in a three-dimensional space.

Hyperplanes arise in several situations of interest to us. For example, the set of solutions to a single linear equation $ax = b$, where $a \in \Re^N$, $b \in \Re$, and $a \neq 0$, is a hyperplane. To see why this set should be a hyperplane, consider first the equation $ax = 0$. The set of solutions is the set of vectors orthogonal to a, which is an $(N-1)$-dimensional subspace. Let d be a particular solution to $ax = b$. Then, given any vector $\overline{x}$ that is orthogonal to a, its translation $\overline{x} + d$ satisfies $a(\overline{x} + d) = b$. The converse is also true: if $\overline{x}$ solves $a\overline{x} = b$ then $a(\overline{x} - d) = 0$. It follows that the solutions of $ax = b$ comprise a hyperplane.

Consider the matrix equation $Ax = b$ where x is unknown. One way to think of this is by partitioning A row by row to obtain

$$Ax = \begin{bmatrix} A_{1*} \\ A_{2*} \\ \vdots \\ A_{M*} \end{bmatrix} x = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \end{bmatrix}$$

or,

$$Ax = \begin{bmatrix} A_{1*}x \\ A_{2*}x \\ \vdots \\ A_{M*}x \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \end{bmatrix}$$

What this means is that $A_{1*}x = b_1$, $A_{2*}x = b_2$, and so on. Thus $Ax = b$, can be interpreted as M equations of the form $A_{i*}x = b_i$. Each of these equations describes a hyperplane, and for x to satisfy the equation, x must lie in the hyperplane. For x to satisfy all of the equations simultaneously, and hence satisfy $Ax = b$, x must be in the intersection of the M hyperplanes defined by the rows of $Ax = b$.

Another way to think about the equation $Ax = b$ involves partitioning A

column by column. We then have

$$\begin{bmatrix} A_{*1} & A_{*2} & \dots & A_{*N} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = b$$

or $A_{*1}x_1 + A_{*2}x_2 + \ldots + A_{*N}x_N = b$. This means b is a linear combination of the columns, for which each x_j is the coefficient of the jth column A_{*j}. Geometrically, the columns of A are vectors in $\Re^M$ and we are trying to express b as a linear combination of them.

The two interpretations we have described for the equation $Ax = b$ are both helpful. Concepts we will study are sometimes more easily visualized with one interpretation or the other.

2.3.1 Solution Sets

In this section we will study some properties of solutions to $Ax = b$. Suppose we have two solutions x and y. Then $A(x - y) = Ax - Ay = b - b = 0$ so that $x - y$ is in the null space of A. The converse is also true, suppose x is a solution, and z is in the null space of A. Then $A(x+z) = Ax+Az = b+0 = b$. This means the set of all solutions is a translation of the null space of A. Because the null space of A is a subspace of $\Re^N$, this means the following:

Theorem 2.3.1. *For any $A \in \Re^{M\times N}$ and $b \in \Re^M$, the set S of solutions to $Ax = b$ is an affine subspace of $\Re^N$, with $\dim(S) = \dim(\mathcal{N}(A))$.*

We make some further observations. If the rank of A is M, the columns of A span $\Re^M$. This means that for any b, there will be at least one solution to $Ax = b$. Now, suppose the rank of A is N. Then, by Theorem 2.2.8, $\dim(\mathcal{N}(A)) = 0$. It follows from Theorem 2.3.1 that there can not be more than one solution to $Ax = b$. We summarize these observations in the following theorem.

Theorem 2.3.2. *For any $A \in \Re^{M\times N}$ and $b \in \Re^M$, the set S of solutions to $Ax = b$ satisfies*

1. *$|S| \geq 1$ if $\dim(\mathcal{C}(A)) = M$;*
2. *$|S| \leq 1$ if $\dim(\mathcal{C}(A)) = N$.*

Given a matrix $A \in \Re^{M\times N}$, we can define a function $f(x) = Ax$, mapping $\Re^N$ to $\Re^M$. The range of this function is the column space of A. If we restrict the domain to the row space of A, the function is a one-to-one mapping from row space to column space, as captured by the following theorem.

Theorem 2.3.3. *For any matrix $A \in \Re^{M\times N}$, the function $f(x) = Ax$ is a one-to-one mapping from the row space of A to the column space of A.*

Proof: Since the columns of A span the column space, the range of f is the column space. Furthermore, each vector in the column space is given by $f(x) \in \Re^M$ for some $x \in \mathcal{C}(A^T)$. This is because any such $y \in \Re^N$ can be written as $y = x + z$ for some $x \in \mathcal{C}(A^T)$ and $z \in \mathcal{N}(A)$, so $f(y) = A(x + z) = Ax = f(x)$. To complete the proof, we need to establish that only one element of the row space can map to any particular element of the column space.

Suppose that for two vectors $x^1, x^2 \in \mathcal{C}(A^T)$, both in the row space of A, we have $Ax^1 = Ax^2$. Because x^1 and x^2 are in the row space of A, there are vectors $y^1, y^2 \in \Re^M$ such that $x^1 = A^T y^1$ and $x^2 = A^T y^2$. It follows that $AA^T y^1 = AA^T y^2$, or $AA^T(y^1 - y^2) = 0$. This implies that $(y^1 - y^2)^T AA^T(y^1 - y^2) = 0$. This is the square of the norm of $A^T(y^1 - y^2)$, and the fact that it is equal to zero implies that $A^T(y^1 - y^2) = 0$. Hence, $x^1 - x^2 = 0$, or $x^1 = x^2$. The result follows □

2.3.2 Matrix Inversion

Given a matrix $A \in \Re^{M\times N}$, a matrix $B \in \Re^{N\times M}$ is said to be a *left inverse* of A if $BA = I$. Analogously, a matrix $B \in \Re^{N\times M}$ is said to be a *right inverse* of A if $AB = I$. A matrix cannot have both a left inverse and a right inverse unless both are equal. To see this, suppose that the left inverse of A is B and that the right inverse is C. We would then have

$$B = B(AC) = (BA)C = C.$$

If $N \neq M$, A cannot have both a left inverse and a right inverse. Only square matrices can have both left and right inverses. If A is a square matrix, any left inverse must be equal to any right inverse. So when a left and right inverse exists, there is a single matrix that is simultaneously the unique left inverse and the unique right inverse. This matrix is simply referred to as the *inverse*, and denoted by A^{-1}.

How do we find inverses of a matrix $A \in \Re^{M\times N}$? To find a right inverse B, so that $AB = I$, we can partition the matrix B into columns $[B_{*1} \cdots B_{*M}]$ and I into columns $[e^1 \cdots e^M]$, and solve $AB_{*j} = e^j$ to find each jth column of B. Therefore, the matrix A has a right inverse if and only if each e^j is in its column space. Since the vectors $e^1, \ldots, e^N$ span $\Re^M$, the column space of A must be $\Re^M$. Hence, A has a right inverse if and only if it has rank M – in other words, full row rank.

For left inverses, the picture is similar. We first transpose the equation $BA = I$ to get $A^T B^T = I^T = I$, then partition by column as we did to find a right inverse. Now we see that A has a left inverse if and only if its row space is $\Re^N$. Hence, A has a left inverse if and only if it has full column rank.

We summarize our observations in the following theorem.

Theorem 2.3.4. *For any matrix A,*
(a) *A has a right inverse if and only if it has full row rank;*
(b) *A has a left inverse if and only if it has full column rank;*
(c) *if A has full rank, then it has at least one left inverse or right inverse;*
(d) *if A is square and has full rank, then A has an inverse, which is unique.*

Note, that in Theorem 2.3.3, we described a function defined by A, and saw that it was a one-to-one mapping from the row space to the column space. If A is has an inverse, then the column space and the row space will both be $\Re^N$ (remember that A must be square). This gives us the following.

Theorem 2.3.5. *Let $f(x) = Ax$ for a matrix $A \in \Re^{M \times N}$. Then, f is a one-to-one mapping from $\Re^N$ to $\Re^M$ if and only if A has an inverse. Furthermore, this can only be the case if $N = M$.*

One final note on inverses, if A has an inverse then so does A^T, and in fact, $(A^T)^{-1} = (A^{-1})^T$. This is verified by simply multiplying the two together $(A^T)(A^{-1})^T = (A^{-1}A)^T = I^T = I$, and the check for left inverse is similar. To simplify notation, the inverse of a transpose is often denoted as A^{-T}.

2.4 Contingent Claims

A contingent claim is a contract that offers payoffs of amounts that depend on outcomes of future events. A lottery ticket is one example. In this case, the payoff depends on a random draw. Publicly traded securities such as stocks, bonds, and options are also contingent claims.

Example 2.4.1. (Stocks, Bonds, and Options) *A share of corporate stock entitles its holder to a fraction of the company's value. This value is determined by the market. A zero-coupon bond is a contract that promises to pay the holder a particular face value on a particular maturity date. A European put option is a contract that offers its holder the option to sell a share of stock to the grantor at a particular strike price on a particular expiration date. A European call option, on the other hand, offers its holder the option to buy a share of stock from the grantor at a particular strike*

price on a particular expiration date. The face values, maturity dates, strike prices, and expiration dates are specified by contract terms.

Consider four securities: a share of stock, a zero-coupon bond with face value \$1 *maturing in one year, and European put and call options on the stock, both expiring in one year, with strike prices of* \$40 *and* \$60*. Assume that the bond can not default, and therefore it guarantees a* \$1 *payment. We could purchase any combination of these securities today and liquidate in one year to receive a payoff. The payoff of each security will only depend on the price of the stock one year from now. Hence, we can visualize the payoff in terms of a function mapping the stock price one year from now to the associated payoff from the security. Figure 2.6 illustrates the payoff functions associated with each security.*

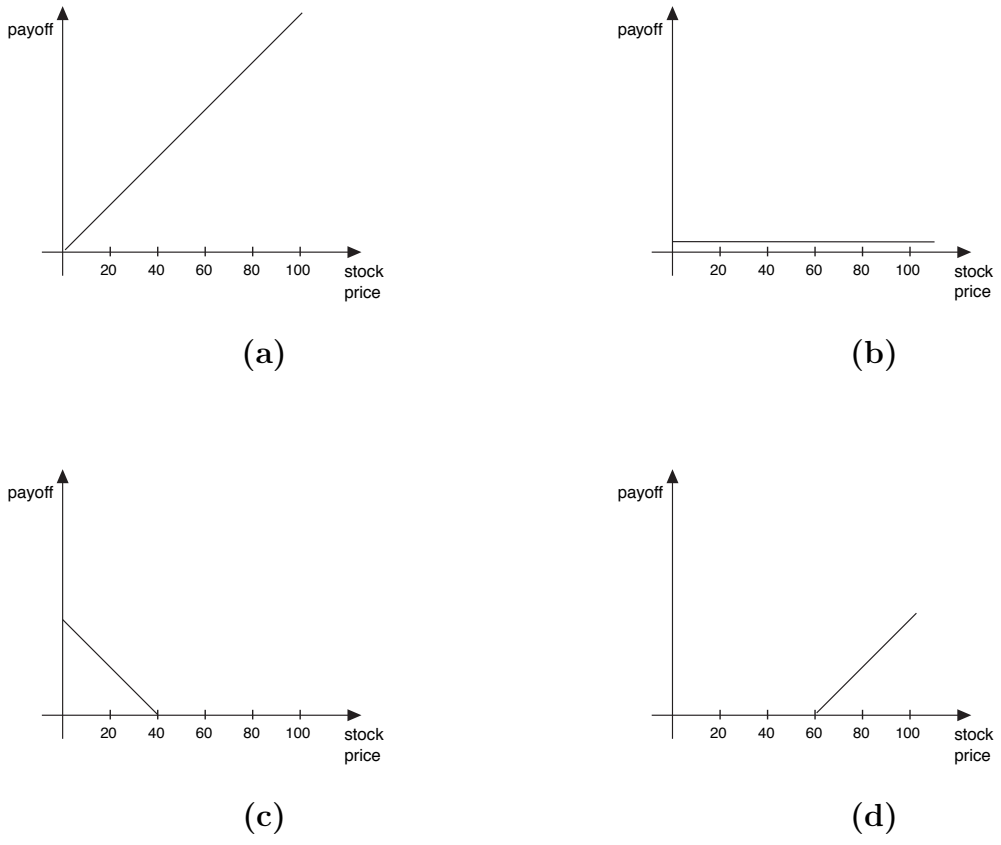


Figure 2.6: Payoff functions of a share of stock (a), a zero-coupon bond (b), a European put option (c), and a European call option (d).

Each payoff function provides the value of one unit of a security one year from now, as a function of the stock price. In the case of the stock, the value

is the stock price itself. For the put, we would exercise our option to sell the stock only if the the strike price exceeds the stock price. In this event, we would purchase a share of stock at the prevailing stock price and sell it at the strike price of \$40*, keeping the difference as a payoff. If the stock price exceeds the strike price, we would discard the contract without exercising. The story is similar for the call, except that we would only exercise our option to buy if the stock price exceeds the strike price. In this event, we would purchase one share of stock at the strike price of* \$60 *and sell it at the prevailing stock price, keeping the difference as a payoff.*

Suppose that the price of the stock in the preceding example can only take on values in $\{1, \ldots, 100\}$ a year from now. Then, payoff functions are conveniently represented in terms of *payoff vectors*. For example, the payoff vector $a^1 \in \Re^{100}$ for the stock would be defined by $a^1_i = i$. Similarly, the payoff vectors for the zero-coupon bond, European put option, and European call option would be $a^2_i = 1$, $a^3_i = \max(40 - i, 0)$, and $a^4_i = \max(i - 60, 0)$, respectively.

More generally, we may be concerned with large numbers of contingent claims driven by multiple sources of uncertainty. Even in such cases, the vector representation applies, so long as we enumerate all possible outcomes of interest. In particular, given a collection of N contingent claims and M possible outcomes, the payoff function associated with contingent claim $j \in \{1, \ldots, N\}$ can be thought of as a payoff vector $a^j \in \Re^M$, where each ith component is the payoff of the contingent claim in outcome $i \in \{1, \ldots, M\}$.

It is often convenient to represent payoff vectors for a set of contingent claims in terms of a single matrix. Each column of this *payoff matrix* P is the payoff vector for a single contingent claim. In particular,

$$P = \left[\begin{array}{ccc} a^1 & \ldots & a^N \end{array} \right].$$

2.4.1 Portfolios

A portfolio is a combination of N contingent claims, represented by a vector $x \in \Re^N$. Each component x_j denotes the number of units of contingent claim j included in the portfolio. We will make a simplifying assumption that fractional quantities of contracts can be acquired so that components of x are not necessarily integral.

We will also assume that we can short-sell contingent claims without paying interest or facing credit constraints. In practice, short-selling is accomplished by borrowing the contingent claim from a broker, selling the claim in the market, and then purchasing an identical claim at a later date to return

to the broker. Acquiring contingent claims results in a *long position* while short-selling results in a *short position*. In the portfolio vector, a long position in contingent claim j is represented by a positive value x_j while a short position is represented by a negative value x_j.

The payoff of a portfolio is the sum of payoffs generated by contingent claims it holds. Given a payoff matrix $P \in \Re^{M \times N}$, the payoff vector of a portfolio $x \in \Re^N$ is $Px \in \Re^M$. Each ith component of this portfolio payoff vector indicates the payoff generated by the portfolio in outcome i.

If contingent claims are traded in the market, each has a price. Let $\rho \in \Re^N$ denote the row vector of prices, so that ρ_j is the price paid to acquire or the price received to short-sell one unit of contingent claim j. The price of a portfolio $x \in \Re^N$ is the sum of the prices of its contents, given by ρx.

2.4.2 Hedging Risk and Market Completeness

Unwanted risks can often be avoided by acquiring carefully constructed portfolios of publicly traded securities. Consider the following example.

Example 2.4.2. (Hedging Currency Risk) *A manufacturer in the United States is considering an offer from a retailer in China that wishes to purchase* $100,000$ *units of a vehicle telematics system for* Y70 *million Chinese yuan. Producing components would cost the manufacturer* \$5 *million. The components could then be assembled in the US for* \$4 *million, or the assembly can be handled in China by the retailer, in which case the retailer wants a* Y20 *million price reduction. The manufacturer will be paid by the retailer when the components or assembled units are shipped, which is planned for three months from now. The manufacturer can wait until then to decide whether to assemble the units in the US. The manufacturer can also decide then not to ship anything to China and instead sell the components to dealers in the US for* \$3 *million.*

The manufacturer is interested in this opportunity but is concerned that the recently volatile exchange rate will influence profitability. In particular, if y *is the is the dollar price of the yuan come delivery time three months from now, the manufacturer's profit in millions of dollars will be the largest among* $70y - 9$, $50y - 5$, *and* -2. *The first expression is the profit if the units are assembled in the US, the second is for the case where components are shipped, and the third represents the loss incurred if the components are sold to dealers in the US. To understand the risks involved, it is useful to plot the manufacturer's projected profit as a function of the dollar price of the yuan, as is done in Figure 2.7(a). Note that the slope of the profit function changes from* 0 *to* 50 *million at* 0.06 *dollars per yuan and from* 50 *million*

to 70 *million at* 0.2 *dollars per yuan. The first change occurs at the point where the manufacturer is indifferent between selling to dealers in the US and shipping components to China. The second change occurs at the point of indifference between shipping components versus assembled units.*

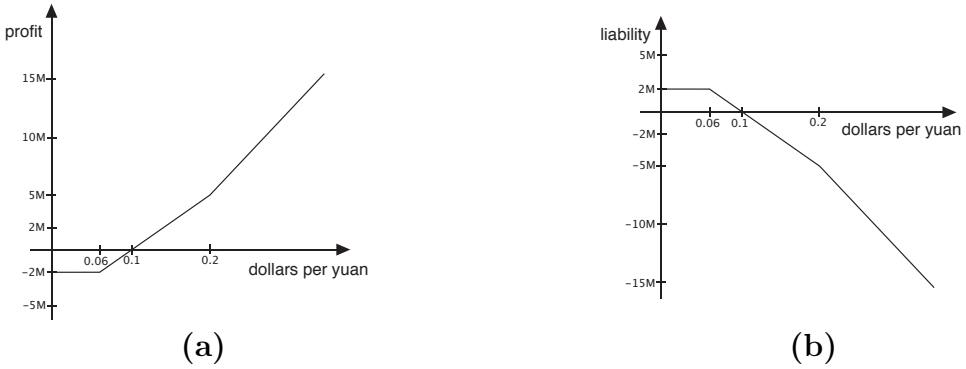


Figure 2.7: The manufacturer's (a) profit or (b) liability as a function of the dollar price of the yuan.

Figure 2.7(b) plots the manufacturer's liability, or loss, as a function of outcome. Note that for each outcome the profit and liability sum to zero. Where the liability is positive, the manufacturer loses money. The graph illustrates that there is risk of losing money in the event that the yuan falls below a tenth of a dollar.

Observe that the liability function is identical to the payoff function of a portfolio that takes the following positions in securities that mature or expire in three months: a long position in 2 *million zero coupon bonds with* \$1 *face value, a short position in* 50 *million European call options at strike price* \$0.06*, and a short position in* 20 *million European call options at strike price* \$0.2*. Let us assume that these securities are available in the market.*

In order to avoid, or "hedge," currency risk, the manufacturer can acquire a portfolio of the kind described. With such a portfolio, the net payoff three months from now, including that from the portfolio and that from production activities, would be zero regardless of the price of the yuan at that time. Acquisition of the portfolio initially results in some revenue or cost. If it is revenue, this can be viewed as a risk-free profit. If the portfolio costs money, the deal is not worthwhile and the manufacturer should turn down the offer.

In the preceding example, the risk associated with a business venture could be avoided because liabilities could be *replicated* by a portfolio of contingent claims available in the market. The example was a simple one, and

as such it was easy to identify a replicating portfolio without any significant analysis. In more complex situations, linear algebra offers a useful framework for identifying replicating portfolios.

Consider a general context where there are M possible outcomes and we wish to replicate a payoff vector $b \in \Re^M$ using N available contingent claims with payoffs encoded in a matrix $P \in \Re^{M \times N}$. A portfolio $x \in \Re^N$ replicates the payoffs b if and only if $Px = b$. To obtain a replicating portfolio, one can use software tools to solve this equation. If there are no solutions to this equation, there is no portfolio that replicates b.

Can every possible payoff vector be replicated by contingent claims available in a market? To answer this question, recall that a payoff function $b \in \Re^M$ can be replicated if and only if $Px = b$ for some x. Hence, to replicate every payoff vector, $Px = b$ must have a solution for every $b \in \Re^M$. This is true if and only if P has full row rank. In this event, the market is said to be *complete* – that is, the payoff function of any new contingent claim that might be introduced to the market can be replicated by existing contingent claims.

2.4.3 Pricing and Arbitrage

We now turn our focus from payoff to prices. Consider a market with N contingent claims, and let $\rho \in \Re^N$be the row vector of prices. Suppose that a new contingent claim is introduced, but since it has not yet been traded, there is no market price. How should we price this new contract?

If the new contract can be replicated by a portfolio of existing ones, the price should be the same as that of the portfolio. For if the price of the replicating portfolio were higher, one could short-sell the portfolio and purchase the contract to generate immediate profit without incurring cost or risk. Similarly, if the price of the replicating portfolio were lower, one would short-sell the contract and purchase the portfolio. In either case, an immediate payment is received and the future payoff is zero, regardless of the outcome.

What we have just described is an arbitrage opportunity. More generally, an *arbitrage opportunity* is an investment strategy that involves a negative initial investment and guarantees a nonnegative payoff. In mathematical terms, an arbitrage opportunity is represented by a portfolio $x \in \Re^N$ such that $\rho x < 0$ and $Px \geq 0$. Under the assumption that arbitrage opportunities do not exist, it is often possible to derive relationships among asset prices. We provide a simple example

Example 2.4.3. (Put-Call Parity) *Consider four securities:*

(a) *a stock currently priced at s_0 that will take on a price $s_1 \in \{1, \ldots, 100\}$ one month from now;*
(b) *a zero-coupon bond priced at β_0, maturing one month from now;*
(c) *a European put option currently priced at p_0 with a strike price $\kappa > 0$, expiring one month from now;*
(d) *a European call option currently priced at c_0 with the same strike price κ, expiring one month from now.*

The payoff vectors $a^1, a^2, a^3, a^4 \in \Re^M$ are given by

$$a_i^1 = i, \quad a_i^2 = 1, \quad a_i^3 = \max(\kappa - i, 0), \quad a_i^4 = \max(i - \kappa, 0),$$

for $i \in \{1, \ldots, M\}$. Note that if we purchase one share of the stock and one put option and short-sell one call option and κ units of the bond, we are guaranteed zero payoff; i.e.,

$$a^1 - \kappa a^2 + a^3 - a^4 = 0.$$

The initial investment in this portfolio would be

$$s_0 - \kappa\beta_0 + p_0 - c_0.$$

If this initial investment is nonzero, there would be an arbitrage opportunity. Hence, in the absence of arbitrage opportunities, we have the pricing relationship

$$s_0 - \kappa\beta_0 + p_0 - c_0 = 0,$$

which is known as the put-call parity.

Because arbitrage opportunities are lucrative, one might wish to determine whether they exist. In fact, one might consider writing a computer program that automatically detects such opportunities whenever they are available. As we will see in the next chapter, linear programming offers a solution to this problem. Linear programming will also offer an approach to hedging risks at minimal cost when payoff functions can not be replicated.

2.5 Exercises

Question 1

Let $a = [1, 2]^T$ and $b = [2, 1]^T$. On the same graph, draw each of the following

1. The set of all points $x \in \Re^2$ that satisfy $a^T x = 0$.

2. The set of all points $x \in \Re^2$ that satisfy $a^T x = 1$.
3. The set of all points $x \in \Re^2$ that satisfy $b^T x = 0$.
4. The set of all points $x \in \Re^2$ that satisfy $b^T x = 1$.
5. The set all points $x \in \Re^2$ that satisfy $[a\ b]x = [0, 1]^T$.
6. The set all points $x \in \Re^2$ that satisfy $[a\ b]x = [0, 2]^T$.

In addition, shade the region that consists of all $x \in \Re^2$ that satisfy $a^T x \leq 0$.

Question 2

Consider trying to solve $Ax = b$ where

$$A = \begin{bmatrix} 1 & 2 \\ 0 & 3 \\ 2 & -1 \end{bmatrix}$$

1. Find a b so that $Ax = b$ has no solution.
2. Find a non-zero b so that $Ax = b$ has a solution.

Question 3

Find two $x \in \Re^4$ that solve all of the following equations.

$$\begin{aligned} [0, 2, 0, 0]\, x &= 4 \\ [1, 0, 0, 0]\, x &= 3 \\ [2, -1, -2, 1]\, x &= 0 \end{aligned}$$

Write $[4, 3, 0]^T$ as a linear combination of $[0, 1, 2]^T, [2, 0, -1]^T, [0, 0, -2]^T$ and $[0, 0, 1]^T$ in two different ways. Note: When we say write x as a linear combination of a, b and c, what we mean is find the coefficients of the a, b and c. For example $x = 5a - 2b + c$.

Question 4

Suppose $A, B \in \Re^{3\times 3}$ are defined by $A_{ij} = i + j$ and $B_{ij} = (-1)^{ij}$ for each i and j. What is A^T? AB?

Suppose we now change some elements of A so that $A_{1j} = e_j{}^1$. What is A now?

Question 5

Suppose U and V are both subspaces of S. Are $U \cap V$ and $U \cup V$ subspaces of S? Why, or why not? Hint: Think about whether or not linear combinations of vectors are still in the set. Note that $\cap$ denotes set intersection and $\cup$ denotes set union.

Question 6

$$a = \begin{bmatrix} 1 \\ 4 \\ -1 \\ 2 \\ 3 \end{bmatrix}, b = \begin{bmatrix} 1 \\ 2 \\ 0 \\ -3 \\ 1 \end{bmatrix}, c = \begin{bmatrix} 0 \\ 2 \\ -1 \\ 5 \\ 2 \end{bmatrix}, d = \begin{bmatrix} 1 \\ 0 \\ 2 \\ -8 \\ -1 \end{bmatrix}, e = \begin{bmatrix} 1 \\ 10 \\ -1 \\ 17 \\ 9 \end{bmatrix}, f = \begin{bmatrix} -4 \\ -2 \\ 1.5 \\ 27 \\ 2 \end{bmatrix}$$

1. The span of $\{a, b, c\}$ is an N dimensional subspace of $\Re^M$. What are M and N?
2. What is $6a + 5b - 3e + 2f$?
3. Write a as a linear combination of b, e, f.
4. write f as a linear combination of a, b, e.
5. The span of $\{a, b, c, d, e, f\}$ is an N dimensional subspace of $\Re^M$. What are M and N?

Question 7

Suppose I have 3 vectors x, y and z. I know $x^T y = 0$ and that y is not a multiple of z. Is it possible for $\{x, y, z\}$ to be linearly dependent? If so, give an example. If not, why not?.

Question 8

Find 2 matrices $A, B \in \Re^{2\times 2}$ so that none of the entries in A or B are zero, but AB is the zero matrix. Hint: Orthogonal vectors.

Question 9

Suppose the only solution to $Ax = 0$ is $x = 0$. If $A \in \Re^{M\times N}$ what is its rank, and why?

Question 10

The system of equations

$$\begin{aligned} 3x + ay &= 0 \\ ax + 3y &= 0 \end{aligned}$$

always has as a solution $x = y = 0$. For some a the equations have more than one solution. Find two such a.

Question 11

In $\Re^3$, describe the 4 subspaces (column, row, null, left-null) of the matrix

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Question 12

In real life there are many more options than those described in class, and they can have differing expiry dates, strike prices, and terms.

1. Suppose there are 2 call options with strike prices \$ 1 and \$ 2. Which of the two will have the higher price?

2. Suppose the options were put options. Which would now have the higher price?

3. If an investor thinks that the market will be particularly volatile in the coming weeks, but does not know whether the market will go up or down, they may choose to buy an option that pays $|S - K|$ where S will be the value of a particular share in one months time, and K is the *current* price of the share. If the price of a zero coupon bond maturing in one month is 1, then what is the difference between a call option and a put option both having strike price K. (Report the answer in terms of the strike price K.)

4. Suppose I am considering buying an option on a company. I am considering options with a strike price of \$ 100, but there are different expiry dates. The call option expiring in one month costs \$ 10 while the put option expiring in one month costs \$ 5. If the call option expiring in two months costs \$ 12 and zero coupon bonds for both time frames cost \$ 0.8, then how much should a put option expiring in two months cost?

Question 13

Find a matrix whose row space contains $[1\ 2\ 1]^T$ and whose null space contains $[1\ -2\ 1]^T$ or show that there is no such matrix.

Question 14

Let U and V be two subspaces of $\Re^N$. True or False:

1. If U is orthogonal to V then $U^\perp$ is orthogonal to $V^\perp$.
2. If U is orthogonal to V and V is orthogonal to W then U us orthogonal to W.
3. If U is orthogonal to V and V is orthogonal to W then U can not be orthogonal to W.

Question 15

Can the null space of AB be bigger than the null space of B? If so, give an example. If not, why not.

Question 16

Find two different matrices that have the same null, row, column, and left-null spaces.

Chapter 3

Linear Programs

A linear program involves maximization or minimization of a linear function subject to linear constraints. A linear inequality constraint on a vector $x \in \Re^N$ takes the form $ax \leq b$ or $a_1x_1+a_2x_2+\ldots+a_Nx_N \leq b$ for some $a \in \Re^N$ and $b \in \Re$. If we have a collection of constraints $a^1x \leq b_1, a^2x \leq b_2, \ldots, a^Mx \leq b_M$, we can group them together as a single vector inequality $Ax \leq b$ where

$$A = \begin{bmatrix} a^1 \\ \vdots \\ a^M \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_M \end{bmatrix}.$$

When we write that the vector Ax is less than or equal to b, we mean that each component of Ax is less than or equal to the corresponding component of b. That is, for each i, $(Ax)_i \leq b_i$. Sometimes, in a slight abuse of language, we refer to the ith row A_{i*} of the matrix A as the ith constraint, and b_i as the value of the constraint.

A linear program can be expressed as follows:

$$\begin{array}{ll} \text{maximize} & cx \\ \text{subject to} & Ax \leq b. \end{array}$$

The maximization is over $x \in \Re^N$. Each component x_j is referred to as a *decision variable*. The matrix $A \in \Re^{M\times N}$ and vector $b \in \Re^M$ specify a set of M *inequality constraints*, one for each row of A. The ith constraint is $A_{i*}x \leq b_i$. Each component c_j represents the benefit of increasing x_j by 1. The set of vectors $x \in \Re^N$ that satisfy $Ax \leq b$ is called the *feasible region*. A linear program can alternatively seek to minimize the objective:

$$\begin{array}{ll} \text{minimize} & cx \\ \text{subject to} & Ax \leq b, \end{array}$$

in which case c_j represents the cost of increasing x_j by 1.

3.1 Graphical Examples

To generate some understanding of linear programs, we will consider two simple examples. These examples each involve two decision variables. In most interesting applications of linear programming there will be many more decision variables – perhaps hundreds, thousands, or even hundreds of thousands. However, we start with cases involving only two variables because it is easy to illustrate what happens in a two dimensional space. The situation is analogous to our study of linear algebra. In that context, it was easy to generate some intuition through two-dimensional illustrations, and much of this intuition generalized to spaces of higher dimension.

3.1.1 Producing Cars and Trucks

Let us consider a simplified model of an automobile manufacturer that produces cars and trucks. Manufacturing is organized into four departments: sheet metal stamping, engine assembly, automobile assembly, and truck assembly. The capacity of each department is limited. The following table provides the percentages of each department's monthly capacity that would be consumed by constructing a thousand cars or a thousand trucks:

Department	Automobile	Truck
metal stamping	4%	2.86%
engine assembly	3%	6%
automobile assembly	4.44%	0%
truck assembly	0%	6.67%

The marketing department estimates a profit of \$3000 per car produced and \$2500 per truck produced. If the company decides only to produce cars, it could produce $22,500$ of them, generating a total profit of \$67.5 million. On the other hand, if it only produces trucks, it can produce $15,000$ of them, with a total profit of \$37.5 million. So should the company only produce cars? No. It turns out that profit can be increased if the company produces a combination of cars and trucks.

Let us formulate a linear program that will lead us to the optimal solution. Define decision variables x_1 and x_2 to be the number in thousands of cars and trucks, respectively, to produce each month. Together, they can be thought of as a vector $x \in \Re^2$. These quantities have to be positive, so we introduce a constraint $x \geq 0$. Several additional constraints arise from capacity limitations. The car assembly and truck assembly departments limit production according to

$$4.44x_1 \leq 100 \quad \text{and} \quad 6.67x_2 \leq 100.$$

The metal stamping and engine assembly activities also introduce constraints:

$$4x_1 + 2.86x_2 \leq 100 \quad \text{and} \quad 3x_1 + 6x_2 \leq 100.$$

The set of vectors $x \in \Re^2$ that satisfy these constraints is illustrated in Figure 3.1(a).

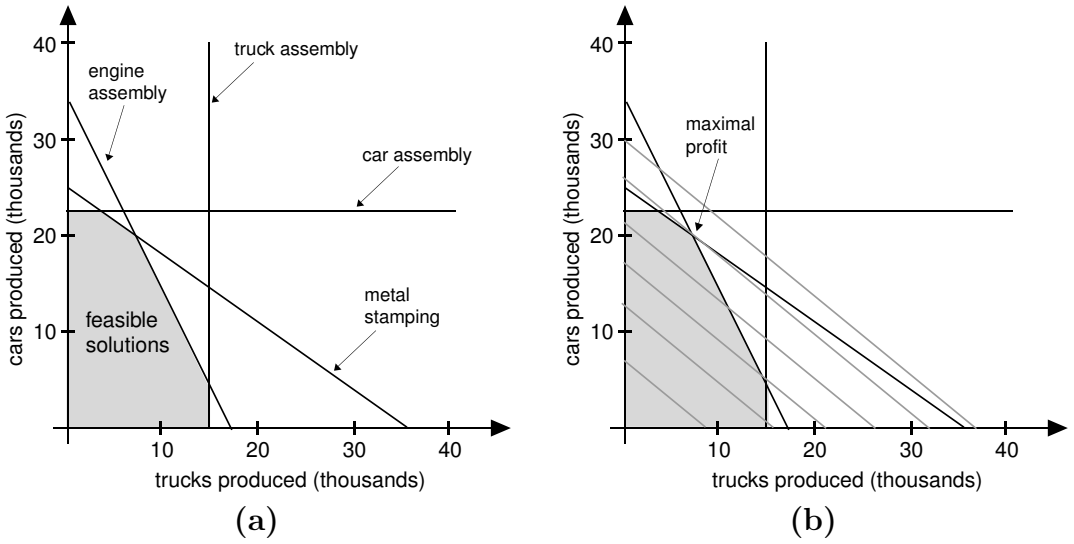


Figure 3.1: (a) Feasible solutions for production of cars and trucks. (b) Finding the solution that maximizes profit.

The anticipated profit in thousands of dollars associated with production quantities x_1 and x_2 is $3x_1 + 2.5x_2$. In Figure 3.1(b), each gray line superimposed on the set of solutions represents a subset for which the associated profit takes on a particular value. In other words, each line represents solutions of the equation $3x_1 + 2.5x_2 = \alpha$ for some value of α. The diagram also identifies the feasible solution that maximizes profit, which is given approximately by $x_1 = 20.4$ and $x_2 = 6.5$. Note that this solution involves making use of the entire capacity available for metal stamping and engine assembly, but does not maximize use of capacity to assemble either cars or trucks. The optimal profit is over \$77.3 million per month, which exceeds by about \$10 million the profit associated with producing only cars.

3.1.2 Feeding an Army

Suppose that two basic types of food are supplied to soldiers in an army: meats and potatoes. Each pound of meats costs \$1, while each pound of

potatoes costs \$0.25. To minimize expenses, army officials consider serving only potatoes. However, there are some basic nutritional requirements that call for meats in a soldier's diet. In particular, each soldier should get at least 400 grams of carbohydrates, 40 grams of dietary fiber, and 200 grams of protein in their daily diet. Nutrients offered per pound of each of the two types of food, as well as the daily requirements, are provided in the following table:

Nutrient	Meats	Potatoes	Daily Requirement
carbohydrates	40 grams	200 grams	400 grams
dietary fiber	5 grams	40 grams	40 grams
protein	100 grams	20 grams	200 grams

Consider the problem of finding a minimal cost diet comprised of meats and potatoes that satisfies the nutritional requirements. Let x_1 and x_2 denote the number of pounds of meat and potatoes to be consumed daily. These quantities cannot be negative, so we have a constraint $x \geq 0$. The nutritional requirements impose further constraints:

$$\begin{aligned} 40x_1 + 200x_2 &\geq 400 \quad \text{(carbohydrates)} \\ 5x_1 + 40x_2 &\geq 40 \quad \text{(dietary fiber)} \\ 100x_1 + 20x_2 &\geq 200 \quad \text{(protein).} \end{aligned}$$

The set of feasible solutions is illustrated in Figure 3.2(a).

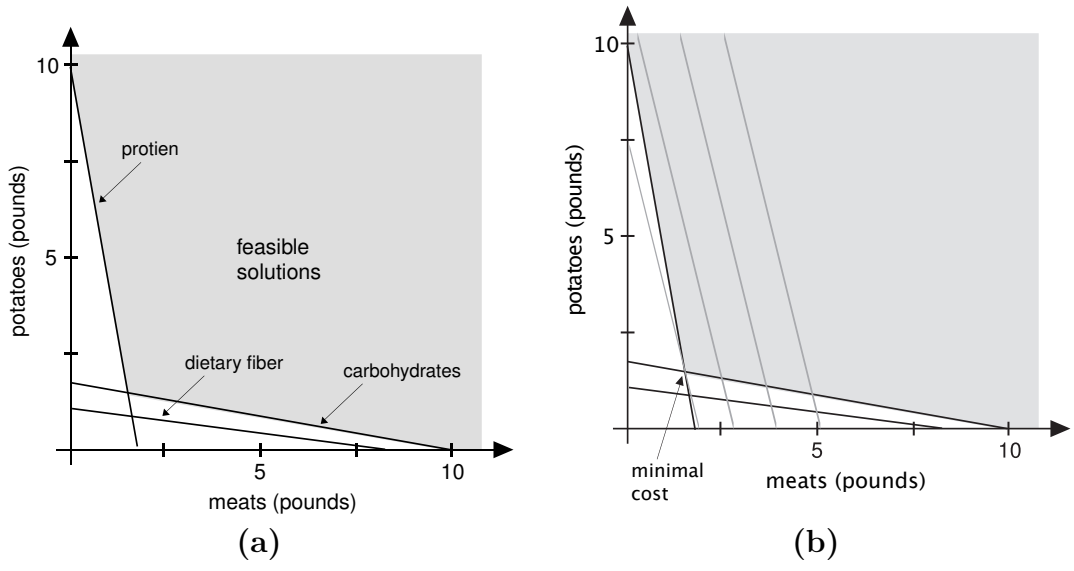


Figure 3.2: (a) Feasible solutions for an army diet. (b) Finding the solution that minimizes cost.

In Figure 3.2(b), superimposed lines identify sets that lead to particular daily costs. Unlike the automobile manufacturing problem we considered in the previous section, we are now *minimizing* cost rather than *maximizing* profit. The optimal solution involves a diet that includes both meats and potatoes, and is given approximately by $x_1 = 1.67$ and $x_2 = 1.67$. The associated daily cost per soldier is about \$2.08. Note that the constraint brought about by dietary fiber requirements does not affect the feasible region. This is because – based on our data – any serving of potatoes that offers sufficient carbohydrates will also offer sufficient dietary fibers.

3.1.3 Some Observations

There are some interesting observations that one can make from the preceding examples and generalize to more complex linear programs. In each case, the set of feasible solutions forms a polygon. By this we mean that the boundary of each is made up of a finite number of straight segments, forming corners where they connect. In the case of producing cars and trucks, this polygon is bounded. In the case of feeding an army, the polygon is unbounded: two of the sides continue out to infinity. But in both cases they are polygons.

Another key commonality between the examples is that optimal solutions appear at corners of the polygon. To see why this is the case when we have two decision variables, consider the line given by $\{x \in \Re^2 | cx = \alpha\}$ for some $\alpha \in \Re$. As we change α we move the line continuously across $\Re^2$. To make α as large as possible, and hence maximize cx, we keep moving the line (increasing α) until any more movement will mean the line no longer intersects the feasible region. At this point, the line must be touching a corner.

In the next two sections, we will formalize and generalize these observations, so that we can make statements that apply to linear programs involving arbitrary numbers of variables and constraints. In higher dimensions, we will be dealing with polyhedra as opposed to polygons, and we will find that optimal solutions still arise at "corners."

3.2 Feasible Regions and Basic Feasible Solutions

The set of vectors $x \in \Re^N$ that satisfies constraints of the form $Ax \leq b$ is called a *polytope*. In three dimensions, the boundaries of the set are formed by "flat faces." In two dimensions, the boundaries are formed by line segments, and a polytope is a polygon.

Note that the feasible region of a linear program is a polytope. Hence, a linear program involves optimization of a linear objective function over a polyhedral feasible region. One way to view a polytope is as the intersection of a collection of half-spaces. A *half-space* is a set of points that satisfy a single inequality constraint. Hence, each constraint $A_{i*}x \leq b_i$ defines a half-space, and the polytope characterized by $Ax \leq b$ is the intersection of M such half-spaces.

As an example, consider the problem of producing cars and trucks described in Section 3.1.1. Each constraint restricts feasible solutions to the half-space on one side of a line. For instance, the constraint that the number of cars produced must be nonnegative restricts the feasible region to vectors in the half-space on the right side of the horizontal axis in Figure 3.1(a). Note that, though this constraint was represented with a greater-than sign ($x_1 \geq 0$), it can also be represented with a less-than sign ($-x_1 \leq 0$) to be consistent with the form of $Ax \leq b$. The constraint introduced by the capacity to assemble engines also restricts solutions to a half-space – the set of points below a diagonal line. The intersection of half-spaces associated with the six constraints produces the polytope of feasible solutions.

In this section, we develop some understanding of the structure of polyhedra. We will later build on these ideas to establish useful properties of optimal solutions.

3.2.1 Convexity

Given two vectors x and y in $\Re^N$, a vector $z \in \Re^N$ is said to be a *convex combination* of x and y if there exists a scalar $\alpha \in [0, 1]$ such that $z = \alpha x + (1 - \alpha)y$. Intuitively, a convex combination of two vectors is in between the vectors, meaning it lies directly on the line segment joining the two vectors. In fact, the line segment connecting two vectors is the set of all convex combinations of the two vectors. We generalize this to more than two vectors by saying y is a convex combination of $x^1, x^2, \ldots, x^M$ if there are some $\alpha_1, \alpha_2, \ldots, \alpha_M \geq 0$ such that $y = \alpha_1 x^1 + \alpha_2 x^2 + \ldots + \alpha_M x^M$ and $\alpha_1 + \alpha_2 + \ldots + \alpha_M = 1$.

A set $U \subseteq \Re^N$ is said to be *convex* if any convex combination of any two elements of U is in U. In other words, if we take two arbitrary points in U, the line segment connecting those points should stay inside U. Figure 3.3 illustrates three subsets of $\Re^2$. The first is convex. The others are not.

Given a set of vectors $x^1, \ldots, x^K \in \Re^N$, their *convex hull* is the smallest convex set containing $x^1, \ldots, x^K \in \Re^N$. Because any intersection of convex sets is convex, there is no ambiguity in this definition. In particular, the convex hull can be thought of as the intersection of all convex sets containing

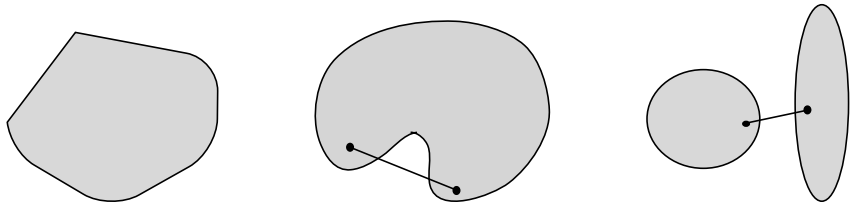


Figure 3.3: (a) A convex set in $\Re^2$. (b and c) Nonconvex sets in $\Re^2$ – in each case, a segment connecting points in the set includes points outside the set.

$x^1, \ldots, x^K \in \Re^N$.

Convex sets are very important in many areas of optimization, and linear programming is no exception. In particular, polyhedra are convex. To see why this is so, consider a polytope $U = \{x \in \Re^N | Ax \leq b\}$. If $z = \alpha x + (1-\alpha)y$ for some $\alpha \in [0, 1]$ and $x, y \in U$ then

$$Az = A(\alpha x + (1-\alpha)y) = \alpha Ax + (1-\alpha)Ay \leq \alpha b + (1-\alpha)b = b$$

so that z is an element of U.

3.2.2 Vertices and Basic Solutions

Let $U \subseteq \Re^N$ be a polytope. We say $x \in U$ is a vertex of U if x is not a convex combination of two other points in U. Vertices are what we think of as "corners."

Suppose $U = \{x \in \Re^N | Ax \leq b\}$ is the feasible region for a linear program and that $y \in U$. If $A_{i*}y = b_i$ then we say the ith constraint is *binding* or *active* at y. If we think of a polytope as a collection of half spaces, then for a constraint to be active, the point in question lies on the hyperplane forming the border of the half space. In three dimensions, it must lie on the face associated with the constraint, and in two dimensions, it must lie on the edge associated with the constraint. If a collection of constraints $a^1 x \leq \beta_1, \ldots, a^K x \leq \beta_K$ are active at a vector $\overline{x} \in \Re^N$, then $\overline{x}$ is a solution to $Bx = \beta$, where

$$B = \begin{bmatrix} a^1 \\ \vdots \\ a^K \end{bmatrix} \quad \text{and} \quad \beta = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_K \end{bmatrix}.$$

A collection of linear constraints $a^1 x \leq \beta_1, \ldots, a^K x \leq \beta_K$ is said to be linearly independent if $a^1, \ldots, a^K$ are linearly independent. For a given linear

program, if there are N linearly independent constraints that are active at a vector $\overline{x} \in \Re^N$ then we say that $\overline{x}$ is a *basic solution*. To motivate this terminology, note that the active constraints form a basis for $\Re^N$. Note also that, because the active constraints form a basis, given N active constraints $a^1 x \leq \beta_1, \ldots, a^K x \leq \beta_K$, the square matrix

$$B = \begin{bmatrix} a^1 \\ \vdots \\ a^K \end{bmatrix},$$

has full rank and is therefore invertible. Hence, $Bx = \beta$ has a unique solution, which is a basic solution of the linear program.

If a basic solution $\overline{x} \in \Re^N$ is feasible, then we say $\overline{x}$ is a *basic feasible solution*. In two dimensions, a basic feasible solution is the intersection of two boundaries of the polygonal feasible region. Clearly, in two dimensions, basic feasible solutions and vertices are equivalent. The following theorem establishes that this remains true for polyhedra in higher-dimensional spaces.

Theorem 3.2.1. *Let U be the feasible region of a linear program. Then, $x \in U$ is a basic feasible solution if and only if x is a vertex of U.*

Proof: Suppose x is a basic feasible solution and also that x is a convex combination of y and z, both in U. Let C be a matrix whose rows are N linearly independent active constraints at x, and c be the vector of corresponding constraint values. Because C has linearly independent rows, it has full rank and is invertible. Also $Cy \leq c, Cz \leq c$ and $Cx = c$.

x is a convex combination of y and z so that $x = \alpha y + (1-\alpha)z$ for some $\alpha \in [0,1]$. This means $Cx = \alpha Cy + (1-\alpha)Cz \leq \alpha c + (1-\alpha)c = c$ can only equal c if at least one of Cy and Cz are equal to c. But because C is invertible, there is only one solution to $Cy = c$, namely $y = x$. Similarly $Cz = c$ gives $z = x$. This means x cannot be expressed as a convex combination of two points in U unless one of them is x, so that x is a vertex.

Conversely, suppose x is not a basic feasible solution. We let C be the matrix of all the active constraints at x. Because C has less than N linearly independent rows, it has a non-empty null space. Let d be a non zero vector in $\mathcal{N}(C)$. Then for small ϵ, we have that $x \pm \epsilon d$ is still feasible ($C(x \pm \epsilon d) = Cx = c$ and for small ϵ, non-active constraints will still be non-active). But $x = \frac{1}{2}(x + \epsilon d) + \frac{1}{2}(x - \epsilon d)$, so that x is a convex combination of two other vectors in U. So, x is not a vertex. □

If the feasible region of a linear program is $\{x \in \Re^N | Ax \leq b\}$ then any N linearly independent active constraints identify a unique basic solution $\overline{x} \in \Re^N$. To see why, consider a square matrix $B \in \Re^{N \times N}$ whose rows are

N linearly independent active constraints. Any vector $x \in \Re^N$ for which these constraints are active must satisfy $Bx = b$. Since its rows of B are linearly independent, B has full rank and therefore a unique inverse B^{-1}. Hence, $B^{-1}b$ is the unique point at which the N constraints are active. Let us capture the concept in a theorem.

Theorem 3.2.2. *Given a polytope $\{x \in \Re^N | Ax \leq b\}$ for some $A \in \Re^{M \times N}$ and $b \in \Re^M$, any set of N linearly independent active constraints identifies a unique basic solution.*

Each basic solution corresponds to N selected constraints. There are M constraints to choose from and only finitely many ways to choose N from M. This implies the following theorem.

Theorem 3.2.3. *There are a finite number of basic solutions and a finite number of basic feasible solutions.*

Note that not every combination of N constraints corresponds to a basic solution. The constraints are required to be linearly independent.

3.2.3 Bounded Polyhedra

A polytope $U \subseteq \Re^N$ is said to be *bounded* if there is a scalar α such that, for each $x \in U$ and each j, $-\alpha \leq x_j \leq \alpha$. In other words, each component of a vector in U is restricted to a bounded interval, or U is contained in a "hyper-cube." The following theorem presents an alternative way to represent bounded polyhedra.

Theorem 3.2.4. *If U is a bounded polytope, it is the convex hull of its vertices.*

Proof: Let $U = \{x \in \Re^N | Ax \leq b\}$, and let H be the convex hull of the vertices of U. Each vertex of U is in U and U is convex. Hence, $H \subseteq U$.

We now have to show $U \subseteq H$. We will do this by showing that any $x \in U$ is a convex combination of the vertices of U, and hence is in H. We will do this by backwards induction on the number of linearly independent active constraints at x.

If the number of linearly independent active constraints is N, then x is a basic feasible solution, and so a vertex. A vertex is a convex combination of itself, and hence is in U. Thus all points with N linearly independent active constraints are in H.

Suppose all points with $K + 1$ or more linearly independent active constraints are in H, and that x has K linearly independent active constraints.

Let C be a matrix whose rows are the active constraints at x, and let c be the vector whose components are the corresponding constraint values. Because C has rank $K < N$, we know that its null space is non-empty. Take any non-zero vector $n \in \mathcal{N}(C)$ and consider the line $x + \alpha n$ for different α. For small α, the points on the line are inside U, but because U is bounded, for sufficiently large positive and negative values of α, the points on the line will not be in U. Take the most positive and negative values of α such that $x+\alpha n$ is in U, and let the corresponding points be y and z. Note $Cy = c = Cz$ so that all the constraints active for x are still active for y and z. However each one of them much also have an additional active constraint because n is in the null space of C and so changing α will not change $C(x + \alpha n)$. Thus each of y and z must have at least $K+1$ linearly independent active constraints. x lies on the line segment connecting y and z and so is a convex combination of them. By the inductive hypothesis, each of y and z are convex combinations of vertices, and hence so is x. Hence all points with K linearly independent active constraints are in H.

By induction, each $x \in U$ is also in H, so $U \subseteq H$. □

3.3 Optimality of Basic Feasible Solutions

Consider the linear program

$$\begin{array}{ll} \text{maximize} & cx \\ \text{subject to} & Ax \leq b. \end{array}$$

If $x^1, x^2, \ldots, x^K$ are the vertices of the feasible region, we say that x^k is an *optimal basic feasible solution* if $cx^k \geq cx^\ell$ for every ℓ. That is x^k is the vertex with the largest objective value among vertices. Note that the optimal basic feasible solution need not be unique.

An *optimal solution* is a feasible solution x such that $cx \geq cy$ for every other feasible solution y. As discussed in Section 3.1, in two dimensions, it is easy to see that an optimal basic feasible solution is also an optimal solution. In this section, we generalize this observation to polyhedra in higher-dimensional spaces.

3.3.1 Bounded Feasible Regions

We first consider linear programs where the feasible region is bounded.

Theorem 3.3.1. *If x^* is an optimal basic feasible solution of a linear program for which the feasible region is bounded, then it is an optimal solution.*

Proof: If x is a convex combination of y and z then $cx \leq \max\{cy, cz\}$. Similarly, if x is a convex combination of $x^1, \ldots, x^K$, then $cx \leq \max_{\ell \in \{1,\ldots,K\}} cx^\ell$. Any x is a convex combination of the vertices, and so cx must attain its largest value at a vertex. □

3.3.2 The General Case

The result also applies to unbounded polyhedra, though the associated analysis becomes more complicated.

Theorem 3.3.2. *If x^* is an optimal basic feasible solution of a linear program that has an optimal solution, then it is an optimal solution.*

Before proving the theorem, we will establish a helpful lemma. Recall that a line is a one dimensional affine space. A set $U \subseteq \Re^N$ is said to *contain a line* if there exists a vector $x \in S$ and a vector $d \neq 0$ such that $x + \alpha d \in S$ for all $\alpha \in \Re$.

Lemma 3.3.1. *Consider a polytope $U = \{x \in \Re^N | Ax \leq b\}$ that is not empty. Then the following statements are equivalent:*

1. *The polytope U has at least one vertex.*
2. *The polytope U does not contain a line.*

Proof of lemma: Suppose that U contains the line $\{x + \alpha d | \alpha \in \Re\}$ where $d \neq 0$. Then for all α, we have $A(x + \alpha d) \leq b$, or rearranging $\alpha Ad \leq b - Ax$. For this to be true for all α it must be that $Ad = 0$, so that A is not full rank, and so cannot have N linearly independent rows. Thus there are no vertices.

Suppose that U does not contain any line. We use a similar line of reasoning to Theorem 3.2.4. Let x be a point with the maximum number of linearly independent active constraints. Let the number of linearly independent active constraints be K. If $K = N$ then x is a vertex, and we are done.

Suppose that $K < N$, the consider the line $L = \{x + \alpha d | \alpha \in \Re\}$ for some d that is perpendicular to the constraints active at x. d must exist because the matrix whose rows are the constraints active at x is not full rank, and hence has a non-zero null space. All the points in L satisfy the K constraints at x. Because L cannot be contained in U, there must be some α for which an additional constraint is active. The point $x + \alpha d$ has $K + 1$ linearly independent active constraints contradicting the fact that K was the maximum attainable. □

Proof of theorem: Note that the fact that the linear program has a basic solution means that is can contain no lines. Let x be an optimal solution with the largest number of linearly independent active constraints, and the number of linearly independent active constraints at x be K.

If $K = N$ then x is a vertex satisfying the conclusion of the theorem. Suppose $K < N$, then take d orthogonal to the constraints active at x. The same reasoning as given in Lemma 3.3.1 shows that all points of the form $x + \alpha d$ have all the same constraints active as x, and also that for some α^*, an additional constraint is satisfied.

But, for sufficiently small $\alpha, x \pm \alpha d$ is still feasible. Because $c(x + \alpha d) = cx + \alpha cd$ can be no larger than cx, we have that $cd = 0$ and that $c(x+\alpha d) = cx$ for all α. But this means $x + \alpha^* d$ is an optimal solution with more than K linearly independent active constraints, contradicting the maximality of K. □

3.3.3 Searching through Basic Feasible Solutions

For any linear program, there are a finite number of basic feasible solutions, and one of them is optimal if the linear program has an optimum. This motivates a procedure for solving linear programs: enumerate all basic feasible solutions and select the one with the largest objective value. Unfortunately, such a procedure is not effective for large problems that arise in practice because there are usually far too many basic feasible solutions. As mentioned earlier, the number of basic solutions is the number of ways of choosing N linearly independent constraints from the entire collection of M constraints. There are $M!/(N!(M-N)!)$ choices. This is an enormous number – if $N = 20$ and $M = 100$, the number of choices $M!/(N!(M-N)!)$ exceeds half a billion trillion. Though many of these choices will not be linearly independent or feasible, the number of them that are basic feasible solutions is usually still enormous.

In Chapter 6, we will study the simplex method, which is a popular linear programming algorithm that searches through basic feasible solutions for an optimal one. It does not enumerate all possibilities but instead intelligently traverses a sequence of improving basic feasible solutions in a way that arrives quickly at an optimum. We will also study interior point methods, another very efficient approach that employs a different strategy. Instead of considering basic feasible solutions, interior point methods generate a sequence of improving solutions in the interior of the feasible region, converging on an optimum.

3.4 Greater-Than and Equality Constraints

We have focused until now on less-than constraints, each taking the form $ax \leq b$ for some $a \in \Re^N$ and $b \in \Re$. Two other forms of constraints commonly used to describe polyhedra are greater-than and equality constraints. Greater-than constraints take the form $ax \geq b$, while equality constraints take the form $ax = b$.

Both greater-than and equality constraints can be replaced by equivalent less-than constraints. A greater than constraint $ax \geq b$ is equivalent to $-ax \leq -b$, whereas an equality constraint $ax = b$ is equivalent to a pair of less-than constraints: $ax \leq b$ and $-ax \geq -b$. Hence, any set of constraints that includes less-than, greater-than, and equality constraints is equivalent to a set of less-than constraints and therefore represents a polytope.

In matrix notation, we can define a polytope involving all types of constraints by

$$S = \{x \in \Re^N | A^1 x \leq b, A^2 x \geq b^2, A^3 x = b^3\}.$$

This is the same polytope as one characterized by $Ax \leq b$, where

$$A = \begin{bmatrix} A^1 \\ -A^2 \\ A^3 \\ -A^3 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} b^1 \\ -b^2 \\ b^3 \\ -b^3 \end{bmatrix}.$$

The vertices of a polytope do not change if we change the way it is represented. This is because the notion of a vertex is geometric; that is, it only depends on which vectors are inside or outside the set. Our definition of *basic feasible solutions*, on the other hand, does rely on the algebraic representation. In particular, our definition determines whether a solution is basic depending on which of the M constraints represented by $Ax \leq b$ are active.

Theorem 3.2.1 establishes that vertices and basic feasible solutions are equivalent. Hence, the theorem relates a geometric, representation-independent concept to an algebraic, representation-dependent one. It is convenient to extend the definition of a basic feasible solution to situations where the representation of a polytope makes use of greater-than and equality constraints. This extended definition should maintain the equivalence between basic feasible solutions and vertices.

Let us now provide the generalized definition of basic and basic feasible solutions. Given a set of equality and inequality constraints defining a polytope $S \in \Re^N$, we say that a vector $\overline{x} \in \Re^N$ is a *basic solution* if all equality constraints are active, and among all constraints that are active at $\overline{x}$, N of

them are linearly independent. A *basic feasible solution* is a basic solution that satisfies all constraints.

To see that basic feasible solutions still correspond to vertices, consider a polytope

$$S = \{x \in \Re^N | A^1 x \leq b^1, A^2 x \geq b^2, A^3 x = b^3\}.$$

Recall that $S = \{x \in \Re^N | Ax \leq b\}$, where

$$A = \begin{bmatrix} A^1 \\ -A^2 \\ A^3 \\ -A^3 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} b^1 \\ -b^2 \\ b^3 \\ -b^3 \end{bmatrix}.$$

From Theorem 3.2.1, we know that basic feasible solutions of $Ax \leq b$ are equivalent to vertices of S. We will show that basic feasible solutions of the inequalities $A^1 x \leq b, A^2 x \geq b^2, A^3 x = b^3$ are equivalent to basic feasible solutions of $Ax \leq b$.

Consider a basic feasible solution $\overline{x}$ of $A^1 x \leq b^1, A^2 x \geq b^2, A^3 x = b^3$. All the equality constraints must be active at $\overline{x}$, and there must be a set of N linearly independent constraints that are active at $\overline{x}$. Let $\mathcal{I}$ denote this set of N linearly independent constraints. Let N_1, N_2, and N_3, be the number of less-than, greater-than, and equality constraints represented by $\mathcal{I}$. Hence, $N_1 + N_2 + N_3 = N$. Now consider the inequality $Ax \leq b$, which we write in a partitioned form

$$\begin{bmatrix} A^1 \\ -A^2 \\ A^3 \\ -A^3 \end{bmatrix} x \leq \begin{bmatrix} b^1 \\ -b^2 \\ b^3 \\ -b^3 \end{bmatrix}.$$

Since $A^3 \overline{x} = b^3$, all rows of A associated with A^3 and $-A^3$ correspond to active constraints. Only $2N_3$ of these correspond to equality constraints in $\mathcal{I}$, and among these $2N_3$ constraints, only only N_3 of them can be linearly independent (since each row of $-A_3$ is linearly dependent on the corresponding row of A_3).

Another $N_1 + N_2$ constraints in $\mathcal{I}$ lead to linearly independent active constraints in rows of A associated with A^1 and $-A^2$. This makes for a total of N linearly independent constraints. Therefore, any basic feasible solution of $A^1 x \leq b^1, A^2 x \geq b^2, A^3 x = b^3$ is also a basic feasible solution of $Ax = b$ and therefore a vertex of S. The converse – that a basic feasible solution of $Ax = b$ is a basic feasible solution of $A^1 x \leq b^1, A^2 x \geq b^2, A^3 x = b^3$ – can be shown by reversing preceding steps. It follows that basic feasible solutions of $A^1 x \leq b^1, A^2 x \geq b^2, A^3 x = b^3$ are equivalent to vertices of S.

3.5 Production

We now shift gears to explore a few of the many application domains of linear programming. A prime application of linear programming is to the allocation of limited resources among production activities that can be carried out at a firm. In this context, linear programming is used to determine the degree to which the firm should carry out each activity, in the face of resource constraints. In this section, we discuss several types of production problems that can be modeled and solved as linear programs.

3.5.1 Single-Stage Production

In a single stage production problem, there is stock in M types of resources and N activities, each of which transforms resources into a type of product. The available stock in each ith resource is denoted by b_i, which is a component of a vector $b \in \Re^M$. The level to which activity j is carried out is a decision variable x_j, which a component of a vector $x \in \Re^N$. This quantity x_j represents the number of units of product type j generated by the activity.

In producing each unit of product j, A_{ij} units of each ith resource are consumed. This gives us a matrix $A \in \Re^{M \times N}$. The activity levels are constrained to be nonnegative ($x_j \geq 0$), and in aggregate, they cannot consume more resources than available ($Ax \leq b$). Each unit of product j generates a profit of c_j, which is a component of a vector $c \in \Re^N$. The objective is to maximize profit. This gives rise to a linear program:

$$\begin{array}{ll} \text{maximize} & cx \\ \text{subject to} & Ax \leq b \\ & x \geq 0. \end{array}$$

Let us revisit the petroleum production problem of Chapter 1, which is an example of a single-stage production problem.

Example 3.5.1. *Crude petroleum extracted from a well contains a complex mixture of component hydrocarbons, each with a different boiling point. A refinery separates these component hydrocarbons using a distillation column. The resulting components are then used to manufacture consumer products such as low, medium, and high octane gasoline, diesel fuel, aviation fuel, and heating oil.*

Suppose we are managing a company that manufactures N petroleum products and have to decide on the number of liters x_j, $j \in \{1, \ldots, n\}$ of each product to manufacture next month. We have M types of resources in the form of component hydrocarbons. A vector $b \in \Re^M$ represents the quantity in

liters of each ith resource to be available to us next month. Each petroleum product is manufactured through a separate activity. The jth activity consumes A_{ij} liters of the ith resource per unit of the jth product manufactured.

Our objective is to maximize next month's profit. Each jth product garners c_j dollars per liter. Hence, the activity levels that maximize profit solve the following linear program:

$$\begin{array}{ll} \text{maximize} & cx \\ \text{subject to} & Ax \leq b \\ & x \geq 0. \end{array}$$

There are typically many possible activities that can be carried out to produce a wide variety of products. In this event, the number of activities N may far exceed the number M of resource types. Is it advantageous to carry out so many activities in parallel? Remarkably, the following theorem establishes that it is not:

Theorem 3.5.1. *If a single-stage production problem with M resource types and N activities has an optimal solution, then there is an optimal solution that involves use of no more than M activities.*

This result follows from Theorem 3.3.2, as we will now explain. Consider a basic feasible solution $\overline{x} \in \Re^N$. At $\overline{x}$, there must be N linearly independent binding constraints. Up to M of these constraints can be associated with the M rows of A. If $N > M$, we need at least $N - M$ additional binding constraints – these must be nonnegativity constraints. It follows that at least $N - M$ components of $\overline{x}$ are equal to zero. In other words, there are at most M activities that are used at a basic feasible solution.

By Theorem 3.3.2, if there is an optimal solution to a linear program, there is one that is a basic feasible solution, provided that a basic feasible solution exists. Hence, there is an optimal solution that entails carrying out no more than M activities. This greatly simplifies implementation of an optimal production strategy. No matter how many activities are available, we will make use of no more than M of them.

Before closing this section, it is worth discussing how to deal with capacity constraints. In particular, production activities are often constrained not only by availability of raw materials, but also by the capacity of manufacturing facilities. In fact, in many practical production activities, capacity is the only relevant constraint. Conveniently, capacity can simply be treated as an additional resource consumed by manufacturing activities. We illustrate this point through a continuation to our petroleum production example.

Example 3.5.2. *Recall that our petroleum production problem led to a linear program*

$$\begin{array}{ll} \text{maximize} & cx \\ \text{subject to} & Ax \leq b \\ & x \geq 0. \end{array}$$

Given an optimal solution $x^* \in \Re^N$, each component x^*_j tells us the quantity in liters of the jth petroleum product that we should manufacture next month.

Suppose that we have two factories that support different manufacturing processes. Each of our N manufacturing activities requires capacity from one or both of the factories. In particular, the manufacturing of each liter of the jth product requires a fraction a^1_j of next month's capacity from factory 1 and a fraction a^2_j of next month's capacity from factory 2. Hence, we face capacity constraints:

$$a^1 x \leq 1 \quad \text{and} \quad a^2 x \leq 1.$$

Let

$$\overline{A} = \begin{bmatrix} A \\ a^1 \\ a^2 \end{bmatrix} \quad \text{and} \quad \overline{b} = \begin{bmatrix} b \\ 1 \\ 1 \end{bmatrix}.$$

A new linear program incorporates capacity constraints:

$$\begin{array}{ll} \text{maximize} & cx \\ \text{subject to} & \overline{A}x \leq b \\ & x \geq 0. \end{array}$$

Note that the capacity constraints play a role entirely analogous to the constraints imposed by limitations in stock of component hydrocarbons. The capacity at each factory is just a resource that leads to an additional constraint.

3.5.2 Multi-Stage Production

Manufacturing of sophisticated products typically entails multiple stages of production activity. For example, in manufacturing a computer, chips are fabricated, then they are connected on a printed circuit board, and finally, printed circuit boards, casing, and other components are assembled to create a finished product. In such a process, not all activities deliver finished products. Some activities generate materials that serve as resources for other activities. Multi-stage production activities of this sort can still be formulated as linear programs. Let us illustrate this with an example.

Example 3.5.3. *Consider a computer manufacturer with two CPU chip fabrication facilities and one computer assembly plant. Components such as keyboards, monitors, casing, mice, disk drives, and other chips such as SRAM*

and DRAM are purchased from other companies. There are three grades of CPU chips manufactured by the company, and they are used to produce three models of computers. Fabrication facility 1 can produce chips of grades 1 and 2, while fabrication facility 2 can produce chips of grade 2 and 3. Completed chips are transported to the assembly plant where they are combined with other components to produce finished products. The only relevant constraints on manufacturing are capacities of the two fabrication facilities and the manufacturing plant.

Consider the decision of how to allocate resources and conduct manufacturing activities over the next month of operation. To formulate the problem as a linear program, we introduce the following decision variables:

x_1	*quantity of model 1 produced*
x_2	*quantity of model 2 produced*
x_3	*quantity of model 3 produced*
x_4	*quantity of grade 1 chip produced at factory 1*
x_5	*quantity of grade 2 chip produced at factory 1*
x_6	*quantity of grade 2 chip produced at factory 2*
x_7	*quantity of grade 3 chip produced at factory 2*

There are 6 *types of resources to consider:*
(1) *capacity at fabrication facility 1;*
(2) *capacity at fabrication facility 2;*
(3) *capacity at the assembly plant;*
(4) *grade 1 chips;*
(5) *grade 2 chips;*
(6) *grade 3 chips.*
Let the quantities available next month be denoted by a vector $b \in \Re^6$.

For each j*th activity and* i*th resource, let* A_{ij} *denote the amount of the* i*th resource consumed or produced per unit of activity* j*. If* $A_{ij} > 0$*, this represents an amount consumed. If* $A_{ij} < 0$*, this represents an amount produced. Consider, as an example, the constraint associated with capacity at the assembly plant:*

$$A_{31}x_1 + A_{32}x_2 + A_{33}x_3 + A_{34}x_4 + A_{35}x_5 + A_{36}x_6 + A_{37}x_7 \leq b_3.$$

The coefficients A_{34}*,* A_{35}*,* A_{36}*, and* A_{37} *are all equal to zero, since capacity at the assembly plant is not used in manufacturing chips. On the other hand, producing the three models of computers does require assembly, so* A_{31}*,* A_{32}*, and* A_{33} *are positive. Let us now consider, as a second example, the constraint associated with stock in grade 2 chips, which is an endogenously*

produced resource:

$$A_{51}x_1 + A_{52}x_2 + A_{53}x_3 + A_{54}x_4 + A_{55}x_5 + A_{56}x_6 + A_{57}x_7 \leq b_5.$$

The coefficients A_{51}, A_{52}, and A_{53} are zero or positive, depending on how many grade 2 chips are used in manufacturing units of each model. The coefficients A_{54} and A_{57} are zero, because manufacturing of grade 1 and grade 3 chips does not affect our stock in grade 2 chips. Finally, $A_{55} = -1$ and $A_{56} = -1$, because each unit of activities 5 and 6 involves production of one grade 2 chip. The value of b_5 represents the number of chips available in the absence of any manufacturing. These could be chips acquired from an exogenous source or left over from the previous month. If $b_5 = 0$, all chips used to produce computers next month must be manufactured during the month.

The profit per unit associated with each of the three models is given by a vector $c \in \Re^7$, for which only the first three components are nonzero. With an objective of maximizing profit, we have a linear program:

$$\begin{array}{ll} \text{maximize} & cx \\ \text{subject to} & Ax \leq b \\ & x \geq 0. \end{array}$$

Note that the basic difference between linear programs arising from multi-stage – as opposed to single-stage – production problems is that elements of the matrix A are no longer nonnegative. Negative elements are associated with the production of resources. There is another way to represent such linear programs that is worth considering. Instead of having a matrix A that can have negative elements, we could define matrices C and P such that both have only nonnegative elements and $A = C - P$. The matrix C represents quantities consumed by various activities, while P represents quantities produced. Then, the multi-stage production problem takes the form

$$\begin{array}{ll} \text{maximize} & cx \\ \text{subject to} & Cx - Px \leq b \\ & x \geq 0. \end{array}$$

As in the single-stage case, a basic feasible solution of the multi-stage production problem with M resource types – including materials produced for use as resources to later stages of production – must have M linearly independent binding constraints. If the number N of activities exceeds the number M of resources, at least $N - M$ activities must be inactive at a basic feasible solution. We therefore have an extension of Theorem 3.5.1:

Theorem 3.5.2. *If a multi-stage production problem with M resource types and N activities has an optimal solution, then there is an optimal solution that involves use of no more than M activities.*

3.5.3 Market Stratification and Price Discrimination

In all the production models we have considered, each unit of a product generated the same profit. Consider, for example, a single-stage production problem:

$$\begin{array}{ll} \text{maximize} & cx \\ \text{subject to} & Ax \leq b \\ & x \geq 0. \end{array}$$

Each unit of each jth product offers a profit of c_j.

In some situations, it is desirable to stratify a market and charge different prices for different classes of customers. For example, coupons can be directed at a certain segment of the market that will only purchase a product if it is below the advertised price. This allows a firm to sell at a high price to those who find it acceptable without loosing the profit it can obtain from the portion of the market that requires a lower price.

When there is a single price c_j associated with each jth product, the profit generated by manufacturing x_j units is $c_j x_j$. Suppose that the market is segmented and price discrimination is viable. In particular, suppose that we can sell up to K units each jth product at price c_j^1 and the rest at price $c_j^2 < c_j^1$. Then, the objective should be to maximize $\sum_{j=1}^N f_j(x_j)$, where

$$f_j(x_j) = \begin{cases} c_j^1 x_j, & \text{if } x_j \leq K \\ c_j^1 K + c_j^2 (x_j - K), & \text{otherwise.} \end{cases}$$

In fact, suppose that this were the case for every product. Then, the single-stage production problem becomes:

$$\begin{array}{ll} \text{maximize} & \sum_{j=1}^N f_j(x_j) \\ \text{subject to} & Ax \leq b \\ & x \geq 0. \end{array}$$

This optimization problem is not a linear program, but fortunately, it can be converted to one, as we now explain.

We introduce new decision variables $x^1, x^2 \in \Re^N$. For each j, let $x_j^1 = \min(x_j, K)$ and $x_j^2 = \max(x_j - K, 0)$. Hence, $x_j = x_j^1 + x_j^2$. Each x_j^1 represents the quantity of product j manufactured and sold for profit c_j^1, while x_j^2 represents the quantity manufactured and sold for profit c_j^2. A linear program

leads to optimal values for these new decision variables:

$$\begin{array}{ll} \text{maximize} & c^1x^1 + c^2x^2 \\ \text{subject to} & A(x^1 + x^2) \leq b \\ & x^1 \leq Ke \\ & x^1 \geq 0 \\ & x^2 \geq 0. \end{array}$$

Recall that e denotes the vector with every component equal to 1.

This idea generalizes to any number of market segments. If there are L different profit vectors $c^1 \geq c^2 \geq \cdots \geq c^L$, let $K_1, \ldots, K_{L-1}$ denote numbers of customers that will purchase each jth product at prices $c_j^1, \ldots, c_j^{L-1}$, respectively. Then, the profit-maximizing linear program is given by

$$\begin{array}{lll} \text{maximize} & \sum_{k=1}^{L} c^k x^k & \\ \text{subject to} & A\left(\sum_{k=1}^{L} x^k\right) \leq b & \\ & x^k \leq K_k e, & \text{for } k \in \{1, \ldots, L-1\} \\ & x^k \geq 0, & \text{for } k \in \{1, \ldots, L\}. \end{array}$$

There are LN decision variables in the above linear program, so at a basic feasible solution $\overline{x}^1, \ldots, \overline{x}^L$, there must be LN linearly independent active constraints. Among these, at most M can be associated with resource constraints, and at most $(L-1)N$ can be associated with the constraints $x^k \leq K_k e$. If $N > M$, this leaves at least $N - M$ additional constraints that must be active. These must be nonnegativity constraints. It follows that at least $N - M$ components among $\overline{x}^1, \ldots, \overline{x}^L$ are equal to zero. We therefore have the following theorem.

Theorem 3.5.3. *If a production problem with M resource types, N activities, and multiple market segments associated with different profits has an optimal solution, then there is an optimal solution that involves use of no more than $M + N(L-1)$ activities.*

3.6 Contingent Claims

In Section 2.4, we introduced the study of contingent claims. We showed how structured products can sometimes be replicated and priced and discussed the notion of arbitrage. In this section, we revisit the topic of contingent claims, bringing to bear our understanding of linear programming. This will allow us to broaden the range of situations where a bank can sell structured products while protecting itself against risk and to identify arbitrage opportunities, when they exist.

3.6.1 Hedging in an Incomplete Market

As in Section 2.4, consider a collection of N assets with M possible outcomes. Payoffs associated with each jth asset are represented by a vector $a^j \in \Re^M$. A payoff matrix

$$P = \left[\begin{array}{ccc} a^1 & \ldots & a^N \end{array} \right],$$

encodes payoff vectors for all assets. The price per unit of each asset is given by the corresponding component of a vector $\rho \in \Re^N$.

Consider outcome-dependent liabilities represented by a vector $b \in \Re^M$, and suppose we want to hedge the risk. One approach we have discussed is to acquire a portfolio $x \in \Re^N$ that replicates the liabilities, in the sense that $Px = b$. The future payoffs and liabilities cancel one another, so the only cost or revenue is the purchase price ρx.

It is not always possible to find a replicating portfolio. One exists only when b is in $\mathcal{C}(P)$. This is true for all b only if the market is complete – that is, if P has full row rank. Otherwise, the market is said to be *incomplete*, and some liability functions cannot be replicated.

How can we protect against risks when our liabilities cannot be replicated? One way involves *super-replicating* the liability function. A portfolio with holdings x is said to *super-replicate* b if $Px \geq b$. If we face liabilities b and purchase a super-replicating portfolio, our net payoff $(Px)_i - b_i$ for any possible future outcome i is nonnegative.

There are typically many super-replicating portfolios. Given the choice, it is natural to select the one that minimizes cost. The price of a portfolio x is ρx. To find the least-expensive super-replicating portfolio, we can solve a linear program:

$$\begin{array}{ll} \text{minimize} & \rho x \\ \text{subject to} & Px \geq b. \end{array}$$

The constraints ensure that the resulting portfolio super-replicates b. The following example illustrates super-replication in the context of hedging currency risk.

Example 3.6.1. (Hedging Currency Risk) *Reconsider Example 2.4.2, but without an assumption that liabilities can be replicated by a portfolio of contingent claims available in the market. Instead, let us consider super-replicating liabilities using the following securities: the yuan, a zero-coupon bond with face value* \$1*, maturing in three months, and a European call option with strike price* \$0.1*, expiring in three months. Index these three securities by* 1*,* 2*, and* 3*.*

Assume that in three months, the dollar price of the yuan will be a multiple of \$0.01 *that is at most* \$0.3*. Hence, there are* 30 *possible price outcomes. We*

will index them by $1, \ldots, 30$ *in ascending order. Based on the data provided in Example 2.4.2, each ith component of the vector* $b \in \Re^{30}$ *of liabilities is given by* $b_i = \min(9 - 0.7i, 5 - 0.5i, 2)$. *The payoff matrix* $P \in \Re^{30\times 3}$ *is given by*

$$P_{ij} = \begin{cases} 0.01i & \text{if } j = 1, \\ 1 & \text{if } j = 2, \\ \max(0, 0.01i - 0.1) & \text{if } j = 3. \end{cases}$$

The first, second, and third columns provide payoffs of the yuan, the bond, and the call option, respectively.

Suppose the current price of the yuan, the bond, and the call option are \$0.15, \$0.99, *and* \$0.07, *respectively. There are many portfolios that super-replicate b. For example, a particularly simple one consists only of* 2 *million bonds. As illustrated in Figure 3.4(a), for any outcome, this portfolio generates a payoff of* \$2 *million, which dominates the liability function. This portfolio costs* \$2.98 *million.*

We can solve the above linear program to identify a minimum-cost super-replicating portfolio. Doing so leads to a portfolio consisting of 2 *million bonds and a short position in about* 70 *million call options. Instead of incurring cost, acquiring this portfolio yields about* \$2.92 *million in revenue. The associated payoff function is illustrated in Figure Figure 3.4(b). By acquiring this portfolio, the manufacturer locks in about* \$2.92 *million without risk of losing money at the end of the three month period. There is, however, some chance that the manufacturer will receive additional revenue then, if the price of the yuan falls between* \$0.06 *and* \$0.2.

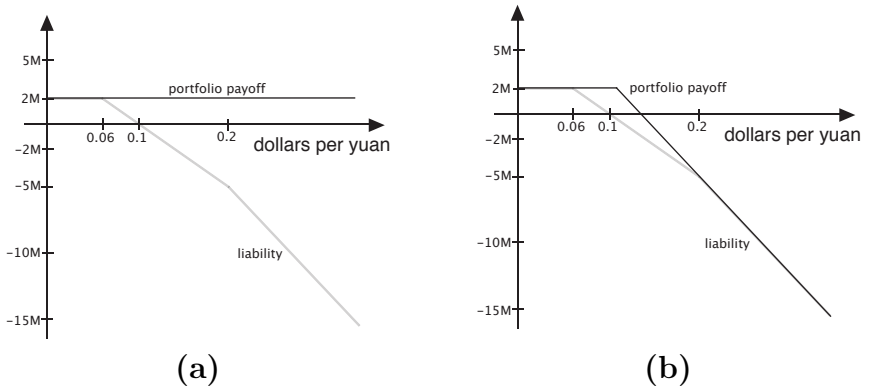


Figure 3.4: Payoff functions of two super-replicating portfolios: (a) one consisting only of bonds, and (b) one the consists of bonds and call options, which minimizes the cost of super-replication.

By Theorem 3.3.2, if the linear program

$$\begin{array}{ll} \text{minimize} & \rho x \\ \text{subject to} & Px \geq b. \end{array}$$

has an optimal solution and there exists a basic feasible solution, then there is a basic feasible solution that is an optimal solution. Let x^* be an optimal basic feasible solution. Then, there must be at least N constraints active at x^*. In other words, at least N components of Px^* are equal to the corresponding components of b. This means that if we plot liabilities b in the same figure as the payoff vector of a minimum-cost super-replicating portfolio, the two graphs will touch at no fewer than N points.

3.6.2 Finding Arbitrage Opportunities

In Section 2.4, we introduced the notion of arbitrage. An arbitrage opportunity was defined to be a vector $x \in \Re^N$ of portfolio holdings with a negative cost $\rho x < 0$ and nonnegative payoffs $Px \geq 0$. By purchasing assets in quantities given by the portfolio weights, we receive an amount $-\rho x > 0$, and in every possible future event, we are not committed to pay any money. Sounds good. But why stop at buying the quantities identified by x? Why not buy quantities $100x$ and garner an income of $-100\rho x$? Indeed, an arbitrage opportunity offers the possibility of making unbounded sums of money.

Using linear programming, we can identify arbitrage opportunities, if they exist. Consider solving the following linear program:

$$\begin{array}{ll} \text{minimize} & \rho x \\ \text{subject to} & Px \geq 0. \end{array}$$

It is clear that this linear program will look for the most profitable arbitrage opportunity. However, because there is no bound to the sum of money that can be made when an arbitrage opportunity presents itself, this linear program will have an unbounded solution when an arbitrage opportunity exists.

In order to generate a more meaningful solution, consider an alternative linear program:

$$\begin{array}{ll} \text{minimize} & \rho x \\ \text{subject to} & Px \geq 0 \\ & \rho x = -1. \end{array}$$

This one identifies a portfolio that offers an initial income of \$1. If an arbitrage opportunity exists, we can use it to make any amount of money, so we could use it to generate a \$1 income. Hence, existence of an arbitrage opportunity implies that the feasible region of this linear program is nonempty.

Once this linear program identifies a portfolio that generates \$1, we can make an arbitrarily large amount of money by purchasing multiples of this portfolio. Note that the objective function here could be anything – any feasible solution is an arbitrage opportunity.

There are other linear programs we could consider in the search for arbitrage opportunities. An example is:

$$\begin{array}{ll} \text{minimize} & e(x^+ + x^-) \\ \text{subject to} & P(x^+ - x^-) \geq 0 \\ & \rho(x^+ - x^-) = -1 \\ & x^+ \geq 0 \\ & x^- \geq 0. \end{array}$$

This linear program involves two decision vectors: $x^+, x^- \in \Re^N$. The idea is to view the difference $x^+ - x^-$ as a vector of portfolio weights. Note that if (x^+, x^-) is a feasible solution to our new linear program, the portfolio $x^+ - x^-$ is an arbitrage opportunity, since $P(x^+ - x^-) \geq 0$ and $\rho(x^+ - x^-) = -1$. Further, if there exists an arbitrage opportunity x that offers a \$1 profit, there is a feasible solution (x^+, x^-) such that $x = x^+ - x^-$. Hence, our new linear program finds the \$1 arbitrage opportunity that minimizes $e(x^+ + x^-)$, which is the number of shares that must be traded in order to execute the opportunity. This objective is motivated by a notion that if there are multiple arbitrage opportunities, one that minimizes trading activity may be preferable.

3.7 Incentive Schemes

Effective management requires incentivizing productive behavior. This calls for incentives that encourage individuals to act in the interest of the organization and to report truthfully. In this section, we explore the use of linear programs to design such incentives.

Within a firm, a manager offers incentives to those that report to him. When procuring services, a customer offers incentives to a contractor through negotiated terms. Government policy makers introduce incentives for firms and citizens through legislation. Independent of context, we will use the terms *principal* and *agent* to refer to the entities that offer and receive incentives.

3.7.1 Moral Hazard

When interests of the principal and the agent are not aligned, the agent may not act in a way that best serves the principal. This problem is referred to as *moral hazard*. One example arises with a contractor who bills by the hour and may consequently choose to work slowly. Another case is the hedge fund manager who may take excessive risks if he has a significant stake in profits but not losses.

We now formulate a linear program that identifies incentives to alleviate moral hazard. Suppose the agent selects from M actions, indexed by $\{1, \ldots, M\}$, and that his work leads to one of N possible outcomes, indexed by $\{1, \ldots, N\}$. If the agent selects action i, he incurs a personal cost c_i. Let $c \in \Re^M$ be the vector of personal costs. The agent requires his expected payment less his personal cost to be at least v in order to participate in the relationship. The principal can not monitor the agent's action, but only observes the outcome, which is random but influenced by the action. Let P_{ij} denote the probability of outcome j given that the agent selects action i, with $P \in \Re^{M \times N}$ being the matrix of these probabilities.

Suppose that the principal would like to incentivize the agent to select a particular action k by offering to pay an amount that depends on the outcome. In particular, for each outcome j, the principal will choose a payment amount x_j. His goal is to select a vector $x \in \Re^N$ of payment amounts that minimizes his expected payment while incentivizing action k. This is accomplished by solving the following linear program:

$$\begin{array}{lll} \text{minimize} & P_{k*}x & \\ \text{subject to} & P_{k*}x - c_k \geq v & \\ & P_{k*}x - c_k \geq P_{i*}x - c_i & \text{for } i \in \{1, \ldots, M\} \\ & x \geq 0. & \end{array}$$

Note that for each action i, $P_{i*}x$ is the expected payment if the agent selects action i. The objective is to minimize the expected payment, which is the expected cost to the principal, assuming the agent selects action k. The first constraint ensures participation. The next set of M constraints ensure that the expected payment for action k is at least as great as that for any other action. Nonnegativitity constraints are imposed on the payments.

Example 3.7.1. (Trader's Option) *Suppose a portion of our capital will be managed over the next year by a trader. For simplicity, let us assume that the return he will generate over the course of the year must be positive or negative ten or twenty percent. The realized return is a random draw from these possible outcomes, with probabilities that depend on how the trader*

chooses to work. For simplicity, let us assume that the trader chooses from four modes, with resulting return probabilities tabulated below.

return	-20%	-10%	10%	20%
high effort	0.02	0.30	0.60	0.08
medium effort	0.03	0.35	0.55	0.07
low effort	0.04	0.4	0.5	0.06
high risk	0.5	0	0	0.5

We would like the trader to apply high effort. Let the trader's personal costs be $\$40,000$ *and* $\$20,000$ *for the high and medium effort levels, respectively, and zero otherwise. Suppose that retaining the trader requires an expected payment less personal cost of at least* $\$200,000$.

Suppose for now that the trader avoids the high risk option, which incurs a high level of risk with zero expected return, for ethical reasons. Then, one way to incentivize high effort is to offer the trader a "bonus" that depends on the realized return. For example, if we offer him $\$240,000$ *for a* 10% *return and* $\$1,200,000$ *for a* 20% *return, the expected payment for high, medium, and low effort are* $\$240,000$, $\$216,000$, *and* $\$192,000$. *Among these, the trader would choose the high level of effort. However, an unscrupulous trader may opt for the high risk option, which would lead to an expected payment of* $\$600,000$. *This undesirable behavior arises because the trader's bonus increases with the size of positive returns but does not decrease with the size of negative returns. This payment plan is reminiscent of a call option, which is why this situation is referred to as the "trader's option."*

To design an incentive scheme that discourages the high risk option, we can solve the above linear program with $N = 4$ *outcomes associated with levels of return,* $M = 4$ *alternative trader actions, a matrix* P *with probabilities taken from the table above, a participation threshold* $v = 200,000$, *and personal costs* $c_1 = 40,000$ *for high effort,* $c_2 = 20,000$ *for medium effort, and* $c_3 = c_4 = 0$. *Solving the linear program yields an optimal payment of* $\$400,000$ *for a* 10% *return and no payment otherwise. An intriguing feature of this incentive scheme is that there is no payment for a* 20% *return while the payment for a* 10% *return is high. This is optimal because when a* 20% *return is rewarded with a bonus, the event also indicates that the trader has probably selected the high risk option which we want to discourage.*

3.7.2 Adverse Selection

It is often important for the principal to receive honest views from the agent. For example, the agent may know more about his own capabilities than

the principal. If the agent is honest about his capabilities, this can help the principal decide how to use the agent or whether to employ him at all. However, for obvious reasons, the agent may misrepresent his assessment. This problem is referred to as *adverse selection*.

We now formulate a linear program that addresses this issue in a context where the principal must decide whether or not to employ an agent. Once again, we consider N possible outcomes, indexed by $\{1, \ldots, N\}$. Suppose that the agent can be classified as one of M possible types, indexed by $\{1, \ldots, M\}$. The principal does not know the agent's type, but assigns a probability π_i to each ith possibility. If an agent of type i is employed, outcome j is realized with probability P_{ij}. Let $P \in \Re^{M \times N}$ be the matrix of these probabilities. An agent of type i requires an expected payment of at least v_i for employment.

Suppose that the principal only wants to employ the agent if his type is in $\{1, \ldots, M_1\}$ for some $M_1 < M$, and to accomplish this he wants to incentivize truthful reporting in a way that minimizes his costs. In particular, for each outcome j, the principal will choose a payment amount x_j. With a goal to minimize the expected payment while incentivizing participation and truthful reporting, optimal incentives solve a linear program:

$$\begin{array}{lll} \text{minimize} & \sum_{i=1}^{M_1} \pi_i \sum_{j=1}^{N} P_{ij} x_j & \\ \text{subject to} & \sum_{j=1}^{N} P_{ij} x_j \geq v_i & \text{for } i \in \{1, \ldots, M_1\} \\ & \sum_{j=1}^{N} P_{ij} x_j \leq v_i & \text{for } i \in \{M_1 + 1, \ldots, M\} \\ & x \geq 0. & \end{array}$$

Note that for any type i, $\sum_{j=1}^{N} P_{ij} x_j$ is the expected payment in the event that the agent is of type i and reports that he is qualified. Hence, the objective is the expected cost to the principal, assuming that the agent reports his true type. The first set of M_1 constraints ensure that qualified agents participate. The next set of $M - M_1$ constraints rule out benefits to dishonest unqualified agents.

Example 3.7.2. (Contracting with a Deadline) *A developer is looking to hire a construction company that can complete a project in one year. He is currently considering one construction company and will hire it if there is a* 90% *chance that the company will meet the deadline. To keep things simple, suppose there are two types of contractors: qualified and unqualified. These types can deliver on time with probability* 0.9 *and* 0.5*, respectively. The developer thinks the company in question is equally likely to be of either type. Qualified contractors require an expected payment of* \$9 *million, while unqualified contractors require an expected payment of* \$7 *million.*

We can solve the above linear program to derive an effective incentive scheme. There are $N = 2$ outcomes corresponding to an on time and delayed completion. There are $M = 2$ types associated with qualified and unqualified contractors. The corresponding type probabilities are $\pi_1 = \pi_2 = 0.5$. The matrix of type-contingent outcome probabilities is

$$P = \begin{bmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{bmatrix}.$$

We have $v_1 = 9$ and $v_2 = 7$. Solving the linear program, we obtain an optimal incentive scheme with $x_1 = 10$ and $x_2 = 0$. The chance of delay is high for an unqualified contractor, and the lack of compensation in that event discourages him from lying to take the project.

3.8 Pattern Classification

Many engineering and managerial activities call for automated classification of observed data. Computer programs that classify observations are typically developed through *machine learning*. We discuss an example involving breast cancer diagnosis.

Example 3.8.1. (Breast Cancer Diagnosis) *Breast cancer is the second largest cause of cancer deaths among women. A breast cancer victim's chances for long-term survival are improved by early detection of the disease. The first sign of breast cancer is a lump in the breast. The majority of breast lumps are benign, however, so other means are required to diagnose breast cancer – that is, to distinguish malignant lumps from benign ones. One approach to diagnosing breast cancer involves extracting fluid from a lump and photographing cell nuclei through a microscope. Numerical measurements of the sizes and shapes of nuclei are recorded and used for diagnosis.*

An automated system for diagnosing breast cancer based on these numerical measurements has proven to be very effective. This system was developed through machine learning. In particular, a computer program processed a large collection of data samples, each of which was known to be associated with a malignant or benign lump. Through this process, the computer program "learned" patterns that distinguish malignant lumps from benign ones to produce a system for classifying subsequent samples.

Pattern classification can be thought of in terms of mapping a feature vector $a \in \Re^K$ to one of a finite set C of classes. Each feature vector is an encoding that represents an observation. For example, in breast cancer diagnosis, each feature vector represents measurements associated with cell nuclei

from a breast lump. In this example, there are two classes corresponding to malignant and benign lumps.

Machine learning involves processing a set of feature vectors $u^1, \ldots, u^L \in \Re^K$ labeled with known classes $z_1, \ldots, z_L \in C$ to produce a mapping from $\Re^K$ to C. This mapping is then used to classify additional feature vectors. Machine learning has been used to generate pattern classifiers in many application domains other than breast cancer diagnosis. Examples include:

1. automatic recognition of handwritten alphabetical characters;
2. detection of faults in manufacturing equipment based on sensor data;
3. automatic matching of finger prints;
4. automatic matching of mug shots with composite sketches;
5. prediction of outcomes in sports events;
6. detection of favorable investment opportunities based on stock market data;
7. automatic recognition of spoken phonemes.

In this section, we discuss an approach to machine learning and pattern classification that involves formulation and solution of a linear program. We will only consider the case of two classes. This comes at no loss of generality, since methods that address two-class problems can also handle larger numbers of classes. In particular, for a problem with n classes, we could generate n classifiers, each distinguishing one class from the others. The combination of these classifiers addresses the n-class problem.

3.8.1 Linear Separation of Data

We consider two classes with labels $C = \{1, -1\}$. Samples in class 1 are referred to as *positive*, while samples in class -1 are *negative*. To develop a classifier we start with positive samples $u^1, \ldots, u^K \in \Re^N$ and negative samples $v^1, \ldots, v^L \in \Re^N$. The positive and negative samples are said to be *linearly separable* if there exists a hyperplane in $\Re^N$ such that all positive samples are on one side of the hyperplane and all negative samples are on the other. Figure 3.5 illustrates for $N = 2$ cases of samples that are linearly separable and samples that are not.

Recall that each hyperplane in $\Re^N$ takes the form $\{y \in \Re^N | yx = \alpha\}$ for some $x \in \Re^N$ and $\alpha \in \Re$. In mathematical terms, the positive and negative

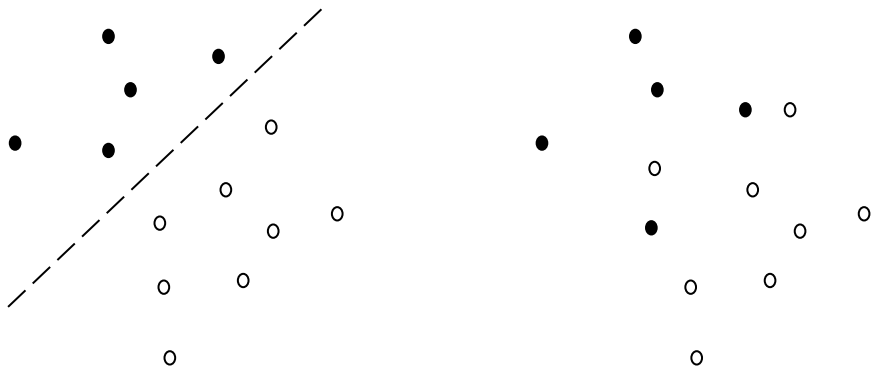


Figure 3.5: (a) Linearly separable data. The dashed line represents a separating hyperplane. (b) Data that are not linearly separable.

samples are linearly separable if there is a vector $x \in \Re^N$ and a scalar $\alpha \in \Re$ such that $u^k x > \alpha$ and $v^\ell x < \alpha$ for all $k \in \{1, \ldots, K\}$ and $\ell \in \{1, \ldots, L\}$.

If positive and negative samples are linearly separable, a hyperplane that separates the data provides a classifier. In particular, given parameters $x \in \Re^N$ and $\alpha \in \Re$ for such a hyperplane, we classify a sample $y \in \Re^N$ as positive if $yx > \alpha$ and as negative if $yx < \alpha$. But how can we find appropriate parameters x and α? Linear programming offers one approach.

To obtain a separating hyperplane, we need to compute $x \in \Re^N$ and $\alpha \in \Re$ such that $u^k x > \alpha$ and $v^\ell x < \alpha$ for all k and ℓ. This does not quite fit into the linear programming framework, because the inequalities are strict. However, the problem can be converted to a linear program, as we now explain. Suppose that we have parameters $\overline{x}$ and $\overline{\alpha}$ satisfying the desired strict inequalities. They then also satisfy $u^k \overline{x} - \overline{\alpha} > 0$ and $v^\ell \overline{x} - \overline{\alpha} < 0$ for all k and ℓ. For a sufficiently large scalar β, we have $\beta(u^k \overline{x} - \overline{\alpha}) \geq 1$ and $\beta(v^\ell \overline{x} - \overline{\alpha}) \leq -1$, for all k and ℓ. Letting $x = \beta \overline{x}$ and $\alpha = \beta \overline{\alpha}$, we have $u^k x - \alpha \geq 1$ and $v^\ell x - \alpha \leq -1$, for all k and ℓ. It follows that – if there is a separating hyperplane – there is a hyperplane characterized by parameters x and α satisfying $u^k x - \alpha \geq 1$ and $v^\ell x - \alpha \leq -1$, for all k and ℓ.

Let

$$U = \begin{bmatrix} u^1 \\ \vdots \\ u^K \end{bmatrix} \quad \text{and} \quad V = \begin{bmatrix} v^1 \\ \vdots \\ v^L \end{bmatrix},$$

so that our inequalities can be written more compactly as $Ux - \alpha e \geq e$ and $Vx - \alpha e \leq -e$, where e is the vector with every component equal to 1. A

linear program with constraints $Ux - \alpha e \geq e$ and $Vx - \alpha e \leq -e$ and any objective function will find a hyperplane that separates positive and negative values, if one exists.

3.8.2 Minimizing Violations

What should we do when the positive and negative samples cannot be separated by a hyperplane? One might aim at minimizing the number of misclassifications. A *misclassification* is either a positive sample u^k such that $u^k x < \alpha$ or a negative sample v^ℓ such that $v^\ell x > \alpha$. Unfortunately, the problem of minimizing the number of misclassifications is very hard. In fact, there are no known methods for efficiently finding a hyperplane that minimizes the number of misclassifications.

One alternative is to find a hyperplane that minimizes the "extent" of misclassifications. In particular, given a hyperplane parameterized by $x \in \Re^N$ and $\alpha \in \Re$, for each kth positive sample, define the *violation* $\delta_k^+ = \max(-Ux + \alpha e + e, 0)$. Similarly, for each ℓth negative sample, define the *violation* $\delta_\ell^- = \max(Vx - \alpha e + e, 0)$. We therefore have two vectors: $\delta^+ \in \Re^K$ and $\delta^- \in \Re^L$. The violation associated with a sample exceeds 1 if and only if the sample is misclassified. The following linear program introduces decision variables δ^+ and δ^-, in addition to x and α, and minimizes the sum of violations:

$$\begin{array}{ll} \text{minimize} & e\delta^+ + e\delta^- \\ \text{subject to} & \delta^+ \geq -Ux + \alpha e + e \\ & \delta^+ \geq 0 \\ & \delta^- \geq Vx - \alpha e + e \\ & \delta^- \geq 0. \end{array}$$

Figure 3.6 presents an example with $N = 2$ of a hyperplane produced by this linear program when positive and negative samples are not linearly separable.

3.9 Exercises

Question 1

Add a single inequality constraint to $x \leq 0, y \leq 0$ so that the feasible region contains only one point.

Question 2

How many vertices does the feasible set given by $x \geq 0, y \geq 0, z \geq 0, x + y + z = 1$ have. What common polytope is it? What is the maximum of

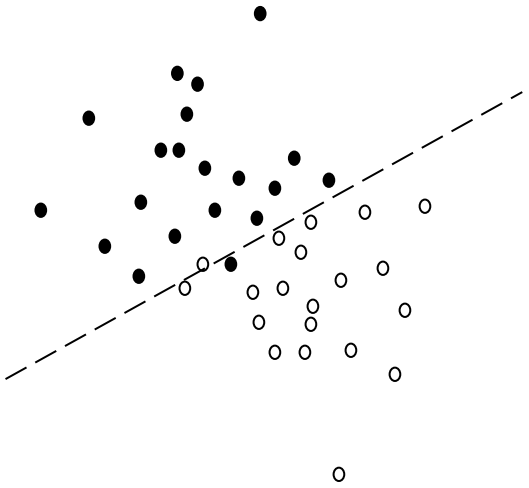


Figure 3.6: A hyperplane constructed by solving the linear program that minimizes violations. The resulting objective value was approximately 3.04.

$x + 2y + 3z$ over this polytope?

Question 3

Show that the feasible set constrained by $x \geq 0, y \geq 0, 2x + 5y \leq 3, -3x + 8y \leq -5$ is empty.

Question 4

Is $\Re^N$ convex? Show that it is, or explain why not.

Question 5

Draw a picture of a polytope in $\Re^2$ where one of the points in the polytope has 3 constraints active at the same time.

Question 6

In a particular polytope in $\Re^3$, one point has 3 constraints active at once. Does this point have to be a vertex? Why, or if not, give an example.

Question 7

Consider the following problem. Maximize $x + y$ subject to the constraints $x \geq 0, y \geq 0, -3x + 2y \leq -1, x - y \leq 2$. Is the feasible region bounded or unbounded? For a particular $L \geq 0$, find an x and y so that $[x\ y]^T$ is feasible, and also $x + y \geq L$. Note that x and y will depend on L.

Question 8

Dwight is a retiree who raises pigs for supplemental income. He is trying to decide what to feed his pigs, and is considering using a combination of feeds from some local suppliers. He would like to feed the pigs at minimum cost while making sure that each pig receives an adequate supply of calories and vitamins. The cost, calorie content, and vitamin supply of each feed is given in the table below.

Contents	Feed Type A	Feed Type B
Calories (per pound)	800	1000
Vitamins (per pound)	140 units	70 units
Cost (per pound)	\$0.40	\$0.80

Each pig requires at least 8000 calories per day and at least 700 units of vitamins. A further constraint is that no more than one-third of the diet (by weight) can consist of Feed Type A, since it contains an ingredient that is toxic if consumed in too large a quantity.

Formulate as a linear program. Explain your formulation, solve the problem, and provide the resulting daily cost per pig.

Question 9

The Apex Television Company has to decide on the number of 27- and 20-inch sets to be produced at one of its factories. Market research indicates that at most 40 of the 27-inch sets and 10 of the 20-inch sets can be sold per month. The maximum number of work-hours available is 500 per month. a 27-inch set requires 10 work hours, and a 20-inch set requires 10 work-hours. Each 27-inch set sold produces a profit of \$120 and each 20-inch set produces a profit of \$80. A wholesaler has agreed to purchase all the television sets (at the market price) produced if the numbers do not exceed the amounts indicated by the market research.

Formulate as a linear program. Explain your formulation, solve the problem, and provide the maximum profit.

Question 10

The Metalco Company desires to blend a new alloy of 40 percent tin, 35 percent zinc and 25 percent lead from several available alloys having the following properties.

Property	Alloy 1	Alloy 2	Alloy 3	Alloy 4	Alloy 5
Percentage of tin	60	25	45	20	50
Percentage of zinc	10	15	45	50	40
Percentage of lead	30	60	10	30	10
Cost ($ per lb)	22	20	25	24	27

The objective is to determine the proportions of these alloys that should be blended to produce the new allow at a minimum cost.

Formulate as a linear program. Explain your formulation and solve the problem. How many alloys are used in the optimal solution.

Question 11

Suppose that two constraints in a system are $cx \leq 1$ and $dx \leq 1$, where c and d are linearly *dependent*. A constraint is called *redundant* if removing it does not change the feasible region.

a) If $cd \geq 0$ does this mean that one of c or d is redundant? If so, explain why. If not, give an example

b) If $c \leq d$ does this mean that one of c or d is redundant? If so, explain why. If not, give an example

Question 12

A paper mill makes rolls of paper that are 80" (80 inches) wide. Rolls are marketed in widths of 14", 31" and 36". An 80" roll may be cut (like a loaf of bread) into any combination of widths whose sum does not exceed 80". Suppose there are orders for 216 rolls of width 14", 87 rolls of width 31" and 341 rolls of with 36". The problem is to minimize the total number of 80" rolls required to fill the orders.

There are six ways – called "cuts" – in which we might consider to cut each roll into widths of 14", 31" and 36". The number of 14", 31" and 36" rolls resulting from each cut are given in the following table:

cut	14"	31"	36"
1	5	0	0
2	3	1	0
3	1	2	0
4	3	0	1
5	0	1	1
6	0	0	2

(a) Let $x_1, x_2, x_3, x_4, x_5, x_6$ be decision variables, each representing the number of rolls cut in one of the six ways. Describe a linear program that determines the minimum number of 80" rolls required to fill the orders (ignore the requirement that each x_i should be an integer).
(b) Provide an optimal solution.
(c) Suppose the orders for next month are yet to be decided. Can we determine in advance how many types of cuts will be needed? Can we determine in advance any cuts that will or will not be used? For each question if your answer is affirmative, then explain why, and if not, explain why not.

Question 13

MSE Airlines (pronounced "messy") needs to hire customer service agents. Research on customer demands has lead to the following requirements on the minimum number of customer service agents that need to be on duty at various times in any given day:

Time Period	**Staff Required**
6am to 8am	68
8am to 10am	90
10am to noon	56
Noon to 2pm	107
2pm to 4pm	80
4pm to 6pm	93
6pm to 8pm	62
8pm to 10pm	56
10pm to midnight	40
Midnight to 6am	15

The head of personnel would like to determine least expensive way to meet these staffing requirements. Each agent works an 8 hour shift, but not all shifts are available. The following table gives the available shifts and daily wages for agents working various shifts:

Shift	**Daily Wages**
6am-2pm	$ 180
8am-4pm	$ 170
10am-6pm	$ 160
Noon-8pm	$ 190
2pm-10pm	$ 200
4pm-Midnight	$ 210
10pm-6am	$ 225
Midnight-8am	$ 210

(a) Formulate and describe a linear program that determines the least expensive way to meet staffing requirements.
(b) Provide an optimal solution.

Question 14

Consider the multi-stage production problem of producing chips and computers given in the lecture notes (Example 3.5.3). Suppose the net profits of selling one unit of Model 1, 2 and 3 are $600, $650 and $800 respectively. Production of one unit of Model 1 consumes 1.5 % of the capacity of the assembly plant. Model 2 consumes 2% and Model 3, 2.5%. Production of one unit of Chip 1 and 2 use 2% of the capacity of fabrication facility 1, each. Production of one unit of Chip 2 uses 3% of the capacity of fabrication facility 2. Production of Chip 3 uses 4%. Model 1 needs one unit of Chip 1 and one unit of Chip 2, Model 2 needs one unit of Chip 1 and one unit of Chip 3, and Model 3 needs one unit of Chip 2 and one unit of Chip 3. Initially there are no chips in stock.

Formulate and describe a linear program for solving this problem. What is the optimal production plan? How many activities are used? Is this the maximal number of activities that would be used for any basic feasible solution? Is this the minimal number of activities that would be used for any basic feasible solution?

Question 15

In the early 1900s, Edgar Anderson collected data on different species of iris' to study how species evolve to differentiate themselves in the course of evolution. In class, we studied three species – iris sesota, iris versicolor, and iris verginica – and used a linear program to show how iris sesota could be distinguished from the others based on sepal and petal dimensions. In particular, we showed that data on iris sesota was linearly separable from

the other species data. The Excel file used in class is available on the course web site.

Is iris versicolor data linearly separable from the other species' data? If so determine parameters (x and α) for a separating hyperplane. If not, determine parameters for a hyperplane that minimizes the violation metric discussed in class and used in the spreadsheet, and determine the number of misclassifications resulting from this hyperplane.

Question 16

Consider a stock that in one year can take on any price within $\{1, 2, \ldots, 99, 100\}$ dollars. Suppose that there are four European put options available on the stock, and these are the only assets that we can trade at the moment. Their strike prices and current market prices for buying/selling are provided in the following table:

Strike Price	Market Price
10	1.5
20	2.55
30	4.59
40	8.72

Is there an arbitrage opportunity involving trading of these four assets? If so, provide the associated portfolio?

Chapter 4

Duality

Given any linear program, there is another related linear program called the *dual*. In this chapter, we will develop an understanding of the dual linear program. This understanding translates to important insights about many optimization problems and algorithms. We begin in the next section by exploring the main concepts of duality through the simple graphical example of building cars and trucks that was introduced in Section 3.1.1. Then, we will develop the theory of duality in greater generality and explore more sophisticated applications.

4.1 A Graphical Example

Recall the linear program from Section 3.1.1, which determines the optimal numbers of cars and trucks to build in light of capacity constraints. There are two decision variables: the number of cars x_1 in thousands and the number of trucks x_2 in thousands. The linear program is given by

maximize	$3x_1 + 2.5x_2$	(profit in thousands of dollars)
subject to	$4.44x_1 \leq 100$	(car assembly capacity)
	$6.67x_2 \leq 100$	(truck assembly capacity)
	$4x_1 + 2.86x_2 \leq 100$	(metal stamping capacity)
	$3x_1 + 6x_2 \leq 100$	(engine assembly capacity)
	$x \geq 0$	(nonnegative production).

The optimal solution is given approximately by $x_1 = 20.4$ and $x_2 = 6.5$, generating a profit of about \$77.3 million. The constraints, feasible region, and optimal solution are illustrated in Figure 4.1.

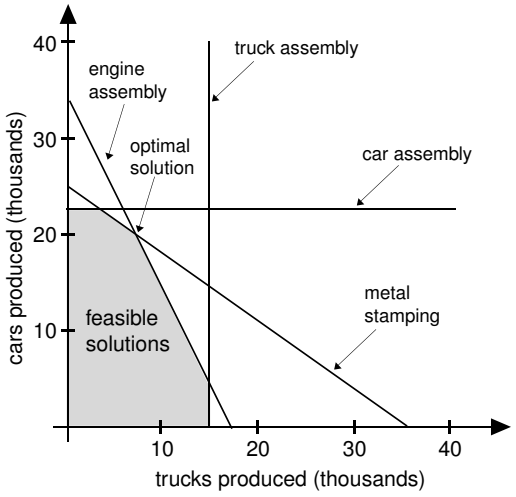


Figure 4.1: The constraints, feasible region, and optimal solution of the linear program associated with building cars and trucks.

Written in matrix notation, the linear program becomes

$$\begin{array}{ll} \text{maximize} & cx \\ \text{subject to} & Ax \leq b \\ & x \geq 0, \end{array}$$

where

$$c = \begin{bmatrix} 3 & 2.5 \end{bmatrix}, \qquad A = \begin{bmatrix} 4.44 & 0 \\ 0 & 6.67 \\ 4 & 2.86 \\ 3 & 6 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 100 \\ 100 \\ 100 \\ 100 \end{bmatrix}.$$

The optimal solution of our problem is a basic feasible solution. Since there are two decision variables, each basic feasible solution is characterized by a set of two linearly independent binding constraints. At the optimal solution, the two binding constraints are those associated with metal stamping and engine assembly capacity. Hence, the optimal solution is the unique solution to a pair of linear equations:

$$\begin{array}{ll} 4x_1 + 2.86x_2 = 100 & \text{(metal stamping capacity is binding)} \\ 3x_1 + 6x_2 = 100 & \text{(engine assembly capacity is binding).} \end{array}$$

In matrix form, these equations can be written as $\overline{A}x = \overline{b}$, where

$$\overline{A} = \begin{bmatrix} A_{3*} \\ A_{4*} \end{bmatrix} \quad \text{and} \quad \overline{b} = \begin{bmatrix} b_3 \\ b_4 \end{bmatrix}.$$

Note that the matrix $\overline{A}$ has full rank. Therefore, it has an inverse $\overline{A}^{-1}$. Through some calculations, we get (approximately)

$$\overline{A}^{-1} = \begin{bmatrix} 0.389 & -0.185 \\ -0.195 & 0.259 \end{bmatrix}.$$

The optimal solution of the linear program is given by $x = \overline{A}^{-1}b$, and therefore, the optimal profit is $c\overline{A}^{-1}\overline{b} = 77.3$.

4.1.1 Sensitivity Analysis

Suppose we wish to increase profit by expanding manufacturing capacities. In such a situation, it is useful to think of profit as a function of a vector $\Delta \in \Re^4$ of changes to capacity. We denote this profit by $z(\Delta)$, defined to be the maximal objective value associated with the linear program

$$\begin{array}{ll} \text{maximize} & cx \\ \text{subject to} & Ax \leq b + \Delta \\ & x \geq 0. \end{array} \tag{4.1}$$

Hence, the maximal profit in our original linear program is equal to $z(0)$. In this section, we will examine how incremental changes in capacities influence the optimal profit $z(\Delta)$. The study of such changes is called *sensitivity analysis*.

Consider a situation where the metal stamping and engine assembly capacity constraints are binding at the optimal solution to the linear program (4.1). Then, this optimal solution must be given by $x = \overline{A}^{-1}(\overline{b} + \Delta)$, and the optimal profit must be $z(\Delta) = c\overline{A}^{-1}(\overline{b} + \overline{\Delta})$, where

$$\overline{\Delta} = \begin{bmatrix} \Delta_3 \\ \Delta_4 \end{bmatrix}.$$

Furthermore, the difference in profit is $z(\Delta) - z(0) = c\overline{A}^{-1}\overline{\Delta}$.

This matrix equation provides a way to gauge the impact of changes in capacities on optimal profit in the event that the set of binding constraints does not change. It turns out that this also gives us the information required to conduct sensitivity analysis. This is because small changes in capacities will not change which constraints are binding. To understand why, consider the illustration in Figure 4.2, where the engine assembly capacity is increased by a small amount. Clearly, the new optimal solution is still at the intersection where metal stamping and engine assembly capacity constraints are binding. Similarly, though not illustrated in the figure, one can easily see that

incremental changes in any of the other capacity constraints will not change the fact that metal stamping and engine assembly capacity constraints are binding.

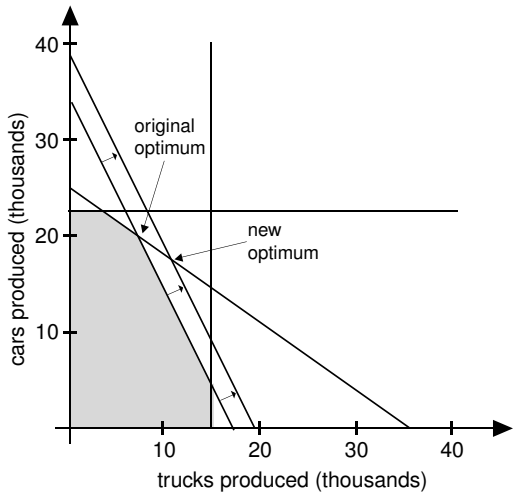


Figure 4.2: Changes in the optimal solution brought about by a small increase in capacity for engine assembly.

This observation does not hold when we consider large changes. As illustrated in Figure 4.3, sufficiently large changes can result in a different set of binding constraints. The figure shows how after a large increase in engine assembly capacity, the associated constraint is no longer binding. Instead, the truck assembly capacity constraint becomes binding.

The *sensitivity* y_i of profit to quantity of the ith resource is the rate at which $z(\Delta)$ increases as Δ_i increases, starting from $\Delta_i = 0$. It is clear that small changes in non binding capacities do not influence profit. Hence, $y_1 = y_2 = 0$. From the preceding discussion, we have $z(\Delta) - z(0) = c\overline{A}^{-1}\overline{\Delta}$, and therefore

$$\left[\begin{array}{cc} y_3 & y_4 \end{array}\right] = c\overline{A}^{-1} = \left[\begin{array}{cc} 3 & 2.5 \end{array}\right] \left[\begin{array}{cc} 0.389 & -0.185 \\ -0.195 & 0.259 \end{array}\right] = \left[\begin{array}{cc} 0.681 & 0.092 \end{array}\right].$$

In other words, the sensitivity is about \$0.681 million per percentage of metal stamping capacity and \$0.092 million per percentage of engine assembly capacity. If a 1% increase in metal stamping capacity requires the same investment as a 1% increase in engine assembly, we should invest in metal stamping.

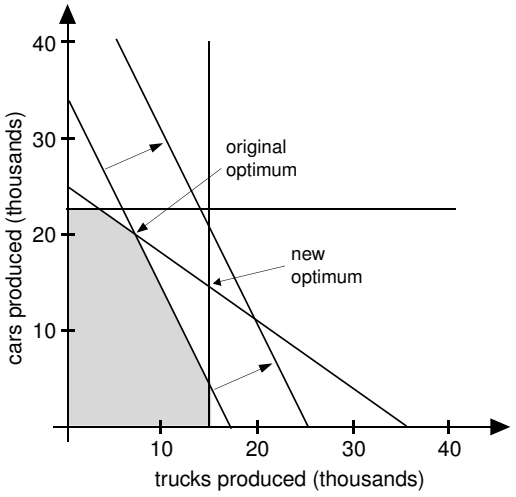


Figure 4.3: Changes in the optimal solution brought about by a large increase in capacity for engine assembly.

4.1.2 Shadow Prices and Valuation of the Firm

The sensitivities of profit to resource quantities are commonly called *shadow prices*. Each ith resource has a shadow price y_i. In our example of building cars and trucks, shadow prices for car and truck assembly capacity are zero. Shadow prices of engine assembly and metal stamping capacity, on the other hand, are \$0.092 and \$0.681 million per percent. Based on the discussion in the previous section, if the metal stamping and engine assembly capacity constraints remain binding when resource quantities are set at $b + \Delta$, the optimal profit is given by $z(\Delta) = z(0) + y\Delta$.

A shadow price represents the maximal price at which we should be willing to buy additional units of a resource. It also represents the minimal price at which we should be willing to sell units of the resource. A shadow price might therefore be thought of as the value per unit of a resource. Interestingly, if we compute the value of our entire stock of resources based on shadow prices, we get our optimal profit. For instance, in our example of building cars and trucks, we have

$$0.092 \times 100 + 0.681 \times 100 = 77.3.$$

As we will now explain, this is not just a coincidence but reflects a fundamental property of shadow prices.

From the discussion above we know that as long as the metal stamping and engine assembly constraints are binding, that $z(\Delta) = z(0) + y\Delta$. If we let $\Delta = -b$, then the resulting linear program has 0 capacity at each plant,

so the optimal solution is 0, with associated profit of 0. Moreover, both the metal stamping and engine assembly constraints are still binding. This means that $0 = z(-b) = z(0) + y(-b)$. Rearranging this indicates that $z(0) = yb$. This is a fundamental result: the net value of our current resources, valued at their shadow prices, is equal to the maximal profit that we can obtain through operation of the firm.

4.1.3 The Dual Linear Program

Shadow prices solve another linear program, called the *dual*. In order to distinguish it from the dual, the original linear program of interest – in this case, the one involving decisions on quantities of cars and trucks to build in order to maximize profit – is called the *primal*. We now formulate the dual.

To understand the dual, consider a situation where we are managing the firm but do not know how to solve linear programs. Therefore, we do not know exactly what the optimal decisions or optimal profit are. Company X approaches us and expresses a desire to purchase capacity at our factories. We enter into a negotiation over the prices $y \in \Re^4$ that we should charge per percentage of capacity at each of our four factories.

To have any chance of interesting us, the prices must be nonnegative: $y \geq 0$. We also argue that there are fixed bundles of capacity that we can use to manufacture profitable products, and the prices y must be such that selling such a bundle would generate at least as much money as manufacturing the product. In other words, we impose requirements that

$$4.44y_1 + 4y_3 + 3y_4 \geq 3 \quad \text{and} \quad 6.67y_2 + 2.86y_3 + 6y_4 \geq 2.5.$$

The first constraint ensures that selling a bundle of capacity that could be used to produce a car is at least as profitable as producing the car. The second constraint is the analog associated with production of trucks.

Given our requirements, Company X solves a linear program to determine prices that minimize the amount it would have to pay to purchase all of our capacity:

minimize	$100y_1 + 100y_2 + 100y_3 + 100y_4$	(cost of capacity)
subject to	$4.44y_1 + 4y_3 + 3y_4 \geq 3$	(car production)
	$6.67y_2 + 2.86y_3 + 6y_4 \geq 2.5$	(truck production)
	$y \geq 0$	(nonnegative prices).

In matrix notation, we have

$$\begin{array}{ll} \text{minimize} & yb \\ \text{subject to} & yA \geq c \\ & y \geq 0. \end{array}$$

The optimal solution to this linear program is

$$y = \begin{bmatrix} 0 & 0 & 0.092 & 0.681 \end{bmatrix},$$

and the minimal value of the objective function is 77.3. Remarkably, we have recovered the shadow prices and the optimal profit!

It is not a coincidence that the minimal cost in the dual equals the optimal profit in the primal and that the optimal solution of the dual is the vector of shadow prices – these are fundamental relations between the primal and the dual. We offer an intuitive explanation now and a more in-depth analysis in the next section.

The constraints ensure that we receive at least as much money from selling as we would from manufacturing. Therefore, it seems clear that the minimal cost in the dual is at least as large as the maximal profit in the primal. This fact is known as *weak duality*. Another result, referred to as *strong duality*, asserts that the minimal cost in the dual *equals* the maximal profit in the primal. This is not obvious. It is motivated to some extent, though, by the fact that Company X is trying to get the best deal it can. It is natural to think that if Company X negotiates effectively, it should be able to acquire all our resources for an amount of money equal to that we would obtain as profit from manufacturing. This would imply strong duality.

Why, now, should an optimal solution to the dual provide shadow prices? To see this, consider changing the resource quantities by a small amount $\Delta \in \Re^4$. Then, the primal and dual become

$$\begin{array}{lll} \text{maximize} & cx & \\ \text{subject to} & Ax \leq b + \Delta & \text{and} \\ & x \geq 0 & \end{array} \qquad \begin{array}{lll} \text{minimize} & y(b + \Delta) \\ \text{subject to} & yA \geq c \\ & y \geq 0. \end{array}$$

The maximal profit in the primal and the minimal cost in the dual are both equal to $z(\Delta)$. Suppose that the optimal solution to the dual is unique – as is the case in our example of building cars and trucks. Then, for sufficiently small Δ, the optimal solution to the dual should not change, and therefore the optimal profit should change by $z(\Delta) - z(0) = y(b + \Delta) - yb = y\Delta$. It follows that the optimal solution to the dual is the vector of shadow prices.

4.2 Duality Theory

In this section, we develop weak and strong duality in general mathematical terms. This development involves intriguing geometric arguments. Developing intuition about the geometry of duality is often helpful in generating useful insights about optimization problem.

Duality theory applies to general linear programs, that can involve greater-than, less-than, and equality constraints. However, to keep things simple, we will only study in this section linear programs that are in *symmetric form*. Such linear programs take the form:

$$\begin{array}{ll} \text{maximize} & cx \\ \text{subject to} & Ax \leq b \\ & x \geq 0. \end{array}$$

for some matrix $A \in \Re^{M \times N}$ and vectors $b \in \Re^M$ and $c \in \Re^N$. The decision variables – called the *primal variables* – make up a vector $x \in \Re^N$. As we will discuss later in the chapter, general linear programs can be converted to symmetric form, so our development of duality theory in this context also applies to general linear programs.

The dual of a symmetric form linear program takes the form

$$\begin{array}{ll} \text{minimize} & yb \\ \text{subject to} & yA \geq c \\ & y \geq 0. \end{array}$$

The decision variables – called the *dual variables* – form a vector $y \in \Re^M$.

Note that each decision variable in the primal problem corresponds to a constraint in the dual problem, and each constraint in the primal problem corresponds to a variable in the dual problem.

4.2.1 Weak Duality

Suppose that x is a feasible solution of the primal and y is a feasible solution of the dual. Then, $Ax \leq b$, $yA \geq c$, $x \geq 0$, and $y \geq 0$. It follows that $yAx \geq cx$ and $yAx \leq yb$. Hence, $cx \leq by$. This is the weak duality theorem, which we state below:

Theorem 4.2.1. (Weak Duality) *For any feasible solutions x and y to primal and dual linear programs, $cx \leq yb$.*

The following theorem states one immediate implication of weak duality.

Theorem 4.2.2. (Certificate of Optimality) *If x and y are feasible solutions of the primal and dual and $cx = yb$, then x and y must be optimal solutions to the primal and dual.*

There is another interesting consequence of weak duality that relates infiniteness of profit/cost in the primal/dual with feasibility of the dual/primal,

as we now explain. Let y be a feasible solution of the dual. By weak duality, we have $cx \leq yb$ for all feasible x. If the optimal profit in the primal is ∞, then $\infty \leq yb$. This is not possible, so the dual cannot have a feasible solution. The following theorem captures this fact, together with its converse, which can be established via a symmetric argument.

Theorem 4.2.3. (Infiniteness and Feasibility in Duality) *If the optimal profit in the primal is ∞, then the dual must be infeasible. If the optimal cost in the dual is $-\infty$, then the primal must be infeasible.*

4.2.2 Strong Duality

Theorem 4.2.2 asserts that if x and y are feasible solutions of the primal and dual and $cx = yb$, then x and y must be optimal solutions of the primal and dual. This does not imply that there are feasible solutions x and y such that $cx = yb$. However the strong duality guarantees this.

Theorem 4.2.4. (Strong Duality) *The dual has an optimal solution if and only if the primal does. If x^* and y^* are optimal solutions to the primal and dual, then $cx^* = y^*b$.*

Note that here, and throughout the book, when we refer to an optimal solution, it is implicitly assumed to be finite. If the optimal value can get arbitrarily large, we say the objective is unbounded. There are two slightly different sorts of unboundedness we discuss, subtly different. In the first, the feasible region is unbounded, and in this situation we say the *problem* is unbounded. In the second, the objective function can get arbitrarily large, and we say that the *objective value* is unbounded. Note that the second sort of unboundedness implies the first.

In order to prove the Strong Duality Theorem, we first have an aside, and discuss optimality of slightly more general functions.

4.2.3 First Order Necessary Conditions

It is possible to establish the Strong Duality Theorem directly, but the KKT conditions (given later in this section) are useful in their own right, and strong duality is an immediate consequence.

Before given the KKT conditions, we digress still more, and talk about convex sets and hyperplanes. Given two sets, U and V, we say that a hyperplane H *separates* U and V if all of U is on one side of the hyperplane, and all of V is on the other side. In other words, if H is given by $\{x|a^Tx = b\}$, then H separates $\Re^N$ into two sets, $H^+ = \{x|a^Tx \geq b\}$ and H^- $\{x|a^Tx \leq b\}$.

H separates U and V if U is contained in H^+ and V is contained in H^- or vice versa.

Theorem 4.2.5. (Separating Hyperplane) *Let U and V be two disjoint convex sets. Then, there exists a hyperplane separating U and V.*

"Picture Proof:" Let $\delta = \inf_{u \in U, v \in V} \|u - v\|$ We will demonstrate the result only for the case where $\delta > 0$ and there is a $u \in U$ and $v \in V$ with $\|u - v\| = \delta$. This case is all that will be needed to cover all the applications we will use of the theorem, and the full result is beyond the scope of this book.

Take $u \in U$ and $v \in V$ with $\|u - v\| = \delta$. Let H be the hyperplane through v that is perpendicular to $u - v$. We claim that H is a separating hyperplane for U and V.

Suppose this were not the case. Then, without loss of generality, we can assume that $v = 0$ (we can translate every point by $-v$). The means that H will be given by $\{x | u^T x = 0\}$. Note that $0 \in H^-$. Suppose not all of U is in H^+. Then there would be some $v \in U$ with $u^T v < 0$. If $d = v - u$ and $\alpha = -\frac{u^T d}{d^T d} \in (0, 1)$, and let $w = u + \alpha d = \alpha v + (1 - \alpha)u$. w must be in U because it is a convex combination of things in U, and the length of w is

$$\begin{aligned} w^T w &= (u + \alpha d)^T (u + \alpha d) \\ &= u^T u + 2\alpha u^T d + \alpha^2 d^T d \\ &= u^T u + \alpha d^T d (2\frac{u^T d}{d^T d} + \alpha) \\ &< u^T u \end{aligned}$$

because $u^T d < 0$. This contradicts the fact that u was the point in U closest to the origin. Thus, all of U is in H^+. A similar argument shows that each point of V must lie in H^- or else the convexity of V would generate a point closer to u than 0, and so H is a separating hyperplane for U and V. □

As discussed in the above proof, we will only be needing a restricted form of the separating hyperplane here. In particular, the following result which follows from the fact that polyhedra are closed (they are the intersection of closed half spaces), and for any point x and any closed set P, there is a point in P that is closest to x.[1]

Corollary 4.2.1. ***Separating Hyperplane Theorem*** *If P is a polyhedron, and x is a point distinct from P, then there is a vector s such that $s^T x < s^T p$ for all $p \in P$.*

[1]Despite being obvious if drawn, this result is typically established using the fact that distance is a continuous function and applying Weierstrass' theorem.

A corollary of separating hyperplanes is Farkas' lemma.

Lemma 4.2.1. ***Farkas' Lemma*** *For any $A \in \Re^{M \times N}$ and $b \in \Re^M$, exactly one of the following two alternatives holds:*
(a) *There exists a vector $x \in \Re^N$ such that $x \geq 0$, $Ax = b$.*
(b) *There exists a vector $y \in \Re^M$ such that $yb < 0$ and $yA \geq 0$.*

Proof: If (a) and (b) are both true, then $0 > yb = yAx \geq 0$, which is a contradiction. This means that (a) being true makes (b) false.

Suppose that (a) is false, then b is not in the polyhedron $P = \{Ax | x \geq 0\}$[2]. Let y be the vector guaranteed by the separating hyperplane theorem. This means that $by < py$ for all $p \in P$. 0 is in P, so this means that $yb < 0$. Suppose $yA_{j*} < 0$ for some j. Then for some α we must have $y(\alpha A_{j*}) < yb$ violating the fact that αA_{j*} is in P. Thus $yA_{j*} \geq 0$ for each j, or $yA \geq 0$ so that (b) is true. □

We can now give the KKT conditions. Note that these conditions are necessary, but not sufficient for a maximizer. There is a slight technical condition on the maximizing point, it needs to be *regular*. A regular point is one where if all active constraints are linearized (that is, replaced with tangent hyperplanes), the set of feasible directions remains the same.[3]. This rules out some extremely rare coincidences of constraints, and note that in linear systems every point is regular.

That caveat aside, here are the KKT conditions.

Theorem 4.2.6. *Suppose that $f, g^1, g^2, \ldots g^M$ are differentiable functions from $\Re^N$ into $\Re$. Let x be the point that maximizes f subject to $g^i(x) \leq 0$ for each i, and assume the first k constraints are active and x is regular. Then there exists $y_1, \ldots, y_k \geq 0$ such that $\nabla f(x) = y_1 \nabla g^1(x) + \ldots + y_k \nabla g^k(x)$.*

Proof: Let $c^T = \nabla f(x)$. The directional derivative of f in the direction of unit vector z is cz. Thus, every feasible direction z must have $cz \leq 0$. Let $A \in \Re^{k \times N}$ be the matrix whose ith row is $(\nabla g^i(x))^T$. Then a direction z is feasible if $Az \leq 0$. Thus, there is no z with $Az \leq 0$ and $cz > 0$. From Farkas' lemma, we can now say that there is a y such that $y \geq 0$ and $yA = c$, which is the statement of the theorem. □

Note that the conditions describe ∇f as a combination of active constraint gradients only. Another way of stating the conditions is to say that $\nabla f = y_1 \nabla g^1(x) + \ldots + y_n \nabla g^n(x)$ where $y^i g^i(x) = 0$. Now the sum is over all constraints (not just the active ones), but the second condition says that the

[2]To see that P is a polyhedron, it can also be written as $\{z | z = Ax, x \geq 0\}$.

[3]If there is a feasible curve from x whose initial direction is d, then d is a feasible direction.

coefficients of non-active constraints must be 0. The condition $y^i g^i(x) = 0$ is called a complementarity condition, and is another certificate of optimality.

The Strong Duality Theorem is an application of the KKT conditions to the particular case where each of the functions being considered is linear.
Proof of strong duality: The primal problem is given by

$$\begin{array}{ll} \text{maximize} & cx \\ \text{subject to} & Ax \leq b \\ & x \geq 0. \end{array}$$

Letting $f(x) = cx$, $g^1(x) = (Ax - b)_1, \ldots, g^M(x) = (Ax - b)_M$, $g^{M+1}(x) = -x_1, \ldots, g^{M+N}(x) = -x_N$, we see that the primal is equivalent to

$$\begin{array}{ll} \text{maximize} & f(x) \\ \text{subject to} & g(x) \leq 0. \end{array}$$

Note that $\nabla g(x^*) = c^T$, $\nabla g^k(x) = (A_{k*})^T$ for $k = 1, \ldots, M$, and $\nabla g^k(x) = -e_k$ for $k = M + 1, \ldots, M + N$.

Suppose x^* is an optimal solution. The KKT conditions ensure existence $y \in \Re^M$ and $z \in \Re^N$ such that $y \geq 0$, $z \geq 0$, $c = yA - z$, and $y(Ax^* - b) - zx^* = 0$. It follows that $yA \geq c$ and $yb = (yA - z)x^* = cx$. The result follows. □

4.2.4 Complementary Slackness

Recall that if x^* and y^* are optimal solutions to primal and dual linear programs, each dual variable y_i^* can be viewed as the sensitivity of the objective value to the value of b_i. If the constraint $A_{i*}x \leq b_i$ is not binding, the objective value should not be sensitive to the value of b_i, and therefore, y_i^* should be equal to zero. The fact that this is true for every i can be expressed concisely in terms of an equation: $y^*(b - Ax^*) = 0$; since all components of both $b - Ax^*$ and y^* are nonnegative, the only way the inner product can be equal to 0 is if, for each ith component, either $A_{i*}x = b_i$ or $y_i = 0$. Similarly, since the primal is the dual of the dual, each x_j^* represents sensitivity of the objective value to c_j, and we have $(y^*A - c)x^* = 0$.

The preceding discussion suggests that, for any optimal solutions x^* and y^* to the primal and dual, $y^*(b - Ax^*) = 0$ and $(y^*A - c)x^* = 0$. Interestingly, in addition to this statement, the converse is true: for any feasible solutions x and y to the primal and dual, if $y(b - Ax) = 0$ and $(yA - c)x = 0$ then x and y are optimal solutions to the primal and dual. These facts are immediate consequences of duality theory. They are captured by the complementary slackness theorem, which we now state and prove.

Theorem 4.2.7. (Complementary Slackness) *Let x and y be feasible solutions to symmetric form primal and dual linear programs. Then, x and y are optimal solutions to the primal and dual if and only if $y(b - Ax) = 0$ and $(yA - c)x = 0$.*

Proof: Feasibility implies that $y(b-Ax) \geq 0$ and $(yA-c)x \geq 0$. Further, if x and y are optimal,

$$(yA - c)x + y(b - Ax) = yb - cx = 0,$$

by strong duality (Theorem 4.2.4). Hence, if x and y are optimal, $y(b-Ax) = 0$ and $(yA - c)x = 0$.

For the converse, suppose that x and y are feasible and that $y(b-Ax) = 0$ and $(yA - c)x = 0$. Then,

$$0 = (yA - c)x + y(b - Ax) = yb - cx,$$

which provides a certificate of optimality (Theorem 4.2.2). □

There are many interesting consequences to complementary slackness. We will consider in Section 4.5 one application involving allocation of a labor force among interdependent production processes.

4.3 Duals of General Linear Programs

For a linear program that isn't in the symmetric form we can still construct the dual problem. To do this, you can transform the linear program to symmetric form, and then construct the dual from that. Alternatively, you can apply the KKT conditions directly. Either approach results in an equivalent problem.

For example, suppose the linear program is minimize cx subject to $Ax \leq b, x \geq 0$. Then because minimizing cx is the same as maximizing $-cx = (-c)x$, the linear program is the same as maximize $(-c)x$ subject to $Ax \leq b, x \geq 0$.

The modifications needed are summarized below.

- If the objective function, cx is minimized rather than maximized, then replace c by $-c$.
- If a constraint is a greater than constraint, $ax \geq \beta$, then take the negative of the constraint to get $(-a)x \leq -\beta$.
- If a constraint is an equality constraint, $ax = \beta$, then treat it as a greater than constraint and a less than constraint to get $ax \leq \beta$ and $(-a)x \leq -\beta$.

- If x is not constrained to be positive, then replace x by x^+ and $x-$, there x^+ represents the positive part of x, and $x-$ represents the negative part of x, just like the arbitrage example of the previous chapter.

As an example, suppose the linear program is

$$\begin{array}{ll} \text{minimize} & cx \\ \text{subject to} & A^1x \leq b^1 \\ & A^2x \geq b^2 \\ & A^3x = b^3 \end{array}$$

Then rearranging into symmetric form would give

$$\begin{array}{ll} \text{maximize} & (-c)x^+ + cx^- \\ \text{subject to} & A^1x^+ - A^1x^- \leq b^1 \\ & (-A^2)x^+ + A^2x^- \leq -b^2 \\ & A^3x^+ - A^3x^- \leq b^3 \\ & (-A^3)x^+ + A^3x^- \leq -b^3 \\ x^+, x^- \geq 0 & \end{array}$$

Note that if x^+, x^- is a solution, then so is $x^+ + y, x^- + y$ for any $y \geq 0$. However, if $y \neq 0$, then this will not represent a vertex.

Taking the dual of the above linear program gives

$$\begin{array}{ll} \text{minimize} & y^1b^1 - y^2b^2 + y^3b^3 - y^4b^3 \\ \text{subject to} & y^1A^1 - y^2A^2 + y^3A^3 - y^4A^3 \geq -c \\ & -y^1A^1 + y^2A^2 - y^3A^3 + y^4A^3 \geq c \\ & y^1, y^2, y^3, y^4 \geq 0 \end{array}$$

Notice that y^3 and y^4 are exactly what they would have been if one had replaced an unconstrained y with $y^3 = y^+$ and $y^4 = y^-$. Thus, writing $y = y^3 - y^4$, we can rewrite the dual as

$$\begin{array}{ll} \text{minimize} & y^1b^1 - y^2b^2 + yb^3 \\ \text{subject to} & y^1A^1 - y^2A^2 + yA^3 \geq -c \\ & -y^1A^1 + y^2A^2 - yA^3 \geq c \\ & y^1, y^2 \geq 0 \end{array}$$

The fact that equality constraints in the primal correspond to unconstrained variables in the dual is but one aspect that can be observed by looking at the above dual. Other features are summarized in the table below, which describes how to take the dual of a general linear program.

PRIMAL	maximize	minimize	DUAL
	$\leq b_i$	≥ 0	
constraints	$\geq b_i$	≤ 0	*variables*
	$= b_i$	*unconstrained*	
	≥ 0	$\geq c_j$	
variables	≤ 0	$\leq c_j$	*constraints*
	unconstrained	$= c_j$	

Note, using the rules in the above table, the dual of

$$\begin{array}{ll} \text{minimize} & cx \\ \text{subject to} & A^1 x \leq b^1 \\ & A^2 x \geq b^2 \\ & A^3 x = b^3 \end{array}$$

becomes

$$\begin{array}{ll} \text{maximize} & y^1 b^1 + y^2 b^2 + y^3 b^3 \\ \text{subject to} & y^1 A^1 + y^2 A^2 + y^3 A^3 = c \\ & y^1 \leq 0 \\ & y^2 \geq 0 \end{array}$$

Since the dual of the dual is the primal, reorganizing the above table yields an alternative procedure for converting primals that involve minimization to their duals.

PRIMAL	minimize	maximize	DUAL
	$\leq b_i$	≤ 0	
constraints	$\geq b_i$	≥ 0	*variables*
	$= b_i$	*unconstrained*	
	≥ 0	$\leq c_j$	
variables	≤ 0	$\geq c_j$	*constraints*
	unconstrained	$= c_j$	

4.4 Two-Player Zero-Sum Games

In this section, we consider games in which each of two opponents selects a strategy and receives a payoff contingent on both his own and his opponent's selection. We restrict attention here to zero-sum games – those in which a payoff to one player is a loss to his opponent. Let us consider an example that illustrates the nature of such problems.

Example 4.4.1. (drug running) *A South American drug lord is trying to get as many of his shipments across the border as possible. He has a fleet of boats available to him, and each time he sends a boat, he can choose one of three ports at which to unload. He could choose to unload in San Diego, Los Angeles, or San Francisco.*

The USA Coastguard is trying to intercept as many of the drug shipments as possible but only has sufficient resources to cover one port at a time. Moreover, the chance of intercepting a drug shipment differs from port to port. A boat arriving at a port closer to South America will have more fuel with which to evade capture than one arriving farther away. The probabilities of interception are given by the following table:

Port	Probability of interception
San Diego	*1/3*
Los Angeles	*1/2*
San Francisco	*3/4*

The drug lord considers sending each boat to San Diego, but the coastguard realizing this would always choose to cover San Diego, and only $2/3$ *of his boats would get through. A better strategy would be to pick a port at random (each one picked with* $1/3$ *probability). Then, the coastguard should cover port 3, since this would maximize the number of shipments captured. In this scenario,* $3/4$ *of the shipments would get through, which is better than* $2/3$*. But is this the best strategy?*

Clearly, the drug lord should consider randomized strategies. But what should he optimize? We consider as an objective maximizing the probability that a ship gets through, assuming that the Coastguard knows the drug lord's choice of randomized strategy. We now formalize this solution concept for general two-person zero-sum games, of which our example is a special case.

Consider a game with two players: player 1 and player 2. Suppose there are N alternative decisions available to player 1 and M available to player 2. If player 1 selects decision $j \in \{1, \ldots, N\}$ and player 2 selects decision $i \in \{1, \ldots, M\}$, there is an expected payoff of P_{ij} to be awarded to player 1 at the expense of player 2. Player 1 wishes to maximize expected payoff, whereas player 2 wishes to minimize it. We represent expected payoffs for all possible decision pairs as a matrix $P \in \Re^{M \times N}$.

A randomized strategy is a vector of probabilities, each associated with a particular decision. Hence, a randomized strategy for player 1 is a vector $x \in \Re^N$ with $ex = 1$ and $x \geq 0$, while a randomized strategy for player 2 is a vector $y \in \Re^M$ with $ye = 1$ and $y \geq 0$. Each x_j is the probability that player 1 selects decision j, and each y_i is the probability that player 2 selects decision

i. Hence, if the players apply randomized strategies x and y, the probability of payoff P_{ij} is $y_i x_j$ and the expected payoff is $\sum_{i=1}^{M} \sum_{j=1}^{N} y_i x_j P_{ij} = yPx$.

How should player 1 select a randomized policy? As a solution concept, we consider selection of a strategy that maximizes expected payoff, assuming that player 2 knows the strategy selected by player 1. One way to write this is as

$$\max_{\{x \in \Re^N | ex=1, x \geq 0\}} \min_{\{y \in \Re^M | ye=1, y \geq 0\}} yPx.$$

Here, y is chosen with knowledge of x, and x is chosen to maximize the worst-case payoff. We will now show how this optimization problem can be solved as a linear program.

First, consider the problem of optimizing y given x. This amounts to a linear program:

$$\begin{array}{ll} \text{minimize} & y(Px) \\ \text{subject to} & e^T y = 1 \\ & y \geq 0. \end{array}$$

It is easy to see that the basic feasible solutions of this linear program are given by $e^1, \ldots, e^M$, where each e^i is the vector with all components equal to 0 except for the ith, which is equal to 1. It follows that

$$\min_{\{y \in \Re^M | ye=1, y \geq 0\}} yPx = \min_{i \in \{1,\ldots,M\}} (Px)_i.$$

This minimal value can also be expressed as the solution to a linear program:

$$\begin{array}{ll} \text{maximize} & v \\ \text{subject to} & ve \leq Px, \end{array}$$

where $v \in \Re$ is the only decision variable and x is fixed. In particular, the optimal value v^* resulting from this linear program satisfies

$$v^* = \min_{\{y \in \Re^M | ye=1, y \geq 0\}} yPx.$$

To determine an optimal strategy for player 1, we find the value of x that maximizes v^*. In particular, an optimal strategy is delivered by the following linear program:

$$\begin{array}{ll} \text{maximize} & v \\ \text{subject to} & ve \leq Px \\ & ex = 1 \\ & x \geq 0, \end{array}$$

where $v \in \Re$ and $x \in \Re^N$ are decision variables. An optimal solution to this linear program provides a stochastic strategy x that maximizes the payoff v,

assuming that player 2 knows the randomized strategy of player 1 and selects a payoff-minimizing counter-strategy. We illustrate application of this linear program through a continuation of Example 4.4.2.

Example 4.4.2. (linear programming for drug running) *To determine an optimal drug running strategy, we formulate the problem in the terms we have introduced. The drug lord's strategy is represented as a vector* $x \in \Re^3$ *of three probabilities. The first, second, and third components represent the probabilities that a ship is sent to San Diego, Los Angeles, or San Francisco, respectively. The payoff is* 1 *if a ship gets through, and* 0 *otherwise. Hence, the expected payoff* P_{ij} *is the probability that a ship gets through if player 1 selects decision* j *and player 2 selects decision* i*. The payoff matrix is then*

$$P = \begin{bmatrix} 2/3 & 1 & 1 \\ 1 & 1/2 & 1 \\ 1 & 1 & 1/4 \end{bmatrix}.$$

The optimal strategy for the drug lord is given by a linear program:

$$\begin{array}{ll} \text{maximize} & v \\ \text{subject to} & ve \leq Px \\ & ex = 1 \\ & x \geq 0. \end{array}$$

Suppose that the drug lord computes an optimal randomized strategy x^* by solving the linear program. Over time, as this strategy is used to guide shipments, the drug lord can estimate the Coastguard's strategy y. Given y, he may consider adjusting his own strategy in response to y, if that will increase expected payoff. But should it be possible for the drug lord to improve his expected payoff after learning the Coastguard's strategy? Remarkably, if the coastguard selects a randomized strategy through an approach analogous to that we have described for the drug lord, neither the drug lord nor the Coastguard should ever need to adjust their strategies. We formalize this idea in the context of general two-player zero-sum games.

Recall from our earlier discussion that player 1 selects a randomized strategy x^* that attains the maximum in

$$\max_{\{x \in \Re^N | ex=1, x \geq 0\}} \min_{\{y \in \Re^M | ye=1, y \geq 0\}} yPx,$$

and that this can be done by solving a linear program

$$\begin{array}{ll} \text{maximize} & v \\ \text{subject to} & ve \leq Px \\ & ex = 1 \\ & x \geq 0. \end{array}$$

Consider determining a randomized strategy for player 2 through an analogous process. An optimal strategy will then be a vector y^* that attains the minimum in

$$\min_{\{y \in \Re^M | ye=1, y \geq 0\}} \max_{\{x \in \Re^N | ex=1, x \geq 0\}} yPx.$$

Similarly with the case of finding a strategy for player 1, this new problem can be converted to a linear program:

$$\begin{array}{ll} \text{minimize} & u \\ \text{subject to} & ue \geq yP \\ & ye = 1 \\ & y \geq 0. \end{array}$$

A remarkable fact is that – if player 1 uses x^* and player 2 uses y^* – neither player should have any reason to change his strategy after learning the strategy being used by the other player. Such a situation is referred to as an *equilibrium*. This fact is an immediate consequence of the minimax theorem:

Theorem 4.4.1. (Minimax) *For any matrix $P \in \Re^{M \times N}$,*

$$\max_{\{x \in \Re^N | ex=1, x \geq 0\}} \min_{\{y \in \Re^M | ye=1, y \geq 0\}} yPx = \min_{\{y \in \Re^M | ye=1, y \geq 0\}} \max_{\{x \in \Re^N | ex=1, x \geq 0\}} yPx.$$

The minimax theorem is a simple corollary of strong duality. In particular, it is easy to show that the linear programs solved by players 1 and 2 are duals of one another. Hence, their optimal objective values are equal, which is exactly what the minimax theorem states.

Suppose now that the linear program solved by player 1 yields an optimal solution x^*, while that solved by player 2 yields an optimal solution y^*. Then, the minimax theorem implies that

$$y^*Px \leq y^*Px^* \leq yPx^*,$$

for all $x \in \Re^N$ with $ex = 1$ and $x \geq 0$ and $y \in \Re^M$ with $ye = 1$ and $y \geq 0$. In other words, the pair of strategies (x^*, y^*) yield an equilibrium.

4.5 Allocation of a Labor Force

Our economy presents a network of interdependent industries. Each both produces and consumes goods. For example, the steel industry consumes coal to manufacture steel. Reciprocally, the coal industry requires steel to support its own production processes. Further, each industry may be served

by multiple manufacturing technologies, each of which requires different resources per unit production. For example, one technology for producing steel starts with iron ore while another makes use of scrap metal. In this section, we consider a hypothetical economy where labor is the only limiting resource. We will develop a model to guide how the labor force should be allocated among industries and technologies.

In our model, each industry produces a single good and may consume others. There are M goods, indexed $i = 1, \ldots, M$. Each can be produced by one or more technologies. There are a total of $N \geq M$ technologies, indexed $j = 1, \ldots, N$. Each jth technology produces $A_{ij} > 0$ units of some ith good per unit of labor. For each $k \neq i$, this jth industry may consume some amount of good k per unit labor, denoted by $A_{kj} \leq 0$. Note that this quantity A_{kj} is nonpositive; if it is a negative number, it represents the quantity of good k consumed per unit labor allocated to technology j. The productivity and resource requirements of all technologies are therefore captured by a matrix $A \in \Re^{M \times N}$ in which each column has exactly one positive entry and each row has at least one positive entry. We will call this matrix A the *production matrix*. Without loss of generality, we will assume that A has linearly independent rows.

Suppose we have a total of one unit of labor to allocate over the next year. Let us denote by $x \in \Re^N$ our allocation among the N technologies. Hence, $x \geq 0$ and $ex \leq 1$. Further, the quantity of each of the M goods produced is given by a vector Ax.

Now how should we allocate labor? One objective might be to optimize social welfare. Suppose that the amount society values each unit of each ith good is $c_i > 0$, regardless of the quantity produced. Then, we might define the social welfare generated by production activities to be cAx. Optimizing this objective leads to a linear program:

$$\begin{array}{ll} \text{maximize} & cAx \\ \text{subject to} & Ax \geq 0 \\ & ex \leq 1 \\ & x \geq 0. \end{array} \tag{4.2}$$

A production matrix A is said to be *productive* if there exists a labor allocation x (with $x \geq 0$ and $ex \leq 1$) such that $Ax > 0$. In other words, productivity means that some allocation results in positive quantities of every good. It turns out that – when the production matrix is productive – only M technologies are beneficial, and the choice of M technologies is independent of societal values. This remarkable result is known as the substitution theorem:

Theorem 4.5.1. (substitution) *If a production matrix A is productive, there is a set of M technologies such that for any vector c of societal values,*

social welfare can be maximized by an allocation of labor among only these M technologies.

In the remainder of this section, we will leverage linear algebra and duality theory to prove the substitution theorem.

4.5.1 Labor Minimization

Consider a related problem with an objective of minimizing the labor required to generate a particular "bill of goods" $b \in \Re^M$. Here, each b_i is nonnegative and represents the quantity of good i demanded by society. This problem is captured by the following linear program:

$$\begin{array}{ll} \text{minimize} & ex \\ \text{subject to} & Ax \geq b \\ & x \geq 0. \end{array} \tag{4.3}$$

As before, each x_j is the amount of labor allocated to the jth technology. The requirement is that we produce at least b_i units of each ith good, and we wish to minimize the quantity ex of labor used to accomplish this. The following lemma relates solutions of (4.3) to (4.2).

Lemma 4.5.1. *Let x^* be an optimal solution to (4.2) and let $b = Ax^*$. Then, the set of optimal solutions to (4.3) is the same as the set of optimal solutions to (4.2).*

Proof: Note that $ex^* = 1$; if this were not true, x^*/ex^* would be another feasible solution to (4.2) with objective value $cAx^*/ex^* > cx^*$, which would contradict the fact that x^* is an optimal solution.

We now show that x^* (and therefore any optimal solution to (4.2)) is an optimal solution to (4.3). Let $\overline{x}$ be an optimal solution to (4.3). Assume for contradiction that x^* is not an optimal solutoin to (4.3). Then, $e\overline{x} < ex^*$ and

$$cA\overline{x}/e\overline{x} = cb/e\overline{x} > cb/ex^* = cAx^*.$$

Since $\overline{x}/e\overline{x}$ is a feasible solution to (4.2), this implies that x^* is not an optimal solution to (4.2), which is a contradiction. The conclusion is that x^* is an optimal solution to (4.3).

Since the fact that x^* is an optimal solution to (4.3) implies that $e\overline{x} = ex^* = 1$. It follows that $\overline{x}$ is a feasible solutoin to (4.2). Since $cA\overline{x} \geq cb = cAx^*$, $\overline{x}$ is also an optimal solution to (4.2). □

4.5.2 Productivity Implies Flexibility

Consider M technologies, each of which produces one of the M goods. Together they can be described by an $M \times M$ production matrix A. Interestingly, if A is productive then any bill of goods can be met exactly by appropriate application of these technologies. This represents a sort of flexibility – any demands for goods can be met without any excess supply. This fact is captured by the following lemma.

Lemma 4.5.2. *If a square production matrix $A \in \Re^{M \times M}$ is productive then for any $b \geq 0$, the equation $Ax = b$ has a unique solution $x \in \Re^M$, which satisfies $x \geq 0$.*

Proof: Since A is productive, there exists a vector $\overline{x} \geq 0$ such that $A\overline{x} > 0$. The fact that only one element of each column of A is positive implies that $\overline{x} > 0$.

Since the rows of A are linearly independent, $Ax = b$ has a unique solution $x \in \Re^N$. Assume for contradiction that there is some $\hat{b} \geq 0$ and $\hat{x}$ with at least one negative component such that $A\hat{x} = \hat{b}$. Let

$$\alpha = \min\{\alpha \geq 0 | \alpha\overline{x} + \hat{x} \geq 0\},$$

and note that $\alpha > 0$ because some component of $\hat{x}$ is negative. Since only one element of each column of A is positive, this implies that at least one component of $A(\alpha\overline{x} + \hat{x})$ is nonpositive.

Recall that $A\overline{x} > 0$ and $A\hat{x} \geq 0$, and therefore

$$A\hat{x} < \alpha A\overline{x} + A\hat{x} = A(\alpha\overline{x} + \hat{x}),$$

contradicting the fact that at least one component of $A(\alpha\overline{x}+\hat{x})$ is nonpositive. □

4.5.3 Proof of the Substitution Theorem

Since A is productive, there exists a vector $\overline{x} \geq 0$ such that $A\overline{x} > 0$. Let $b^1 = A\overline{x}$ and consider the labor minimization problem:

$$\begin{array}{ll} \text{minimize} & e^T x \\ \text{subject to} & Ax \geq b^1 \\ & x \geq 0. \end{array}$$

Let x^1 be an optimal basic feasible solution and note that $Ax^1 \geq b^1 > 0$. Since x^1 is a basic feasible solution, at least $N - M$ components must be

equal to zero, and therefore, at most M components can be positive. Hence, the allocation x^1 makes use of at most M technologies. Lemma 4.5.2 implies that these M technologies could be used to fill any bill of goods.

Given an arbitrary bill of goods $b^2 \geq 0$, we now know that there is a vector $x^2 \geq 0$, with $x^2_{j_k} = 0$ for $k = 1, \ldots, N - M$, such that $Ax^2 = b^2$. Note that $x^2 \geq 0$ and satisfies N linearly independent constraints of and is therefore a basic feasible solution of the associated labor minimization problem:

$$\begin{array}{ll} \text{minimize} & ex \\ \text{subject to} & Ax \geq b^2 \\ & x \geq 0. \end{array}$$

Let y^1 be an optimal solution to the dual of the labor minimization problem with bill of goods b^1. By complementary slackness, we have $(e - y^1 A)x^1 = 0$. Since $x^2_j = 0$ if $x^1_j = 0$, we also have $(e - y^1 A)x^2 = 0$. Further, since $Ax^2 = b^2$, we have $y^1(Ax^2 - b^2) = 0$. Along with the fact that x^2 is a feasible solution, this gives us the complementary slackness conditions required to ensure that x^2 and y^1 are optimal primal and dual solutions to the labor minimization problem with bill of goods b^2.

We conclude that there is a set of M technologies that is sufficient to attain the optimum in the labor minimization problem for any bill of goods. It follows from Lemma 4.5.1 that the same set of M technologies is sufficient to attain the optimum in the social welfare maximization problem for any societal values. □

4.6 Exercises

Question 1

Consider the following linear program (LP).

$$\begin{array}{rrcl} & \max\ x_1 - x_2 & & \\ s.t. & -2x_1 - 3x_2 & \leq & -4 \\ & -x_1 + x_2 & \leq & -1 \\ & x_1, x_2 & >= & 0 \end{array}$$

(a) Plot the feasible region of the primal and show that the primal objective value goes to infinity.

(b) Formulate the dual, plot the feasible region of the dual, and show that it is empty.

Question 2

Convert the following optimization problem into a symmetric form linear program, and then find the dual.

$$\begin{aligned} \max \quad & -x_1 - 2x_2 - x_3 & & \\ s.t. \quad & x_1 + x_2 + x_3 & = & \ 1 \\ & |x_1| & \leq & \ 4 \\ & x_1, x_2, x_3 & \geq & \ 0 \end{aligned}$$

Note: $|x|$ denotes the absolute value of x.

Question 3

consider the LP

$$\begin{aligned} \max \quad & -x_1 - x_2 & & \\ s.t. \quad & -x_1 - 2x_2 & \leq & -3 \\ & -x_1 + 2x_2 & \leq & \ 4 \\ & x_1 + 7x_2 & \leq & \ 6 \\ & x_1, x_2 & \geq & \ 0 \end{aligned}$$

(a) solve this problem in Excel using solver. After solver finds an optimal solution, ask it to generate the sensitivity report.

(b) Find the dual of the above LP. Read the shadow prices from the sensitivity report, and verify that it satisfies the dual and gives the same dual objective value as the primal.

Question 4

Consider the LP

$$\begin{aligned} \min \quad & 2x_1 + x_2 & & \\ s.t. \quad & x_1 + x_2 & \leq & \ 6 \\ & x_1 + 3x_2 & \geq & \ 3 \\ & x_1, x_2, x_3 & \geq & \ 0 \end{aligned}$$

Note: there is an x_3.
(a) plot the feasible region and solve the problem graphically.

(b) Rearrange into symmetric form and find the dual. Solve the dual graphically.
(c) Verify that primal and dual optimal solutions satisfy the Strong Duality Theorem.

Question 5

(a) Consider the problem of feeding an army presented in Section 3.1.2. Provide a dual linear program whose solution gives sensitivities of the cost of an optimal diet to nutritional requirements. Check to see whether the sensitivities computed by solving this dual linear program match the sensitivities given by Solver when solving the primal.
(b) Suppose a pharmaceutical company wishes to win a contract with the army to sell digestible capsules containing pure nutrients. They sell three types of capsule, with 1 grams of carbohydrates, 1 gram of fiber, and 1 grams of protein, respectively. The army requires that the company provide a price for each capsule independently, and that substituting the nutritional value of any food item with capsules should be no more expensive than buying the food item. Explain the relation between the situation faced by the pharmaceutical company and the dual linear program.

Question 6

Consider a symmetric form primal linear program:

$$\begin{array}{ll} \text{maximize} & cx \\ \text{subject to} & Ax \leq b \\ & x \geq 0. \end{array}$$

(a) Find the dual of this problem. Noting that $\max f(x) = \min -f(x)$ rearrange the dual into symmetric form. Take the dual of you answer, and rearrange again to get into symmetric form.

(b) Explain why the sensitivities of the optimal dual objective to the dual constraints should be optimal primal variables.

Question 7

Recall Question 8, Homework 4, the diet Problem for the pigs. We found an optimal solution for that problem (see Solution Homework 4). Now, suppose that Dwight doesn't have a good estimate of the price of Feed type A because of some turbulence in the market. Therefore, he would like to know how

sensitive is his original optimal solution with respect to changes in the price of Feed Type A. In particular, in what range around \$0.4 can the price of Feed Type A change, without changing the original optimal solution? For prices out of this range, what are the new optimal solutions? Now suppose Dwight doesn't have a good estimate of the requirements of vitamins. In what range around 700 does the requirement of vitamins can change without changing the original optimal solution? For values out of this range, what are the new optimal solutions? Your arguments should be geometric (and not based on Excel), that is, you should draw the problem in R^2 and see what is going on.

Question 8

Consider the linear program

$$\begin{array}{ll} \text{maximize} & -q^T z \\ \text{subject to} & Mz \leq q \\ & z \geq 0. \end{array}$$

where

$$M = \begin{bmatrix} 0 & A \\ -A^T & 0 \end{bmatrix}, \quad q = \begin{bmatrix} c \\ -b \end{bmatrix}, \quad z = \begin{bmatrix} x \\ y \end{bmatrix}.$$

a) Derive the dual
b) Show that the optimal solution to the dual is given by that of the primal and vice-versa.
c) Show that the primal problem has an optimal solution if and only if it is feasible.

Question 9

Consider the following LP.

$$\begin{array}{ll} \text{maximize} & cx \\ \text{subject to} & Ax \leq b \\ & x \geq 0. \end{array}$$

Show that if A, b and c are all positive, then both the primal and dual feasible regions are non-empty.

Question 10

Why is it that, if the primal has unique optimal solution x^*, there is a sufficiently small amount by which c can be altered without changing the optimal solution?

Question 11 - Games and Duals

Show that the linear programs given in the notes to determine the strategies of player 1 and player 2 are indeed duals of one another.

Question 12 - Drug Runners

Consider the drug running example from the notes. Imagine the drug lord has a 4th alternative which involves transporting his drugs overland.

a) Suppose that the DEA can reassign its agents from coastguard duty to guarding the border, and if it does so any shipment of drugs transported overland will be caught. Will the drug lord ever choose to send shipments overland? If so, with what probability? If not, why not.

b) Suppose that guarding the border does not require the coastguard to reassign its agents, so that it can still guard a port (for instance, suppose the customs agents at the border are sufficiently equipped to detect most drug shipments). Suppose that an overland shipment will get through 85% of the time. Will the drug lord ever choose to go overland? With what probability?

c) Suppose the percentage in part b) were changed to %80. What would be the probability now.

d) Suppose the percentage in part b) were changed to %90. What would be the probability now.

e) Suppose that if the DEA reassigns its agents to the border, they will certainly catch any overland drug shipments, but if the agents are not reassigned, then the customs agents will catch %80 of all drug shipments. Should the DEA ever assign their agents to the border? With what probability?

Question 13 - Elementary Asset Pricing and Arbitrage

You are examining a market to see if you can find arbitrage opportunities. For the same of simplicity, imagine that there are M states that the market could be in next year, and you are only considering buying a portfolio now, and selling it in a years time. There are also N assets in the market, with price vector $\rho \in \Re^N$. The payoff matrix is P. So, the price of asset i is ρ_i and the payoff of asset i in state j is P_{ji}.

You have observed some relationships amongst the prices of the assets, in particular you have found that the price vector is in the *row space* of P.

a) Suppose that $\rho = qP$ for some $q \in \Re^M$. An *elementary asset* is another name for an Arrow-Debreu security. That is, it is an asset that pays \$1 in a particular state of the market, and \$0 in any other state. So, there are M elementary assets, and the ith elementary asset pays \$1 if the market is

in state i, and \$0 otherwise. Suppose that the ith elementary asset can be constructed from a portfolio $x \in \Re^N$. What is the price of x in terms of q and ρ? (Assume that there are no arbitrage opportunities.)

b) Suppose that however I write ρ as a linear combination of rows of P, the coefficient of each row is non-negative. Is it possible for there to be an arbitrage opportunity? If not, why not. If so, give an example.

c) Suppose that ρ can be written as a linear combination of rows of P, where the coefficient of each row is non-negative. Is it possible for their to be an arbitrage opportunity? If not, why not. If so, give an example. Note, this is a weaker condition than (b) because there may be many ways of writing ρ as a linear combination of rows of P.

Hint: Note that ρ being in the row space of P means that there is a vector q such that $\rho = qP$. Consider a portfolio x.

Chapter 5

Network Flows

A wide variety of engineering and management problems involve optimization of network flows – that is, how objects move through a network. Examples include coordination of trucks in a transportation system, routing of packets in a communication network, and sequencing of legs for air travel. Such problems often involve few indivisible objects, and this leads to a finite set of feasible solutions. For example, consider the problem of finding a minimal cost sequence of legs for air travel from Nuku'alofa to Reykjavik. Though there are many routes that will get a traveller from one place to the other, the number is finite. This may appear as a striking difference that distinguishes network flows problems from linear programs – the latter always involves a polyhedral set of feasible solutions. Surprisingly, as we will see in this chapter, network flows problems can often be formulated and solved as linear programs.

5.1 Networks

A network is characterized by a collection of nodes and directed edges, called a *directed graph*. Each edge points from one node to another. Figure 5.1 offers a visual representation of a directed graph with nodes labelled 1 through 8. We will denote an edge pointing from a node i to a node j by (i, j). In this notation, the graph of Figure 5.1 can be characterized in terms of a set of nodes $V = \{1, 2, 3, 4, 5, 6, 7, 8\}$ and a set of edges $E = \{(1, 2), (1, 3), (1, 6), (2, 5), (3, 4), (4, 6), (5, 8), (6, 5), (6, 7), (7, 8)\}$.

Graphs can be used to model many real networked systems. For example, in modelling air travel, each node might represent an airport, and each edge a route taken by some flight. Note that, to solve a specific problem, one often requires more information than the topology captured by a graph. For

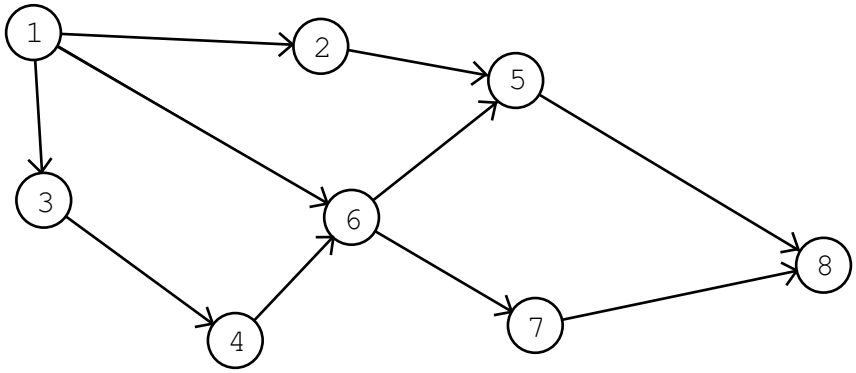


Figure 5.1: A directed graph.

example, to minimize cost of air travel, one would need to know costs of tickets for various routes.

5.2 Min-Cost-Flow Problems

Consider a directed graph with a set V of nodes and a set E of edges. In a min-cost-flow problem, each edge $(i, j) \in E$ is associated with a cost c_{ij} and a capacity constraint u_{ij}. There is one decision variable f_{ij} per edge $(i, j) \in E$. Each f_{ij} represents a flow of objects from i to j. The cost of a flow f_{ij} is $c_{ij}f_{ij}$. Each node $j \in V$ satisfies a flow constraint:

$$\sum_{\{k|(j,k)\in E\}} f_{jk} - \sum_{\{i|(i,j)\in E\}} f_{ij} = b_j,$$

where b_j denotes an amount of flow generated by node j. Note that if b_j is negative, the node consumes flow.

The min-cost-flow problem is to find flows that minimize total cost subject to capacity and flow conservation constraints. It can be written as a linear program:

$$\begin{array}{lll} \text{minimize} & \sum_{(i,j)\in E} c_{ij}f_{ij} & \\ \text{subject to} & \sum_{\{k|(j,k)\in E\}} f_{jk} - \sum_{\{i|(i,j)\in E\}} f_{ij} = b_j, & \forall j \in V \\ & 0 \le f_{ij} \le u_{ij}, & \forall (i,j) \in E. \end{array} \tag{5.1}$$

An important property of the min-cost-flow problem is that basic feasible solutions are integer-valued if capacity constraints and quantities of flow produced at each node are integer-valued, as captured by the following theorem.

Theorem 5.2.1. *If for all $i \in V$, b_i is an integer, and for all $(i,j) \in E$, u_{ij} is an integer, then for any basic feasible solution of the linear program (5.1), each flow f_{ij} is an integer.*

Proof. Our proof builds on three facts:

1. A basic feasible solution can not be a convex combination of two other feasible solutions.

2. If f is a feasible solution and there is an incoming or outgoing edge at node i for which the flow is fractional then there must be at least one additional incoming or outgoing edge at node i for which the flow is fractional.

3. If there is fractional flow along an edge, the capacity constraints for that edge cannot be binding.

The first fact was established in Chapter 3. The second follows from the flow conservation constraint. The third is a consequence of integer-valued capacity constraints.

Assume for contradiction that there is a basic feasible solution f such that for some edge (i,j), the flow f_{ij} takes on a fractional value. It follows from Fact 2 above that there must be a cycle of edges in the graph, each of which has fractional flow. Then, by Fact 3, a small flow of ϵ can be added to or subtracted from the cycle, while retaining feasibility either way. Clearly, f is the convex combination of these two alternatives. Therefore, by Fact 1, f is not a basic feasible solution. □

Since basic feasible solutions are integer-valued, when there is an optimal solution, there will be one that is integer-valued. This enables use of linear programming algorithms to solve min-cost-flow problems even when integer-valued solutions are required. We discuss some examples in the following subsections.

5.2.1 Shortest Path Problems

Consider a graph with nodes V and a set of edges E. Think of each node as a city and each edge as a highway that can be used to send a truck from one city to another. Suppose we are given the travel time c_{ij} associated with each highway, and we want to get the truck from node o (the origin) to node d (the destination) in minimum time. Suppose that for each $(i,j) \in E$, we take f_{ij} to be a binary variable to which we would assign a value of 1 if edge (i,j) is to be part of the route and 0 otherwise. Then, the route with shortest travel time can be found by solving the a min-cost-flow problem with $b_o = 1$,

$b_d = -1$, $b_i = 0$ for $i \notin \{o, d\}$, $u_{ij} = 1$ for each $(i, j) \in E$, and an additional constraint that $f_{ij} \in \{0, 1\}$ for each $(i, j) \in E$.

The additional set of constraints that require flow variables to be binary are introduced because it is not possible to send a fraction of the truck along a highway. With the additional constraints, the problem is not a linear program. Fortunately, we can drop these additional constraints and by doing so obtain a min-cost-flow problem – a linear program for which basic feasible solutions are integer-valued. Because basic feasible solutions are integer-valued, if there exists an optimal solution, which is the case if there is a path from node o to node d, then there will be an integer-valued optimal solution. This means a binary-valued solution since each flow is constrained to be between 0 and 1.

5.2.2 Transportation Problems

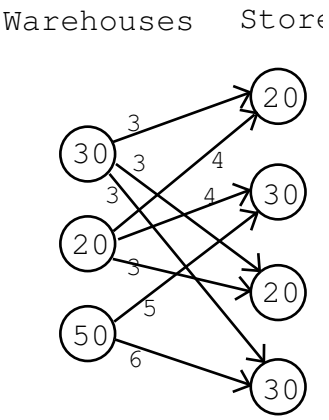


Figure 5.2: A transportation problem.

Consider a situation illustrated in Figure 5.2 where we have inventory at warehouses to be transported to retail stores. Each circle in the left column represents a warehouse, while each circle on the right column represents a store. Each warehouse holds a certain supply of inventory, given by the number written in its corresponding circle. Each store demands a certain amount of inventory, given by the number written in *its* circle. Each line represents a route through which inventory can be transported from a warehouse to a store and is marked with a number that indicates per unit inventory transportation cost. The units are indivisible. The goal is to satisfy demands while minimizing transportation costs.

The problem illustrated in Figure 5.2 is an instance of the transportation problem. More generally, the transportation problem involves:

- n suppliers with inventory amounts $S_1, \ldots, S_m$,
- m demand sites with inventory demands $D_1, \ldots, D_n$,
- a set T of routes from suppliers to demand sites,
- costs c_{ij}, for each $(i, j) \in T$ for transporting one unit from supplier i to demand site j.

Transportation problems can be formulated as min-cost-flow problems with one node per supplier and one node per demand site. Edges correspond to elements of T. There are not capacity constraints (i.e., each u_{ij} is set to ∞). The flow quantities produced by suppliers are given by $S_1, \ldots, S_m$ and the quantities required at demand sites are given by $D_1, \ldots, D_n$. Since this is a min-cost-flow problem, basic feasible solutions assign integer values to variables. Hence, this formulation addresses the requirement that units of inventory are indivisible.

5.2.3 Assignment Problems

The assignment problem involves allocating a number of jobs to workers in some optimal manner. A simple example should be sufficient to demonstrate the central ideas.

Example 5.2.1. *The coach of a swim team needs to assign swimmers to a 200-yard medley relay team to compete in a tournament. The problem facing him is that his best swimmers are good in more than one stroke, so it is not clear which swimmer to assign to which stroke. The 5 fastest swimmers and the best times (in seconds) they have achieved with each of the strokes (for 50 yards) are given below.*

Stroke	*Carl*	*Chris*	*David*	*Tony*	*Ken*
Backstroke	*37.7*	*32.9*	*33.8*	*37.0*	*35.4*
Breaststroke	*43.4*	*33.1*	*42.2*	*34.7*	*41.8*
Butterfly	*33.3*	*28.5*	*38.9*	*30.4*	*33.6*
Freestyle	*29.2*	*26.4*	*29.6*	*28.5*	*31.1*

The problem is to try to minimize the sum of the best times for the people competing in the race.

The assignment problem is a special case of the transportation problem. To convert the above example to a transportation problem think of each swimmer as a supplier with one unit of inventory and each stroke as a site

demanding one unit of inventory. There is an edge from each swimmer to each stroke in which he can participate. Assign a cost to each edge equal to the time it takes the swimmer to do the stroke. The only problem that arises is that the number of swimmers is greater than the number of strokes. One way to deal with this is to introduce a dummy stroke (the "not taking part" stroke), and treat that as an additional site with one unit of demand. The time for each swimmer assigned to the dummy stroke would be zero.

5.3 Max-Flow Problems

Consider a graph with a set of vertices V, a set of edges E, and two distinguished nodes o and d. Each edge has an associated capacity u_{ij} but no associated cost. We will assume that there is no edge from d to o. In a max-flow problem, the goal is to maximize the total flow from o to d. In our formulation, nodes do not produce or consume flow, but rather, we introduce an auxiliary edge (d, o) with no capacity limit and aim to maximize flow along this edge. By doing so, we indirectly maximize flow from o to d via the edges in E. We define an augmented set of edges $E' = E \cup \{(d, o)\}$. The max-flow problem is given by

$$\begin{array}{lll} \text{maximize} & f_{do} & \\ \text{subject to} & \sum_{\{k|(j,k)\in E'\}} f_{jk} - \sum_{\{i|(i,j)\in E'\}} f_{ij} = 0, & \forall j \in V \\ & 0 \le f_{ij} \le u_{ij}, & \forall (i,j) \in E. \end{array} \tag{5.2}$$

Note that this can be thought of as a special case of the min-cost-flow problem, where all edges have zero cost, except for (d, o), which is assigned a cost of $c_{do} = -1$.

5.3.1 Min-Cut Problems

An interesting related problem is the min-cut problem. A *cut* is a partition of the nodes V into two disjoint sets V_o and V_d such that $o \in V_o$, $d \in V_d$, and $V_o \cup V_d = V$. The objective in the min-cut problem is to find a cut such that the capacity for flows from V_o to V_d is minimized. One way to represent a choice of partition involves assigning a binary variable to each node. In particular, for each node $i \in V$, let $p_i = 0$ if $i \in V_o$ and $p_i = 1$ if $i \in V_d$. Note that for nodes o and d, we always have $p_o = 0$ and $p_d = 1$. Further, for each $(i, j) \in E$, if $i \in V_o$ and $j \in V_d$, let $q_{ij} = 1$. Otherwise, let $q_{ij} = 0$. Note that

$q_{ij} = \max(p_j - p_i, 0)$. The min-cut problem can be written as

$$\begin{array}{lll} \text{minimize} & \sum_{(i,j)\in E} u_{ij} q_{ij} & \\ \text{subject to} & q_{ij} = \max(p_j - p_i, 0), & \forall (i,j) \in E, \\ & p_d - p_o = 1, & \\ & p_i \in \{0,1\}, & \forall i \in V. \end{array} \tag{5.3}$$

The decision variables being optimized here include each p_i for $i \in V$ and each q_{ij} for $(i,j) \in E$. Since the problem involves minimizing a weighted combination of q_{ij}'s, with positive weights, and each q_{ij} is constrained to $\{0,1\}$, the constraint that $q_{ij} = \max(p_j - p_i, 0)$ can be converted to $q_{ij} \geq p_j - p_i$, without changing the optimal solution. Hence, an equivalent optimization problem is:

$$\begin{array}{lll} \text{minimize} & \sum_{(i,j)\in E} u_{ij} q_{ij} & \\ \text{subject to} & q_{ij} \geq p_j - p_i, & \forall (i,j) \in E, \\ & p_d - p_o = 1, & \\ & q_{ij} \geq 0, & \forall (i,j) \in E, \\ & p_i \in \{0,1\}, & \forall i \in V. \end{array} \tag{5.4}$$

It is easy to see that the optimal objective value for the min-cut problem is greater than or equal to the optimal objective value for the max-flow problem. This is because for any cut, flow that gets from o to d must pass from V_o to V_d. Hence, the maximal flow is limited by the capacity for flow from V_o to V_d.

The min-cut problem as stated above is not a linear program because the variables are constrained to be binary. As we will now explain, the following linear program also solves the min-cut problem:

$$\begin{array}{lll} \text{minimize} & \sum_{(i,j)\in E} u_{ij} q_{ij} & \\ \text{subject to} & q_{ij} \geq p_j - p_i, & \forall (i,j) \in E, \\ & p_d - p_o = 1, & \\ & q_{ij} \geq 0, & \forall (i,j) \in E. \end{array} \tag{5.5}$$

This linear program is the dual of the max-flow problem (5.2). We will not go through the mechanics of converting the max-flow problem to its dual here – that is straightforward to do. However, we will argue that, as a consequence, at each optimal basic feasible solution, each q_{ij} is binary valued.

At an optimal basic feasible solution, each q_{ij} can be viewed as the sensitivity of max-flow to the capacity constraint u_{ij}. At an optimal basic feasible solution to the dual, q_{ij} is equal to either the rate at which flow increases as a consequence increasing u_{ij} or the rate at which flow decreases as a consequence of decreasing u_{ij}. Because all capacities are integer-valued, if increasing u_{ij} slightly can increase flow, the rate of increase in flow must

equal the rate of increase in capacity. The situation with decreasing u_{ij} is analogous. Hence, if the sensitivity is nonzero, it is equal to one.

Note that the p_is may not be binary-valued; for example, adding a constant to every p_i maintains feasibility and does not alter the objective value. So given binary-valued q_{ij}s, how do we recover the cut? Well, at an optimal basic feasible solution, $q_{ij} = 1$ if and only if the edge (i, j) goes from V_o to V_d. Based on this, one can easily recover the cut.

5.3.2 Matching Problems

Consider making matches among n potential grooms and n potential brides. We have data on which couples will or will not be happy together, and the goal is to create as many happy couples as possible. One might represent the compatibilities in terms of a graph as depicted in Figure 5.3. The problem of finding a matching that maximizes the number of happy couples is known as the *matching problem.*

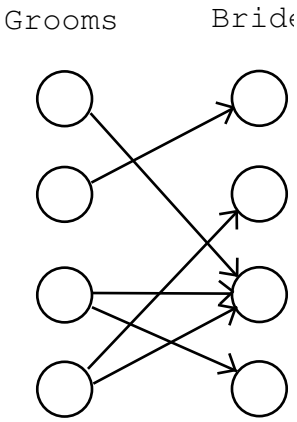


Figure 5.3: A matching problem.

Clearly, this problem can be treated as a max-flow problem by adding origin and destination nodes as shown in Figure 5.4. The capacity of each edge is one. By solving the linear program associated with this max-flow problem, we arrive at a solution that maximizes the number of happy couples. If we are lucky this will turn out to be n. But sometimes, it may not. Interestingly, there is a simple rule for determining whether or not there will be n happy couples without solving the linear program.

Theorem 5.3.1. (Hall's Marriage Lemma) *There exists a matching such that every bride and every groom is happy if and only if for any set of grooms,*

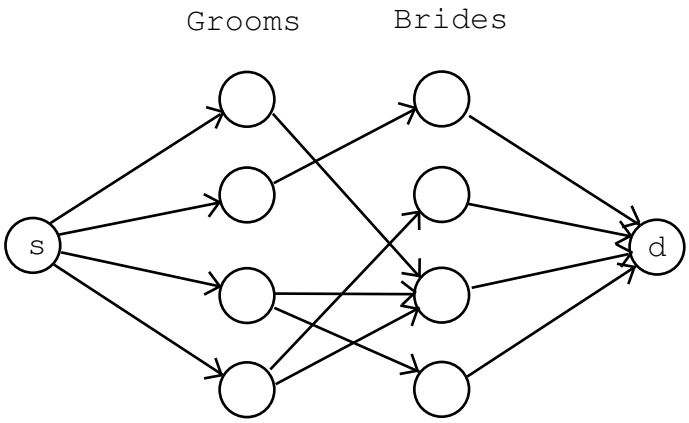


Figure 5.4: Conversion to a max-flow problem.

the set of brides that would form a happy couple with at least one of these grooms is at least as large as the set of grooms.

Proof. It is obvious that for any set A of grooms, there has to be a set of brides of at least equal size in order for there to be any hope of a perfect matching. What is less clear is that there must be a perfect matching if this condition holds for every set of grooms. This follows from the fact that there is a perfect matching if the maximal flow/minimal cut equals n (the number of grooms/brides). It is easy to see that if the conditions on grooms and brides holds, there cannot be any cut of value less than n, and the result follows. □

5.4 Exercises

Question 1 - Shortest Path and LP

You are going to visit a national park, and have never been there before. You are using a map to try and make the distance travelled as short as possible. There are 5 intermediate towns, A, B, C, D, and E, you may go through on your way to the park, and the distances between the various locations are given below.

	Miles between Adjacent Towns					
Town	A	B	C	D	E	Destination
Origin	40	60	50	-	-	-
A		10	-	70	-	-
B			20	55	40	-
C				-	50	-
D					10	60
E						80

In the table above, a dash means that there is no direct road between the two locations.

1. Draw the graph corresponding to the above problem.

2. Give a linear program which will find the shortest path from Origin to Destination.

3. Solve the problem in Excel.

4. Suppose the numbers in the table now represented the time taken (rather than the distance). What would be the minimum time taken to travel from Origin to Destination?

5. Suppose the number in the table now represented capacities for water pipes connecting the various locations (in units of gal/sec). What is the maximum amount of water that can be pumped from Origin to Destination?

Question 2 - Networks and Swimming

The coach of a swim team needs to assign swimmers to a 200-yard medley relay team to compete in a tournament. The problem facing him is that his best swimmers are good in more than one stroke, so it is not clear which swimmer to assign to which stroke. The 5 fastest swimmers and the best times (in seconds) they have achieved with each of the strokes (for 50 yards) are given below.

Stroke	Carl	Chris	David	Tony	Ken
Backstroke	37.7	32.9	33.8	37.0	35.4
Breaststroke	43.4	33.1	42.2	34.7	41.8
Butterfly	33.3	28.5	38.9	30.4	33.6
Freestyle	29.2	26.4	29.6	28.5	31.1

The problem is to try to minimize the sum of the best times for the people competing in the race.

1. Draw a network that describes this problem. Note that one person will not be assigned.

2. Formulate the problem as a linear program. Solve it on Excel.

3. Write the dual of the problem.

4. If the coach can concentrate on improving one persons time on one stroke by 0.5 seconds. On whom should the coach spend his time (note, there may be more than one answer).

Question 3 - Graduation Party Matching

Suppose you are hosting a graduation party in your dorm. n women and n men live in your dorm (including yourself) and the party will only include the residents of the dorm. Tired of the usual process of finding a partner, you as a group decide to use a centralized matching mechanism for setting up couples for the party (each couple consists of one man and one woman). Moreover you decide this is a good chance to use for the last time linear programming before graduating.

The matching mechanism is described in what follows:

- Let $i = 1, ..., n$ be an index of the men and let $j = 1, ..., n$ be an index of the women, so that each woman (and each man) is given a label between 1 and n.

- Each woman and each man declares her/his preferences as a ranking. Let $w^j \in \Re^n$ be woman j's ranking. w^j_k is the ranking of the kth man for woman j. For example, if there are 4 men, a possible ranking for woman 1 would be $w^1 = (3, 4, 2, 1)$, meaning that man 1 is ranked in the third place, man 2 is ranked in the fourth place (he is the least preferred), man 3 is ranked second and man 4 is ranked first (he is the most preferred). Similarly, let $m^i \in \Re^n$ be man's i ranking of women.

- Each woman (man) must be paired with exactly one man (woman). The centralized matching allocates the n woman-man pairs in order to minimize the sum of the products of the rankings of the allocated pairs. For example, if woman 3 is paired with man 4, then the product of rankings is $c_{3,4} = w^3_4 \cdot m^4_3$. We want to minimize the sum of these products over the allocated pairs. We call the outcome(s) of this algorithm the "Product Matching(s)" (the solution is not necessarily unique).

Answer the following:

1. Formulate a Linear Programming problem that determines the "Product Matching(s)".

2. Is the solution of the LP an integer solution? Why or why not?

3. Suppose there are 6 women and 6 men. The rankings are given in the attached excel file, *rankings.xls*. Find the "Product Matching" using Excel.

4. We say that a matching is "stable" if there is no couple that would like to deviate from the matching to become partners. For example, suppose the matching assigns couples 1-2 and 3-4, the first number being the woman and the second being the man. Suppose woman 1 ranks man 2 sixth and man 4 fifth; and man 4 ranks woman 1 fifth and woman 3 sixth. Then woman 1 and man 4 would like to deviate from the matching. They are matched to their sixth preference, but if they deviate and become partners they will be matched to their fifth. Is the Product Matching from the previous example stable? Why or why not?

Chapter 6

The Simplex Method

Standard Form

The simplex algorithm solves a linear program in standard form

$$\begin{array}{ll} \text{maximize} & cx \\ \text{subject to} & Ax = b \\ & x \geq 0, \end{array}$$

for which the rows of $A \in \Re^{M \times N}$ are linearly independent. If the rows of A were not linearly independent and the linear program was feasible, we could remove equality constraints corresponding to linearly dependent rows to arrive at an equivalent linear program with a new matrix A made up of linearly independent rows.

Note that any linear program can be converted to standard form. For example, consider a linear program in symmetric form:

$$\begin{array}{ll} \text{maximize} & cx \\ \text{subject to} & Ax \leq b \\ & x \geq 0. \end{array}$$

Introducing slack variables $w \in \Re^M$, we can rewrite the linear program as

$$\begin{array}{ll} \text{maximize} & [c \;\; 0] \left[\begin{array}{c} x \\ w \end{array} \right] \\ \text{subject to} & [A \;\; I] \left[\begin{array}{c} x \\ w \end{array} \right] = b \\ & \left[\begin{array}{c} x \\ w \end{array} \right] \geq 0, \end{array}$$

which is in standard form.

Vertices

Recall that when there are N decision variables, a vertex is identified by N linearly independent binding constraints. The constraints $Ax = b$ are always binding since they are equality constraints, and since the rows of A are linearly independent, this supplies M of the N linearly independent constraints required to identify a vertex. Note that $M \leq N$ because a matrix A with N columns can have at most N linearly independent rows. The remaining $N-M$ binding constraints must therefore be nonnegativity constraints. This observation gives rise to an elegant way of identifying a vertex. In particular, any given vertex can be identified by specifying $N - M$ nonnegativity constraints that are binding and linearly independent of the equality constraints $Ax = b$. Solving for $Ax = b$ with those constraints binding yields a unique solution.

In linear programming nomenclature, the $N - M$ variables constrained to zero by specifying $N - M$ binding nonnegativity constraints are called *nonbasic variables*. The remaining M variables, which are determined by solving $Ax = b$, are called *basic variables*. Hence, to identify any given vertex, we can simply indicate which M variables are basic at that solution. This seems like an efficient way of encoding a vertex; in particular, it does not require specifying numerical values of the decision variables.

To reinforce the point, consider a vertex $x^\dagger \in \Re^N$. Let $B \subseteq \{1, \ldots, N\}$ be a set of indices of basic variables that identifies $x^\dagger$ and let $\overline{B}$ be the set of indices of the corresponding nonbasic variables. Note that B has M elements while $\overline{B}$ has $N - M$ elements. Based on our earlier discussion, the set B uniquely identifies the vertex $x^\dagger$. In particular, $x^\dagger$ is the unique solution to

$$\begin{aligned} Ax &= b \\ x_j &= 0 \qquad \forall j \in \overline{B}. \end{aligned}$$

Moving Along an Edge

Consider again a set $B \subseteq \{1, \ldots, N\}$ of indices of basic variables that identifies a vertex $x^\dagger$ and let $\overline{B}$ be the set of indices of the corresponding nonbasic variables. Let $A_B \in \Re^{M \times M}$ and $A_{\overline{B}} \in \Re^{M \times (N-M)}$ be matrices made up of columns of A indexed by B and $\overline{B}$, respectively. Similarly, let $c_B, x_B \in \Re^M$ and $c_{\overline{B}}, x_{\overline{B}} \in \Re^{N-M}$ be vectors made up of components of x indexed by B or $\overline{B}$. Note that $x^\dagger$ is given by

$$\begin{aligned} x^\dagger_B &= A_B^{-1} b \\ x^\dagger_{\overline{B}} &= 0. \end{aligned}$$

Consider a vertex $x^\diamond$ that is adjacent to $x^\dagger$. This means that the two vertices are connected by an edge of the polytope of feasible solutions. It also means that there are $N-1$ shared constraints that are binding at both vertices. In other words, the set of basic variables B that identifies $x^\dagger$ and the set $B^\diamond$ that identifies $x^\diamond$ share $M-1$ common elements. Equivalently, $\overline{B}$ and $\overline{B}^\diamond$ share $N-M-1$ common elements.

Let k be the index of the variable that is in $\overline{B}$ but not $\overline{B}^\diamond$. To move along the edge from $x^\dagger$ to $x^\diamond$, the value of the kth variable is increased from $x^\dagger_k = 0$ to some $\alpha > 0$. Solutions x along this edge are given by fixing $x_k = \alpha$ and imposing constraints $x_j = 0$ for remaining nonbasic variables $j \in \overline{B} \setminus k$ as well as the equality constraints $Ax = b$ to obtain x_B. Noting that $Ax = A_B x_B + A_{*k} x_k$ in this context, we obtain an expression for basic variables $x_B = x^\dagger_B - A_B^{-1} A_{\overline{B}} x_{\overline{B}}$. The new vertex $x^\diamond$ is arrived at when α is just large enough so that one of the variables in B becomes zero. Note that for any $\ell \in B$, x_ℓ becomes zero when

$$\begin{aligned} 0 &= x_\ell \\ &= x^\dagger_\ell - (A_B^{-1} A_{\overline{B}} x_{\overline{B}})_\ell \\ &= (A_B^{-1} b)_\ell - (A_B^{-1} A_{*k} \alpha)_\ell. \end{aligned}$$

Consequently, x_ℓ becomes zero when

$$\alpha = \frac{(A_B^{-1} b)_\ell}{(A_B^{-1} A_{*k})_\ell}.$$

It follows that the value of α and the index ℓ of the first variable to become zero are computed by solving

$$\alpha = \min \left\{ \beta_\ell = \frac{(A_B^{-1} b)_\ell}{(A_B^{-1} A_{*k})_\ell} \;\middle|\; \ell \in B, \beta_\ell \geq 0 \right\}.$$

This involves computing the above ratio for each possible value of $\ell \in B$ and selecting the minimizer among those that are nonnegative. The set of basic variables $B^\diamond$ is given by removing ℓ from B and including k instead.

Reduced Profits

Note that as a nonbasic variable $x_k = \alpha$ is increased and we move along an edge, the objective value of solutions x along the edge are given by

$$c_B (x^\dagger_B - A_B^{-1} A_{\overline{B}} x_{\overline{B}}) + c_{\overline{B}} x_{\overline{B}} = c_B x^\dagger_B + (c_{\overline{B}} - c_B A_B^{-1} A_{\overline{B}}) x_{\overline{B}}.$$

Note that the only part of this expression that varies with x_k is

$$(c_{\overline{B}} - c_B A_B^{-1} A_{\overline{B}}) x_{\overline{B}} = (c_{\overline{B}} - c_B A_B^{-1} A_{\overline{B}})_k x_k.$$

The value of $(c_{\overline{B}} - c_B A_B^{-1} A_{\overline{B}})_k$ represents the rate at which profits increase as x_k increases. In linear programming nomenclature, this is called the *reduced profit* of variable k. The vector r of reduced profits at a vertex with basic variables B is given by

$$r_B = 0$$

$$r_{\overline{B}} = c_{\overline{B}} - c_B A_B^{-1} A_{\overline{B}}$$

Simplex Algorithm

Each iteration of the simplex algorithm begins with a vertex, identified by a set B of basic variables. The reduced profits $r_{\overline{B}}$ are computed. If every component of $r_{\overline{B}}$ is nonpositive, the current vertex is optimal. Otherwise, the index $k \in \overline{B}$ with maximal reduced profits is identified, and the corresponding variable x_k is converted to a basic variable. This means that x_k is increased until $x_\ell = 0$ for some $\ell \in B$. Algorithm 1 assembles the ideas we have discussed to provide an algorithm for solving a standard form linear program given an initial vertex with basic variables B.

Other Issues

Our discussion of the simplex method sidestepped a few issues that are discussed at length in a number of more extensive treatments. One issue is initialization: how do we obtain a vertex to start at before executing simplex iterations? This turns out to be easy if $b \geq 0$ and the standard form linear program to be solved was obtained from a symmetric form linear program as discussed earlier. In this case, the linear program can be written as

$$\begin{aligned} \text{minimize} \quad & [c \;\; 0] \begin{bmatrix} x \\ w \end{bmatrix} \\ \text{subject to} \quad & [A \;\; I] \begin{bmatrix} x \\ w \end{bmatrix} = b \\ & \begin{bmatrix} x \\ w \end{bmatrix} \geq 0, \end{aligned}$$

and setting $x = 0$ and $w = b$ generates a vertex. Other cases require more complicated initialization methods which we will not discuss further here.

Algorithm 1 Simplex Algorithm

1: **repeat**

2: Compute reduced profits

$$r_{\overline{B}} = c_{\overline{B}} - c_B A_B^{-1} A_{\overline{B}}$$

$$r_B = 0$$

3: Select nonbasic variable to increase

$$k = \operatorname*{argmax}_j r_j$$

4: **if** $r_k > 0$ **then**

5: Identify first basic variable to become zero as x_k is increased

$$\beta_j = \frac{(A_B^{-1} b)_j}{(A_B^{-1} A_{*k})_j}, \quad \text{for } j \in B$$

$$\ell = \operatorname*{argmin}_{\{j \in B | \beta_j > 0 \text{ or } ((A_B^{-1} b)_j = 0,\ (A_B^{-1} A_{*k})_j > 0)\}} \beta_j.$$

If there exist multiple minimizing j's, choose the smallest one.

6: Modify set of basic variables

$$B := (B \cup k) \setminus \ell$$

7: **end if**

8: **until** $r_k \leq 0$

8: Compute optimal solution

$$x_B^* = A_B^{-1} b$$

$$x_{\overline{B}}^* = 0$$

Another pair of issues involve degeneracy and cycling. Degeneracy arises when more than N constraints are binding at a vertex. In this case, one or more basic variables are equal to 0 in addition to the nonbasic variables, which are equal to zero whether or not there is any degeneracy. Though this is not immediately obvious, degeneracy arises routinely in practical problem formulations. When there is degeneracy, the amount by which a nonbasic variable can be increased before a basic variable becomes 0 may itself be 0. This means a simplex iteration can revise the set of basic variables without changing numerical values of the variables. This would be a problem if it leads to cycling, where the set of basic variables keeps changing back and forth but the variable values never do. However, cycling arises so rarely in practice that this is considered an irrelevant issue.

For those who are interested in how to avoid cycling, let us introduce a very simple rule called Bland's rule: Choose the smallest index j when there are ties between minimum values of β_j in Line 5 of Algorithm 1. It can be shown that this rule guarantees no cycling.

Another host of issues arise when one tries to organize the computations required for simplex iterations in an efficient way. Central to this is the common use of a data structure called the *tableau*. We will not discuss these computational tricks here, though.

Chapter 7

Quadratic Programs

In this chapter we consider a generalization of linear programs in which the objective function is quadratic rather than linear. In particular, we consider maximizing an objective function of the form $f(x) = cx - x^T Hx$, where H is a positive semidefinite symmetric matrix. In other words, the matrix H satisfies $H = H^T$ and, for all $x \in \Re^N$, $x^T Hx \geq 0$. As was the case with linear programs, we will require for such problems that the feasible region is defined by linear constraints. The resulting class of optimization problems, which we will refer to as quadratic programs, hence take the form

$$\begin{array}{ll} \text{maximize} & cx - x^T Hx \\ \text{subject to} & Ax \leq b. \end{array}$$

Such problems can be solved efficiently, though not quite as efficiently as linear programs. Because there are problem instances where maxima appear only in the interior of the feasible region, the simplex method is not suited for quadratic programs. However, quadratic programs are commonly solved by *interior point methods*. The following sections discuss examples of quadratic programs that arise in important application domains.

7.1 Portfolio Optimization

Portfolio optimization addresses the problem of distributing investments in a way that strikes a suitable balance between expected profit and risk. The problem is to select dollar amounts $x_1, \ldots, x_N$ to invest in each of N opportunities. Note that negative amounts here represent short positions. We consider a formulation where investment decisions are guided by the rate of return we expect from each of the N opportunities, as well as a covariance matrix that reflects our beliefs concerning risks associated with each investment as well as their interdependencies.

Let $\mu \in \Re^N$ denote our expected returns and Σ denote the covariance matrix. Note that covariance matrices are always positive semidefinite and typically positive definite. Given investment amounts $x \in \Re^N$, we should expect profit of μx, placing a variance of $x^T \Sigma x$ around this expectation. As a mathematical formulation for guiding our investment decisions, we consider as an objective maximizing expected profit while penalizing for variance. In particular, we consider the problem:

$$\text{maximize} \quad \mu x - \rho x^T \Sigma x.$$

Here, $\rho \in \Re_+$ is a positive scalar that represents our level of risk aversion. In particular, larger values of ρ reflect a larger penalty being associated with any given level of variance. Note that this problem is a quadratic program. The objective function takes the form $cx - x^T Hx$ with $c = \mu$ and $H = \rho\Sigma$, while the feasible region is the entire Euclidean space $\Re^N$ because there are no constraints. Unlike linear programs, which are typically unbounded in the absence of constraints, quadratic programs typically are not. In this case, if ρ is positive and Σ is positive definite, the penalty on variance gives rise to an objective function that takes on increasingly large negative values as the solution x becomes sufficiently far from the origin in any direction. As such, the problem should not be unbounded.

In many practical contexts it is important to place constraints on the vector x of dollar investments. For example, regulations prevent certain classes of investment funds from holding short positions. It is easy to incorporate constraints that lead to solutions meeting such requirements:

$$\begin{array}{ll} \text{maximize} & \mu x - \rho x^T \Sigma x \\ \text{subject to} & x \geq 0. \end{array}$$

This new problem is also a quadratic program, since the objective is quadratic, $\rho\Sigma$ is positive semidefinite, and the constraints are linear.

7.2 Maximum Margin Classification

Recall the problem of finding a hyperplane to separate two classes of data. As before, denote positive samples by $u^1, \ldots, u^M \in \Re^N$ and negative samples by $v^1, \ldots, v^K \in \Re^N$. A separating hyperplane can be obtained by finding a vector $x \in \Re^N$ and scalar $\alpha \in \Re$ such that

$$\begin{array}{ll} x^T u^m - \alpha \geq 1 & \forall m = 1, \ldots, M \\ -x^T v^k + \alpha \geq 1 & \forall k = 1, \ldots, K. \end{array}$$

This set of linear inequalities can be solved using linear programming software. Further, any separating hyperplane can be encoded by some (x, α) that satisfies these inequalities.

In this section, we consider a quadratic program that not only finds a separating hyperplane but, additionally, among all separating hyperplane, finds one that maximizes distance to the closest data sample. We will call this distance the *margin* of the separating hyperplane. The problem of maximizing margin is of interest because hyperplanes that maximize margin tend to generalize more effectively than arbitrary separating hyperplanes. In particular, though any separating hyperplane perfectly classifies data in the *training set*, defined to be the positive and negative samples used to select the hyperplane, it can err on distinct data samples that the hyperplane is subsequently used to classify. One guarantee that can be made, however, is that a sample is classified correctly so long as its distance from some training sample of the same class is less than the margin. Hence, maximizing the margin increases the chances that samples beyond the training set will be correctly classified.

In order to formulate the problem of maximizing margin, we will derive a formula for margin. Consider a hyperplane encoded by (x, α); i.e., the hyperplane made up of all points $y \in \Re^N$ such that $x^T y = \alpha$. The distance between the hyperplane and an arbitrary point $z \in \Re^N$ is the length of a line segment that is orthogonal to the hyperplane, touching the hyperplane at one end and z at the other. The point at which this line segment touches the hyperplane must be equal to $z + \beta x$ for some $\beta \in \Re$, since x identifies the direction orthogonal to the hyperplane. Note that the scalar β uniquely solves $x^T(z + \beta x) = \alpha$. Hence, $\beta = (\alpha - x^T z)/\|x\|^2$. It follows that the distance between the hyperplane and z is

$$\|z - (z + \beta x)\| = \left\| \frac{\alpha - x^T z}{\|x\|^2} x \right\| = \frac{|\alpha - x^T z|}{\|x\|}.$$

Given this formula for the distance between a hyperplane and a point, we arrive at the following formula for the margin of a hyperplane (x, α) given positive and negative data samples:

$$\text{margin}(x, \alpha) = \frac{1}{\|x\|} \min_{z \in \{u^1, \ldots, u^M, v^1, \ldots, v^K\}} |\alpha - x^T z|.$$

We would like to find a hyperplane (x, α) that maximizes margin. However, the associated optimization problem, which can be written as

$$\begin{array}{lll} \text{maximize} & \text{margin}(x, \alpha) & \\ \text{subject to} & x^T u^m - \alpha \geq 1 & \forall m = 1, \ldots, M \\ & -x^T v^k + \alpha \geq 1 & \forall k = 1, \ldots, K. \end{array}$$

is not a linear or quadratic program, given our formula for margin.

For reasons we will explain, any optimal solution of the following quadratic program turns out to maximize margin among separating hyperplanes:

$$\begin{array}{lll} \text{minimize} & \|x\|^2 & \\ \text{subject to} & x^T u^m - \alpha \geq 1 & \forall m = 1, \ldots, M \\ & -x^T v^k + \alpha \geq 1 & \forall k = 1, \ldots, K. \end{array}$$

Note that this is a quadratic program because the constraints are linear and the objective can be viewed as maximizing $c^T x - x^T H x$, with $c = 0$ and $H = I$. To see why solving this optimization problem results in margin maximization, first note that, given an optimal solution (x, α), at least one of the inequality constraints must be met with equality; otherwise, x must be in the interior of the feasible region, and therefore, $\|x\|$ can be decreased without violating constraints. It follows that, at an optimal solution, $\min_{z \in \{u^1, \ldots, u^M, v^1, \ldots, v^K\}} |\alpha - x^T z| = 1$, and therefore, $\text{margin}(x, \alpha) = 1/\|x\|$. It is easy to see that any optimal solution to our quadratic program also maximizes $1/\|x\|$ over feasible solutions, since $1/\|x\|$ is monotonically decreasing in $\|x\|^2$. Hence, any optimal solution to our quadratic program must also optimize

$$\begin{array}{lll} \text{maximize} & \text{margin}(x, \alpha) & \\ \text{subject to} & x^T u^m - \alpha \geq 1 & \forall m = 1, \ldots, M \\ & -x^T v^k + \alpha \geq 1 & \forall k = 1, \ldots, K. \end{array}$$