Linear Algebra

Foundations of Machine Learning

Paul A. Jensen
University of Illinois at Urbana-Champaign

Spring 2022 Edition

Contents

Introduction

This class has three parts. In Part I, we analyze linear systems that transform a multidimensional vector (**x**) into another vector (**y**). This transformation is often expressed as a linear system via matrix multiplication: $\mathbf{y} = \mathbf{A}\mathbf{x}$. Some linear systems can be solved exactly and uniquely, but we will also consider solution strategies with alternative objectives when the system contains either too much or not enough information. In Part II we shift to nonlinear systems that must be solved approximately via iterative methods. The nonlinear methods include many of the foundational techniques in machine learning. Part III focuses on matrix theory, expanding our understanding of high-dimensional data.. We will learn how to analyze and extract information from matrices without a clear input/output relationship.

Notation

We will distinguish scalars, vectors, and matrices with the following typographic conventions:

Object	Font and Symbol	Examples
Scalars	italicized, lowercase letters	x, α, y
Vectors	bold lowercase letters	$\mathbf{x}$, $\mathbf{y}$, $\mathbf{n}$, $\mathbf{w}$
Matrices	bold uppercase	$\mathbf{A}$, $\mathbf{A}^{-1}$, $\mathbf{B}$, $\mathbf{\Gamma}$

There are many ways to represent rows or columns of a matrix **A**. Since this course uses Matlab, I think it is convenient to use a matrix addressing scheme that reflects Matlab's syntax. So, the ith row of matrix **A** will be $\mathbf{A}(i,:)$, and the jth column will be $\mathbf{A}(:,j)$. Rows or columns of a matrix are themselves vectors, so we choose to keep the boldface font for the matrix **A** even when it is subscripted. We could also use Matlab syntax for vectors ($\mathbf{x}(i)$, for example). However, the form x_i is standard across many fields of mathematics and engineering, so we retain the common notation. The lack of boldface font reminds us that elements of vectors are scalars in the field.

One goal of this class is to increase the precision of your mathematical writing. We will regularly use the symbols in Table 1 to describe mathematical concepts and relations. These symbols succinctly express mathematical ideas. For example, we can define the set of rational numbers as

A number is rational if and only if it can be expressed as the quotient of two integers.

with the statement

$$r \in \mathbb{Q} \Leftrightarrow \exists\ p, q \in \mathbb{Z} \text{ s.t. } r = p/q$$

While the latter statement is shorter, it is more difficult to understand. So whenever possible I recommend writing statements with as few symbols as necessary. Rely on mathematical symbols only when a textual definition would be unwieldy or imprecise, or when brevity is important (like when writing on a chalkboard).

Acknowledgements

Michael Insana had the foresight to include linear algebra as part of every bioengineer's training. Gregory Underhill co-taught the course in 2018 and helped transform it into the present structure. Several teaching assistants and graders developed problems and solutions, including Dikshant Pradhan, Hyeon Ryoo, Cynthia Liu, and Boeun Hwang. The BIOE 210 students provided enormous feedback to improve the content and delivery.

Table 1: Mathematical notation used in this book.

Symbol	Read As	Description
$\Rightarrow$	implies	$p \Rightarrow q$ means that whenever p is true, q must also be true.
$\Leftrightarrow$	if and only if	A symmetric, stronger version of $\Rightarrow$. The expression $p \Leftrightarrow q$ means $p \Rightarrow q$ and $q \Rightarrow p$.
$\forall$	for all	Remember this symbol as an upside down "A", as in "for **A**ll".
$\exists$	there exists	Remember this symbol as a backwards "E", as in "there **E**xists". To say something does not exist, use $\nexists$.
$\in$ ($\notin$)	is (not) a member of	Used to state that a single element is a member of a set, i.e. $1 \in \mathbb{Z}$. To say that a set is a subset of another set, use $\subset$.
s.t.	such that	Other texts use the symbol $\mid$ (a vertical pipe) instead of "s.t.". Note that "s.t." is set in normal, not italicized font.
$\mathbb{R}$	the real numbers	The numbers along a line. The reals include both rational and irrational numbers.
$\mathbb{R}^n$	the set of n-dimensional vectors of real numbers	Each value of n is a different set. If $r \in \mathbb{R}^2$ then $r \notin \mathbb{R}^3$. Also, $\mathbb{R}^2$ is not a subset of $\mathbb{R}^3$, etc.
$\mathbb{Z}$	the integers	The integers contain the natural numbers (1, 2, 3, . . .), their negatives (-1, -2, -3, . . .), and the number zero (0). The symbol comes from the German word for "number" (Zahlen). The word "integer" (Latin for "whole") is used since integers have no fractional part.
$\mathbb{Q}$	the rationals	The rational numbers are all numbers that are the quotient of two integers. The symbol derives from the word "quotient".
$\mapsto$	maps to	Describes the inputs and outputs of an operation. An operation that maps a vector of reals to a real number is $\mathbb{R}^n \mapsto \mathbb{R}$. An operation that maps two integers to a rational is $\mathbb{Z} \times \mathbb{Z} \mapsto \mathbb{Q}$.
$\equiv$	is defined as	Two expressions are equivalent because we have defined them as such, not because the relationship can be shown logically. For example, $a/b \equiv a \times b^{-1}$ defines the division operator using the multiplicative inverse.

Part I

Linear Systems

Chapter 1

Fields and Vectors

1.1 Algebra

Algebra is a branch of mathematics that contains symbols and a set of rules to manipulate them.

You are probably familiar with the idea of symbols. We call them variables, and in previous algebra courses you used them to represent unknown real numbers. In this course we will use variables to represent *vectors*. Vectors are collections of elements, such as the real numbers. When we say vector, we assume a *column vector*—a vertical array of elements. A *row vector* is a horizontal array of elements. We will see that column vectors are more convenient. The number of elements in a vector is its dimension. We can write an n-dimensional vector as

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}.$$

We surround the elements of a vector with either parentheses ( ) or square brackets [ ]. Straight lines | | or curly braces { } are not allowed, as these have special meanings. While some vectors have elements that are real numbers, vectors themselves are not numbers. An n-dimensional vector does not belong to the set $\mathbb{R}$ of real numbers; it belongs to a special set $\mathbb{R}^n$ of all other vectors of dimension n with real elements.

We use the rules of algebra to manipulate elements. However, only certain sets of elements are amenable to the rules of algebra. These algebra-compatible sets are called *fields*. A set of conditions, or *axioms*, must be true about a set before we can consider it a field. These field axioms define the rules of algebra.

After spending years studying algebra, you might think that there are many byzantine rules that govern fields. In fact, there are only five. The five axioms describe only two operations (addition and multiplication) and define two special elements that must be in every field (0 and 1).

1.2 The Field Axioms

Given elements a, b, and c in a field:

1. **Associativity**.
$$a + b + c = (a + b) + c = a + (b + c)$$
$$abc = (ab)c = a(bc)$$

2. **Commutativity**.
$$a + b = b + a$$
$$ab = ba$$

3. **Distribution** of multiplication over addition.
$$a(b + c) = ab + ac$$

4. **Identity**. There exist elements 0 and 1, both in the field, such that
$$a + 0 = a$$
$$1 \times a = a$$

5. **Inverses**.
   - For all a, there exists an element $(-a)$ in the field such that $a + (-a) = 0$. The element $-a$ is called the *additive inverse* of a.
   - For all $a \neq 0$, there exists an element (a^{-1}) in the field such that $a \times a^{-1} = 1$. The element a^{-1} is called the *multiplicative inverse* of a.

It might surprise you that only five axioms are sufficient to recreate everything you know about algebra. For example, nowhere do we state the special property of zero that $a \times 0 = 0$ for any number a. We don't need to state this property, as it follows from the field axioms.

Theorem. $a \times 0 = 0$.

Proof.

$$\begin{aligned} a \times 0 &= a \times (1 - 1) \\ &= a \times 1 + a \times (-1) \\ &= a - a \\ &= 0 \end{aligned}$$

□

Similarly, we can prove corollaries from the field axioms.

A corollary is a statement that follows directly from a theorem.

Corollary. *If $ab = 0$, then either $a = 0$ or $b = 0$ (or both).*

Proof. Suppose $a \neq 0$. Then there exists a^{-1} such that

$$\begin{aligned} a^{-1}ab &= a^{-1} \times 0 \\ 1 \times b &= 0 \\ b &= 0 \end{aligned}$$

A similar argument follows when $b \neq 0$. □

The fundamental theorem of algebra relies on the above corollary when solving polynomials. If we factor a polynomial into the form $(x - r_1)(x - r_2) \cdots (x - r_k) = 0$, then we know the polynomial has roots $r_1, r_2, \ldots, r_k$. This is only true because the left hand side of the factored expression only reaches zero when at least one of the factors is zero, i.e. when $x = r_i$.

1.2.1 Common Fields in Mathematics

The advantage of fields is that once a set is proven to obey the five field axioms, we can operate on elements in the field just like we would operate on real numbers. Besides the real numbers (which the concept of fields was designed to emulate), what are some other fields?

The rational numbers are a field. The numbers 0 and 1 are rational, so they are in the field. Since we add and multiply rational numbers just as we do real numbers, these operations commute, associate, and distribute. All that remains is to show that the rationals have additive and multiplicative inverses in the field. Let us consider a rational number p/q, where p and q are integers.

- We know that $-p/q$ is also rational, since $-p$ is still an integer. The additive inverse of a rational number is in the field of rational numbers.

- The additive inverse of p/q is q/p, which is also rational. The multiplicative inverse of a rational is also in the field.

So the rational numbers are a field. What does this mean? If we are given an algebraic expression, we can solve it by performing any algebraic manipulation and still be assured that the answer will be another rational number.

The integers, by contrast, are not a field. Every integer has a reciprocal ($2 \longrightarrow 1/2$, $-100 \longrightarrow -1/100$, etc.). However, the reciprocals are themselves not integers, so they are not in the same field. The field axioms require that the inverses for every element are members of the field. When constructing a field, every part of every axiom must be satisfied.

Example 1.1. Let's demonstrate why *every* axiom must hold in a field. Imagine the simple equation $y = ax + b$, which we solve for x to yield

$$x = \frac{y-b}{a}.$$

If we wanted to solve this equation using only rational numbers, we would not need to change anything. So long as the values we input for the variables a, b, and y were rational, the value of x would also be rational. We solved the equation using field algebra, and the rationals constitute a field. Everything works out.

Now imagine you wanted only integer solutions. Even if the values for a, b, and y were integers, there is no guarantee that x would be an integer. ($a = 2$, $b = 4$, and $y = 3$ yields $x = -1/2$, for example). Because the integers are not a field, algebra does not work on them. In particular, the integers do not have integer multiplicative inverses (except for 1 and -1). When we divide by a, we assumed that the value $1/a$ exists in the field, which it does not. ■

Interestingly, the integers always have integer additive inverses, so the solution to the equation $y = x - b$ is always an integer (for integer y and b) since we could solve the equation with only additive inverses.

1.3 Vector Addition

Addition of two vectors is defined *elementwise*, or element-by-element.

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \vdots \\ x_n + y_n \end{pmatrix}$$

Since this is a direct extension of scalar addition, it is clear that vector addition commutes $[\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}]$ and is associative $[\mathbf{x} + \mathbf{y} + \mathbf{z} = (\mathbf{x} + \mathbf{y}) + \mathbf{z} = \mathbf{x} + (\mathbf{y} + \mathbf{z})]$.

The additive inverse of a vector $\mathbf{x}$ (written $-\mathbf{x}$) is also constructed elementwise.

$$-\mathbf{x} = \begin{pmatrix} -x_1 \\ -x_2 \\ \vdots \\ -x_n \end{pmatrix}$$

From our elementwise definition of vector addition, we can construct the zero element for the vectors. We know from the field axioms that $\mathbf{x} + \mathbf{0} = \mathbf{x}$, so the zero element must be a vector of the same dimension with all zero entries.

Notice that each set of n-dimensional vectors has its own zero element. In $\mathbb{R}^2$, $\mathbf{0} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. In $\mathbb{R}^3$, $\mathbf{0} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$.

$$\mathbf{x} + \mathbf{0} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} x_1 + 0 \\ x_2 + 0 \\ \vdots \\ x_n + 0 \end{pmatrix} = \mathbf{x}$$

1.4 Vector Multiplication is not Elementwise

What happens when we try to define multiplication as an elementwise operation? For example

$$\begin{pmatrix} -1 \\ 0 \\ 4 \end{pmatrix} \times \begin{pmatrix} 0 \\ 2 \\ 0 \end{pmatrix} = \begin{pmatrix} -1 \times 0 \\ 0 \times 2 \\ 4 \times 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} = \mathbf{0}.$$

This is bad. Very bad. Here we have an example where $\mathbf{xy} = \mathbf{0}$, but neither $\mathbf{x}$ nor $\mathbf{y}$ is the zero element $\mathbf{0}$. This is a direct violation of a corollary of the field axioms, so **elementwise vector multiplication is not a valid algebraic operation**.

On the bright side, if vectors were a field this class would be far too short.

Sadly, vectors are not a field. There is no way to define multiplication using only vectors that satisfies the field axioms. Nor is there anything close to a complete set of multiplicative inverses, or even the element $\mathbf{1}$. Instead, we will settle for a weaker result—showing that vectors live in a *normed inner product space*. The concepts of a vector norm and inner product will let us create most of the operations and elements that vectors need to be a field.

Example 1.2: Do we need multiplication? When you were first taught to multiply, it was probably introduced as a "faster" method of addition, i.e. $4 \times 3 = 3+3+3+3$. If so, why do we need multiplication as a separate requirement for fields? Couldn't we simply require the addition operator and construct multiplication from it? The answer is no, for two reasons. First, the idea of multiplication as a shortcut for addition only makes sense when discussing the nonnegative integers. What, for example, does it mean to have -2.86×-3.2? What do -2.86 or -3.2 groups look like in terms of addition?

Also, the integers are not a field!

Second, we must realize that multiplication is a much stronger relationship between numbers. To understand why, we should start talking about the "linear" part of linear algebra. ■

1.5 Linear Systems

Linear systems have two special properties.

1. **Proportionality**. If the input to a linear system is multiplied by a scalar, the output is multiplied by the same scalar: $f(kx) = kf(x)$.
2. **Additivity**. If two inputs are added, the result is the sum of the original outputs: $f(x_1 + x_2) = f(x_1) + f(x_2)$.

We can combine both of these properties into a single condition for linearity.

Definition. *A system f is linear if and only if*

$$f(k_1x_1 + k_2x_2) = k_1f(x_1) + k_2f(x_2)$$

for all inputs x_1 and x_2 and scalars k_1 and k_2.

Consider a very simple function, $f(x) = x + 3$. Is this function linear? First we calculate the lefthand side of the definition of linearity.

$$f(k_1x_1 + k_2x_2) = k_1x_1 + k_2x_2 + 3$$

We compare this to the righthand side.

$$\begin{aligned} k_1f(x_1) + k_2f(x_2) &= k_1(x_1 + 3) + k_2(x_2 + 3) \\ &= k_1x_1 + k_2x_2 + 3(k_1 + k_2) \\ &\neq f(k_1x_1 + k_2x_2) \end{aligned}$$

This does not follow the definition of linearity. The function $f(x) = x + 3$ is not linear. Now let's look at a simple function involving multiplication: $f(x) = 3x$. Is this function linear?

$$\begin{aligned} f(k_1x_1 + k_2x_2) &= 3(k_1x_1 + k_2x_2) \\ &= k_1(3x_1) + k_2(3x_2) \\ &= k_1f(x_1) + k_2f(x_2) \end{aligned}$$

The function involving multiplication is linear.

These results might not be what you expected, at least concerning the nonlinearity of functions of the form $f(x) = x + b$. This is probably because in earlier math courses you referred to equations of straight lines ($y = mx + b$) as linear equations. In fact, any equation of this form (with $b \neq 0$) is called *affine*, not linear.

Truly linear functions have the property that $f(0) = 0$. Addition is, in a way, not "strong" enough to drive a function to zero. The expression $x + y$ is zero only when both x and y are zero. By contrast, the product xy is zero when either x or y is zero.

This follows from proportionality. If $f(k0) = kf(0)$ for all k, then $f(0)$ must equal zero.

1.6 Vector Norms

One of the nice properties of the real numbers is that they are *well ordered*. Being well ordered means that for any two real numbers, we can determine which number is larger (or if the two numbers are equal). Well orderedness allows us to make all sorts of comparisons between the real numbers.

Vectors are not well ordered. Consider the vectors $\begin{pmatrix}3\\4\end{pmatrix}$ and $\begin{pmatrix}5\\2\end{pmatrix}$. Which one is larger? Each vector has one element that is larger than the other (4 in the first, 5 in the second). There is no unambiguous way to place all vectors in order.

This doesn't stop us from making comparisons between vector quantities. Consider velocity, which, contrary to how many people use the term, is a vector. Since vectors are not ordered, we should not be able to compare velocities. Instead, we often compare speeds, which are the magnitude of the velocity vectors. Traveling 30 mph due north and 30 mph due east are technically two different velocities. However, they have the same magnitude (30 mph), so most people consider them equivalent.

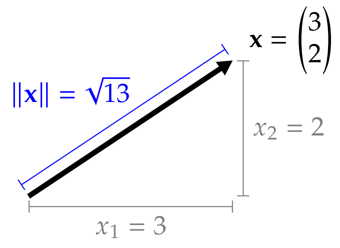

$\mathbf{x} = \begin{pmatrix}3\\2\end{pmatrix}$
$\|\mathbf{x}\| = \sqrt{13}$
$x_2 = 2$
$x_1 = 3$

Figure 1.1: The vector norm.

Vector magnitudes are calculated by taking a *norm* of the vector. There are many different kinds of norms, but the most commonly used norm is the 2-norm (or Euclidean or Pythagorean norm). We will refer to the 2-norm as simply "the norm" unless we state otherwise. If we treat a vector as a point in n-dimensional space, the norm is the length of the arrow drawn from the origin to that point. We use a pair of two vertical bars ($\|\cdot\|$) to represent the norm. This differentiates the norm from the absolute value (which is, in fact, the 1-norm). Sometimes we use a subscript to identify which norm we are taking, i.e. $\|\mathbf{x}\|_2$ is the 2-norm of $\mathbf{x}$.

In 2D, the norm corresponds to the hypotenuse of the right triangle with sides equivalent to the two elements in the vector—hence the name "Pythagorean norm" since the norm can be calculated by the Pythagorean theorem. In higher dimensions, we generalize the Pythagorean definition of the norm to

$$\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}.$$

In one dimension, taking the 2-norm yields the same result as taking the absolute value:

$$\|-3\| = \sqrt{(-3)^2} = 3 = |-3|.$$

Be careful not to confuse the 2-norm and the absolute value, as they are not the same thing. Their equivalence in one dimension is a coincidence. However, the absolute value is a norm—it returns the magnitude of a number, but strips away the direction (negative or positive).

There are two useful properties of norms that derive directly from its definition. These properties must be true of all norms, not just the 2-norm.

1. **Nonnegativity**. $\|\mathbf{x}\| \geq 0$
2. **Zero Identity**. $\|\mathbf{x}\| = 0 \Leftrightarrow \mathbf{x} = \mathbf{0}$

There are other properties that define norms, such as the triangle inequality ($\|\mathbf{x}+\mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$) or scaling ($\|k\mathbf{x}\| = |k|\,\|\mathbf{x}\|$). The nonnegativity and zero identity properties will be the most useful in this book.

1.6.1 Generalized Norms

We mentioned above that the Euclidean (or 2-norm) is only one type of norm. Some common norms include:

1. The **1-norm**, taxicab, or Manhattan norm, is the sum of the absolute values of the element in a vector:

   $$\|\mathbf{x}\|_1 \equiv |x_1| + |x_2| + \cdots + |x_n|.$$

   The "taxicab" name comes from distance you'd travel between two points when driving in cities with a grid street layout (like Manhattan). If you can't drive diagonally through any city blocks, the distance between two points is the 1-norm.

2. The **∞-norm** is the absolute value of the largest element in the vector:

   $$\|\mathbf{x}\|_\infty \equiv \max\{|x_1|, |x_2|, \ldots, |x_n|\}.$$

3. The **0-norm**, also called the cardinality of a vector, is the number of nonzero element in the vector:

   $$\|\mathbf{x}\|_0 \equiv \#\text{ of nonzero } x_i.$$

   The 0-norm is not a true norm since it does not satisfy all of the properties of norms. Still, the 0-norm is useful in many machine learning applications, so we include it here.

All of the norms listed above are members of the family of p-norms. In general, the p-norm of an n-dimensional vector $\mathbf{x}$ is

$$\|\mathbf{x}\|_p \equiv \left(\sum_{i=1}^{n} x_i^p\right)^{1/p} .$$

Setting $p = 2$ gives the Euclidian norm, and $p = 1$ is the taxicab norm. As p approaches infinity, the p norm becomes the ∞-norm. Setting $p = 0$ would give the 0-norm if we overlook the $1/0$ that appears in the exponent—again, the 0-norm is not a true norm, so some mathematical leniency is needed.

1.6.2 Normalized (Unit) Vectors

A vector contains information about both its magnitude and its orientation. We've seen how to extract the magnitude as a scalar from the vector by taking the norm. Is it possible to similarly separate out the vector's orientation? Yes, by *normalizing* the vector. A normalized vector (or *unit vector*) is any vector with magnitude equal to one. We can convert a vector to a normalized vector by dividing each element by the vector's magnitude. For example

$$\mathbf{x} = \begin{pmatrix} 3 \\ -4 \end{pmatrix} \Rightarrow \|\mathbf{x}\| = \sqrt{3^2 + (-4)^2} = 5.$$

The normalized unit vector ($\hat{\mathbf{x}}$) is

$$\hat{\mathbf{x}} = \begin{pmatrix} 3/\|\mathbf{x}\| \\ -4/\|\mathbf{x}\| \end{pmatrix} = \begin{pmatrix} 3/5 \\ -4/5 \end{pmatrix}.$$

We use the hat symbol (ˆ) over a unit vector to remind us that it has been normalized.

Intuitively, the idea of a normalized unit vector as a direction makes sense. If a vector is a product of both a magnitude and a direction, then the vector divided by the magnitude (the norm) should equal a direction (a unit vector), as shown in Figure 1.2.

$$\begin{tikzcd}
 & \mathbf{x} = \begin{pmatrix} 3 \\ -4 \end{pmatrix} \arrow[dl, "\text{norm}"'] \arrow[dr, "\text{unit vector}"] & \\
\|\mathbf{x}\| = 5 \arrow[ur, bend right, "\text{scalar multiplication}"'] & & \hat{\mathbf{x}} = \begin{pmatrix} 3/5 \\ -4/5 \end{pmatrix} \arrow[ul, bend left]
\end{tikzcd}$$

Figure 1.2: Vectors separate into a length (norm) and direction (unit vector). The length and direction can be combined by scalar multiplication

1.7 Scalar Vector Multiplication

We saw earlier that elementwise multiplication was a terrible idea. In fact, defining multiplication this way violates a corollary of the field axioms ($\mathbf{xy} = \mathbf{0}$ implies that $\mathbf{x} = \mathbf{0}$ or $\mathbf{y} = \mathbf{0}$). However, elementwise multiplication does work in one case—*scalar* *multiplication*, or the product between a scalar (real number) and a vector:

$$k\mathbf{x} = k\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} kx_1 \\ kx_2 \\ \vdots \\ kx_n \end{pmatrix}$$

where k is a scalar real number. Notice that scalar multiplication does not suffer from the same problem as elementwise vector multiplication. If $k\mathbf{x} = 0$, then either the scalar k equals zero or the vector $\mathbf{x}$ must be the zero vector.

What happens when you multiply a vector by a scalar? For one, the norm changes:

Remember that $\sqrt{k^2} = |k|$, not k itself. We consider the square root to be the positive root.

$$\begin{aligned} \|k\mathbf{x}\| &= \sqrt{(kx_1)^2 + (kx_2)^2 + \cdots + (kx_n)^2} \\ &= \sqrt{k^2(x_1^2 + x_2^2 + \cdots + x_n^2)} \\ &= |k|\,\|\mathbf{x}\| \end{aligned}$$

Scalar multiplication scales the length of a vector by the scalar. If the scalar is negative, the direction of the vector "reverses".

1.8 Dot (Inner) Product

Now we see why we use the symbol $\times$ for multiplication; the dot $(\cdot)$ is reserved for the dot product.

One way to think of the product of two vectors is to consider the product of their norms (magnitudes). Such operations are common in mechanics. Work, for example, is the product of force and displacement. However, simply multiplying the magnitude of the force vector and the magnitude of the displacement vector disregards the orientation of the vectors. We know from physics that only the component of the force aligned with the displacement should count.

In general, we want an operation that multiplies the magnitude of one vector with the *projection* of a second vector onto the first. We call this operation the *inner product* or the *dot product*. Geometrically, the dot product is a measure of both the product of the vectors' magnitudes and how well they are aligned. For vectors $\mathbf{x}$ and $\mathbf{y}$ the dot product is defined

$$\mathbf{x} \cdot \mathbf{y} = \|\mathbf{x}\|\,\|\mathbf{y}\| \cos\theta$$

where θ is the angle between the vectors.

If two vectors are perfectly aligned, $\theta = 0°$ and the dot product is simply the product of the magnitudes. If the two vectors point in exactly opposite directions,

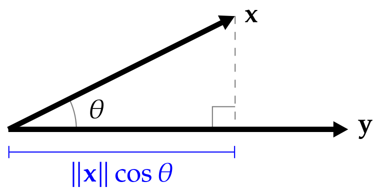


Figure 1.3: The projection of $\mathbf{x}$ onto $\mathbf{y}$ is a scalar equal to $\|\mathbf{x}\| \cos\theta$.

$\theta = 180°$ and the dot product is -1 times the product of the magnitudes. If the vectors are *orthogonal*, the angle between them is 90°, so $\cos\theta = 0$ and the dot product is zero. Thus, **the dot product of two vectors is zero if and only if the vectors are orthogonal**.

1.8.1 Computing the Dot Product

We know how to calculate norms, but how do we calculate the angle between two n-dimensional vectors? The answer is that we don't need to. There is an easier way to calculate $\mathbf{x} \cdot \mathbf{y}$ than the formula $\|\mathbf{x}\| \, \|\mathbf{y}\| \cos\theta$.

First, we need to define a special set of vectors—the unit vectors $\hat{\mathbf{e}}_i$. Unit vectors have only a single nonzero entry, a 1 at element i. For example,

$$\hat{\mathbf{e}}_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \hat{\mathbf{e}}_2 = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}, \ldots, \hat{\mathbf{e}}_n = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}.$$

Every vector can be written as a sum of scalar products with unit vectors. For example,

$$\begin{aligned} \begin{pmatrix} -3 \\ 6 \\ 2 \end{pmatrix} &= -3 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + 6 \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + 2 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \\ &= -3\hat{\mathbf{e}}_1 + 6\hat{\mathbf{e}}_2 + 2\hat{\mathbf{e}}_3 \end{aligned}$$

In general

$$\begin{aligned} \mathbf{x} &= x_1 \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + x_2 \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} + \cdots + x_n \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix} \\ &= \sum_{i=1}^{n} x_i \hat{\mathbf{e}}_i \end{aligned}$$

Now let's compute the dot product using the unit vector expansion for **x** and **y**.

$$\begin{aligned}\mathbf{x}\cdot\mathbf{y} =& \ (x_1\hat{\mathbf{e}}_1 + x_2\hat{\mathbf{e}}_2 + \cdots + x_n\hat{\mathbf{e}}_n)\cdot\left(y_1\hat{\mathbf{e}}_1 + y_2\hat{\mathbf{e}}_2 + \cdots + y_n\hat{\mathbf{e}}_n\right)\\ =& \ x_1\hat{\mathbf{e}}_1\cdot(y_1\hat{\mathbf{e}}_1 + y_2\hat{\mathbf{e}}_2 + \cdots + y_n\hat{\mathbf{e}}_n)\\ &+ x_2\hat{\mathbf{e}}_2\cdot(y_1\hat{\mathbf{e}}_1 + y_2\hat{\mathbf{e}}_2 + \cdots + y_n\hat{\mathbf{e}}_n)\\ &+\cdots\\ &+ x_n\hat{\mathbf{e}}_n\cdot(y_1\hat{\mathbf{e}}_1 + y_2\hat{\mathbf{e}}_2 + \cdots + y_n\hat{\mathbf{e}}_n)\end{aligned}$$

Consider each of the terms $x_i\hat{\mathbf{e}}_i\cdot(y_1\hat{\mathbf{e}}_1 + y_2\hat{\mathbf{e}}_2 + \cdots + y_n\hat{\mathbf{e}}_n)$. By distribution, this is equivalent to

$$x_iy_1\hat{\mathbf{e}}_i\cdot\hat{\mathbf{e}}_1 + \cdots + x_iy_j\hat{\mathbf{e}}_i\cdot\hat{\mathbf{e}}_j + \cdots + x_iy_n\hat{\mathbf{e}}_i\cdot\hat{\mathbf{e}}_n.$$

The only nonzero term in this entire summation is $x_iy_i\hat{\mathbf{e}}_i\cdot\hat{\mathbf{e}}_i$, which equals x_iy_i. The dot product reduces to

Think about the dot product $\hat{\mathbf{e}}_i\cdot\hat{\mathbf{e}}_j$. If $i = j$, this product is 1 since $\|\hat{\mathbf{e}}_i\| = \left\|\hat{\mathbf{e}}_j\right\| = 1$ and $\theta = 0^\circ$. However, if $i \neq j$, the vectors are always orthogonal and the dot product is 0.

$$\begin{aligned}\mathbf{x}\cdot\mathbf{y} &= x_1y_1 + x_2y_2 + \cdots + x_ny_n\\ &= \sum_{i=1}^{n} x_iy_i\end{aligned}$$

Although the above formula is convenient for computing dot products, it lacks the intuition of our previous method ($\mathbf{x}\cdot\mathbf{y} = \|\mathbf{x}\|\,\|\mathbf{y}\|\cos\theta$). Whenever you use the former method, be sure to remember that you're really calculating the product magnitudes after one vector is projected onto the other.

1.8.2 Dot Product Summary

- Dot products are defined between two vectors with the same dimension.
- Dot products return a scalar from two vectors. This is the projected product of the two magnitudes.
- $\mathbf{x}\cdot\mathbf{y} = \|\mathbf{x}\|\,\|\mathbf{y}\|\cos\theta = x_1y_1 + x_2y_2 + \cdots + x_ny_n$
- $\mathbf{x}\cdot\mathbf{y} = 0 \Leftrightarrow$ **x** and **y** are orthogonal.
- Dot products commute [$\mathbf{x}\cdot\mathbf{y} = \mathbf{y}\cdot\mathbf{x}$] and distribute over addition [$\mathbf{x}\cdot(\mathbf{y}+\mathbf{z}) = \mathbf{x}\cdot\mathbf{y}+\mathbf{x}\cdot\mathbf{z}$]. There is no such thing as an associative property for dot products since the dot product of three vectors is not defined. The dot product between the first two vectors would produce a scalar, and the dot product between this scalar and the third vector is not defined.

Chapter 2

Matrices

2.1 Matrix/Vector Multiplication

Let's take stock of the operations we've defined so far.

- The **norm** (magnitude) maps a vector to a scalar. ($\mathbb{R}^n \mapsto \mathbb{R}$)
- The **scalar product** maps a scalar and a vector to a new vector ($\mathbb{R} \times \mathbb{R}^n \mapsto \mathbb{R}^n$), but can only scale the magnitude of the vector (or flip it if the scalar is negative).
- The **dot product** maps to vectors to a scalar ($\mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$) by projecting one onto the other and multiplying the resulting magnitudes.

All of these operations appear consistent with the field axioms. Unfortunately, we still do not have a true multiplication operation—one that can transform any vector into any other vector. Can we define such an operation using only the above methods?

Let's construct a new vector **y** from the vector **x**. To be as general as possible we should let each element in **y** be an arbitrary linear combination of the elements in **x**:

$$\begin{aligned}
y_1 &= a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\
y_2 &= a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\
&\vdots \\
y_n &= a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n
\end{aligned}$$

The scalars a_{ij} determine the relative weight of x_j when constructing y_i. There are n^2 scalars required to unambiguously map $\mathbf{x}$ to $\mathbf{y}$. For convenience, we collect the set of weights into an n by n numeric grid called a *matrix*.

If $\mathbf{A}$ is a real-valued matrix with dimensions $m \times n$, we say $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\dim(\mathbf{A}) = m \times n$.

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

Using the matrix $\mathbf{A}$, we can write the above system of equations as

$$\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

or, more succinctly

$$\mathbf{y} = \mathbf{A}\mathbf{x}.$$

Writing the equations in this *matrix form* requires a new definition of multiplication between the matrix $\mathbf{A}$ and the vector $\mathbf{x}$. For example, we can write the following linear system

$$\begin{aligned} x_1 - 2x_2 + x3 &= 0 \\ 3x_1 - x_3 &= 4 \\ x_2 + 3x_3 &= -1 \end{aligned}$$

in matrix form as

$$\begin{pmatrix} 1 & -2 & 1 \\ 3 & 0 & -1 \\ 0 & 1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 4 \\ -1 \end{pmatrix}.$$

These equations transform the input vector $\mathbf{x}$ into a new vector with elements 0, 4, and -1. The first element (0) is the dot product between the first row of the matrix and the vector:

$$\begin{pmatrix} 1 & -2 & 1 \\ 3 & 0 & -1 \\ 0 & 1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 4 \\ -1 \end{pmatrix}.$$

A subtle technical point: we consider a single row of a matrix to be a vector—in fact a column vector—to allow the dot product with the input vector. The distinction between row and column vectors is often glossed over by humans but is important to computers. (See below when we talk about the matrix transpose.)

Similarly, the second element in the output vector (4) is the dot product between the second row of the matrix and the input vector,

$$\begin{pmatrix} 1 & -2 & 1 \\ 3 & 0 & -1 \\ 0 & 1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 4 \\ -1 \end{pmatrix}$$

and the third entry in the output is the dot product between the third row and the input vector:

$$\begin{pmatrix} 1 & -2 & 1 \\ 3 & 0 & -1 \\ 0 & 1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 4 \\ -1 \end{pmatrix}.$$

2.2 Matrix Multiplication

What we have been calling "vectors" all along are really just matrices with only one column. Thinking of vectors as matrices lets us write a simple, yet powerful, definition of multiplication.

Definition. *The product of matrices* **AB** *is a matrix* **C***; each element* c_{ij} *in* **C** *is the dot product between the ith row in* **A** *and the jth column in* **B***:*

$$c_{ij} = \mathbf{A}(i,:) \cdot \mathbf{B}(:,j).$$

This definition is perfectly consistent with matrix/vector multiplication if we view a vector as a single-column matrix.

Example 2.1. Let's multiply the matrices

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & -2 \\ 4 & 3 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{B} = \begin{pmatrix} 0 & 1 \\ -2 & 3 \\ 1 & 1 \end{pmatrix}.$$

We'll call the resulting matrix **C**. The entry c_{11} is the dot product between row 1 of matrix **A** and column 1 of matrix **B**.

$$\begin{pmatrix} -2 & \\ & \end{pmatrix} = \begin{pmatrix} 1 & 0 & -2 \\ 4 & 3 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -2 & 3 \\ 1 & 1 \end{pmatrix}$$

$$c_{11} = \mathbf{A}(1,:) \cdot \mathbf{B}(:,1) = 1 \times 0 + 0 \times (-2) + -2 \times 1 = -2$$

The entry c_{12} is the dot product between row 1 of matrix **A** and column 2 of matrix **B**.

$$\begin{pmatrix} -2 & -1 \\ & \end{pmatrix} = \begin{pmatrix} 1 & 0 & -2 \\ 4 & 3 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -2 & 3 \\ 1 & 1 \end{pmatrix}$$

$$c_{12} = \mathbf{A}(1,:) \cdot \mathbf{B}(:,2) = 1 \times 1 + 0 \times 3 + -2 \times 1 = -1$$

The entry c_{21} is the dot product between row 2 of matrix **A** and column 1 of matrix **B**.

$$\begin{pmatrix} -2 & -1 \\ -5 & \end{pmatrix} = \begin{pmatrix} 1 & 0 & -2 \\ 4 & 3 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -2 & 3 \\ 1 & 1 \end{pmatrix}$$

$$c_{21} = \mathbf{A}(2,:) \cdot \mathbf{B}(:,1) = 4 \times 0 + 3 \times (-2) + 1 \times 1 = -5$$

Finally, the entry c_{22} is the dot product between row 2 of matrix **A** and column 2 of matrix **B**.

$$\begin{pmatrix} -2 & -1 \\ -5 & 14 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -2 \\ 4 & 3 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -2 & 3 \\ 1 & 1 \end{pmatrix}$$

$$c_{22} = \mathbf{A}(2,:) \cdot \mathbf{B}(:,2) = 4 \times 1 + 3 \times 3 + 1 \times 1 = 14$$

■

In the previous example, the matrices **A** and **B** did not have the same dimensions. If **C** = **AB**, each entry in the matrix **C** is the dot product between a row of **A** and a column of **B**. Thus the number of columns in **A** must equal the number of rows in **B** since the dot product is only defined for vectors of the same dimension.

The length of a *row* in **A** is equal to the number of *columns* in **A**. The length of a single *column* in **B** is equal to the number of *rows* in **B**.

Any matrices **A** and **B** are *conformable* for multiplication if the number of columns in **A** matches the number of rows in **B**. If the dimensions of **A** are $m \times n$ and the dimensions of **B** are $n \times p$, then the product will be a matrix of dimensions $m \times p$. To check if two matrices are conformable, write the dimensions next to each other:

MATLAB returns an error that "matrix dimensions must agree" when multiplying nonconformable objects.

For the system $\mathbf{y} = \mathbf{Ax}$, if dim(**A**)=$m \times n$ and dim(**x**)=$n \times 1$, then dim(**y**) = $m \times 1$, i.e. **y** is a column vector in $\mathbb{R}^m$.

$$\overbrace{(m \times \underbrace{n)(n}_{\text{must agree}} \times p)}^{\text{dimensions of product}}$$

The inner two numbers (here n and n) must be the same. The dimensions of the product matrix are the outer two numbers (m and p).

Matrix multiplication is associative [**ABC**= (**AB**)**C**= **A**(**BC**)] and distributive over addition [**A**(**B**+**C**) = **AB**+**AC**] provided **A**, **B**, and **C** are all conformable. However, matrix multiplication is **not** commutative. To see why, consider an $(m \times n)$ matrix **A** and an $(n \times p)$ matrix **B**. The product **AB** is an $m \times p$ matrix, but the product **BA** is not conformable since $p \neq m$. Even if **BA** were conformable, it is not the same as the product **AB**:

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 0 & 1 \\ -1 & 2 \end{pmatrix}$$

$$\mathbf{AB} = \begin{pmatrix} 1\times 0 + 2\times(-1) & 1\times 1 + 2\times 2 \\ 3\times 0 + 4\times(-1) & 3\times 1 + 4\times 2 \end{pmatrix} = \begin{pmatrix} -2 & 5 \\ -4 & 11 \end{pmatrix}$$

$$\mathbf{BA} = \begin{pmatrix} 0\times 1 + 1\times 3 & 0\times 2 + 1\times 4 \\ -1\times 1 + 2\times 3 & -1\times 2 + 2\times 4 \end{pmatrix} = \begin{pmatrix} 3 & 4 \\ 5 & 6 \end{pmatrix}$$

2.3 Identity Matrix

We need to find an element that serves as **1** for vectors. The field axioms define this element by the property that $1 \times x = x$ for all x in the field. For vectors, we defined multiplication to involve matrices, so the element **1** will be a matrix that we call the *identity matrix*, or **I**. We require that

$$\mathbf{Ix} = \mathbf{xI} = \mathbf{x}$$

for all **x**. Assuming that **x** is n-dimensional, **I** must have n columns to be conformable. Also, the output of **Ix** has n elements, so **I** must have n rows. Therefore, we know that **I** is a square $n \times n$ matrix whenever **x** has dimension n.

Consider the first row of **I**, i.e. $\mathbf{I}(1,:)$. We know from the definition of **I** that $\mathbf{I}(1,:)\cdot\mathbf{x} = x_1$, so $\mathbf{I}(1,:) = (1\,0\,0\,\cdots\,0)$. For the second row, $\mathbf{I}(2,:)\cdot\mathbf{x} = x_2$, so $\mathbf{I}(2,:) = (0\,1\,0\,\cdots\,0)$. In general, the ith row of **I** has a 1 at position i and zeros everywhere else

$$\mathbf{I} = \begin{pmatrix} 1 & 0 & 0 & & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & & 0 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}.$$

The identity matrix **I** for a vector in $\mathbb{R}^n$ is an $n \times n$ matrix with ones along the diagonal and zeroes everywhere else.

In $\mathbb{R}^1$, $\mathbf{I} = (1)$, which behaves like the real number 1.

Our definition of the identity matrix also works for matrix multiplication. For any square matrix **A**

$$\mathbf{IA} = \mathbf{AI} = \mathbf{A}.$$

The identity matrix works on "both sides" only if the matrix **A** is square. If the matrix **A** were not square, we would need separate identify matrices for left and right multiplication so the dimensions are compatible for multiplication. Consider a nonsquare matrix $\mathbf{A}_{5\times 3}$ with five rows and three columns. It is still true that $\mathbf{I}_{5\times 5}\mathbf{A}_{5\times 3} = \mathbf{A}_{5\times 3}$ and $\mathbf{A}_{5\times 3}\mathbf{I}_{3\times 3} = \mathbf{A}_{5\times 3}$; however, notice how we used a 5×5 identify matrix for left multiplication and a 3×3 identity matrix for right multiplication.

2.4 Matrix Transpose

The transpose operator flips the rows and columns of a matrix. The element a_{ij} in the original matrix becomes element a_{ji} in the transposed matrix. The transpose operator is a superscript "T", as in $\mathbf{A}^\mathsf{T}$. A transposed matrix is reflected about a diagonal drawn from the upper left to the lower right corner.

Other notations for the matrix transpose include $\mathbf{A}^t$ and $\mathbf{A}'$. The latter is used in MATLAB.

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}^\mathsf{T} = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$$

Transposing an $m \times n$ matrix creates an $n \times m$ matrix.

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

Transposing a column vector creates a row vector, and vice versa.

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}^\mathsf{T} = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}, \quad \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}^\mathsf{T} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

Why do we introduce the matrix transpose now? There are connections between the dot product, matrix multiplication, and transposition.

2.4.1 Transposition and the Dot Product

For any vectors $\mathbf{x}$ and $\mathbf{y}$:

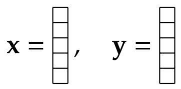
$$\mathbf{x} \cdot \mathbf{y} = \mathbf{x}^\mathsf{T}\mathbf{y}$$

The dot product between $\mathbf{x}$ and $\mathbf{y}$ is equivalent to the product between the transposed vector $\mathbf{x}$ and $\mathbf{y}$. Said simply, $\mathbf{x}$ "dot" $\mathbf{y}$ equals $\mathbf{x}^\mathsf{T}$ "times" $\mathbf{y}$.

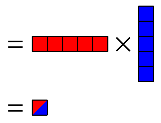
Notice that $\mathbf{x} \cdot \mathbf{y} \neq \mathbf{x}\mathbf{y}$ if we forget to transpose $\mathbf{x}$. Be careful to distinguish dot products and multiplication.

Why does $\mathbf{x} \cdot \mathbf{y} = \mathbf{x}^\mathsf{T}\mathbf{y}$? Computing the dot product between the vectors $\mathbf{x}$ and $\mathbf{y}$ requires both have the same dimension, which we'll call n. Both vectors are column vectors, so we can also view them as a single-column matrix with dimensions $n \times 1$. The transposed vector $\mathbf{x}^\mathsf{T}$ is therefore a row vector with dimensions $1 \times n$, and we know the product of an $1 \times n$ matrix with an $n \times 1$ matrix will be a 1×1 matrix—which is a scalar. By definition of matrix multiplication, $\mathbf{x}^\mathsf{T}\mathbf{y}$ is the dot product between the first row of $\mathbf{x}^\mathsf{T}$ (which has only one row) and the first column of $\mathbf{y}$

(which has only one column). Thus, $\mathbf{x}^\mathsf{T}\mathbf{y}$ is the dot product between vectors $\mathbf{x}$ and $\mathbf{y}$.



$$\mathbf{x} \cdot \mathbf{y} = \mathbf{x}^\mathsf{T}\mathbf{y}$$



We will use the relationship between dot products and matrix multiplication to solve linear statistical models later in this book. Until then, keep in mind the convergence between these operations.

2.4.2 Transposition and Matrix Multiplication

An interesting identity relates matrix multiplication and matrix transposition. For any matrices $\mathbf{A}$ and $\mathbf{B}$,

$$(\mathbf{AB})^\mathsf{T} = \mathbf{B}^\mathsf{T}\mathbf{A}^\mathsf{T}$$

Said in words, if we want to take the transpose of the product of two matrices, we could equivalently transpose the two matrices first and them multiply them **in reverse order**. The same idea holds when we transpose the product of several matrices. Consider a set of k matrices $\mathbf{A}_1$, $\mathbf{A}_2$, …, $\mathbf{A}_k$. Then

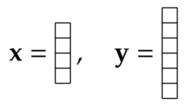
$$(\mathbf{A}_1\mathbf{A}_2 \cdots \mathbf{A}_{k-1}\mathbf{A}_k)^\mathsf{T} = \mathbf{A}_k^\mathsf{T}\mathbf{A}_{k-1}^\mathsf{T} \cdots \mathbf{A}_2^\mathsf{T}\mathbf{A}_1^\mathsf{T}.$$

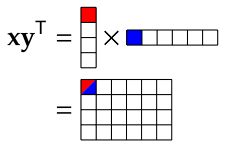
Notice two things about this identify. First, many students confuse this identify with the commutative property. As we've said before, matrix multiplication does not commute. The above identity is unrelated to the commutative property. Second, the identity holds for square and non-square matrices provided the matrices are compatible for multiplication. Try to prove to yourself that if the product $\mathbf{AB}$ is compatible, then the product $\mathbf{B}^\mathsf{T}\mathbf{A}^\mathsf{T}$ is also be compatible.

2.5 Outer Product and Trace

If the product $\mathbf{x}^\mathsf{T}\mathbf{y}$ is a scalar, what about the product $\mathbf{x}\mathbf{y}^\mathsf{T}$? Imagine $\mathbf{x}$ and $\mathbf{y}$ have n elements, making them $n \times 1$ matrices. (Both $\mathbf{x}$ and $\mathbf{y}$ must have the same number of elements or multiplication is impossible.) Then $\mathbf{x}\mathbf{y}^\mathsf{T}$ is the product of an $n \times 1$ matrix and a $1 \times n$ matrix. The product is therefore an $n \times n$ matrix. While $\mathbf{x}^\mathsf{T}\mathbf{y}$ and $\mathbf{x}\mathbf{y}^\mathsf{T}$ may look similar, they produce wildly different results. The former—the dot product—is also known as the *inner product* since it collapses to vectors into a scalar. The latter ($\mathbf{x}\mathbf{y}^\mathsf{T}$) is called the *outer product* since it expands vectors outward into a matrix. While the inner (dot) product requires both vectors have the same dimension, the outer product is compatible with any two vectors.

The matrix formed by the outer product contains all pairwise products between the elements in the two vectors.





There is an interesting connection between the outer and inner products. The *trace* of a square matrix is the sum of its diagonal elements. For example, the matrix

$$\mathbf{A} = \begin{pmatrix} -1 & 3 & 4 \\ 0 & 2 & -3 \\ 1 & 0 & 6 \end{pmatrix}$$

has trace

$$\operatorname{tr}(\mathbf{A}) = -1 + 2 + 6 = 7$$

The trace of a matrix $\mathbf{A}$ is written $\operatorname{tr}(\mathbf{A})$.

For a challenge, show that the trace of the outer product of two vectors is equal to the inner product of the same vectors, or

$$\operatorname{tr}(\mathbf{x}\mathbf{y}^\mathsf{T}) = \mathbf{x}^\mathsf{T}\mathbf{y}$$

Notice that this fact only works when the vectors $\mathbf{x}$ and $\mathbf{y}$ have the same dimension. Otherwise, the inner product is not defined. Similarly, if $\mathbf{x}$ and $\mathbf{y}$ did not have the same dimension, the outer product $\mathbf{x}\mathbf{y}^\mathsf{T}$ would not be a square matrix so the trace would not be defined.

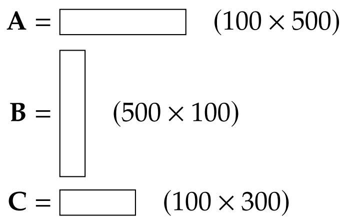
2.6 Computational Complexity of Matrix Multiplication

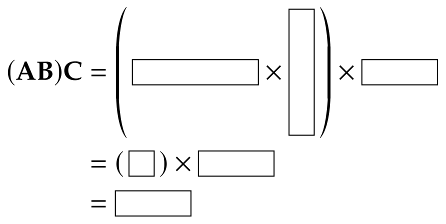
You may be feeling that matrix multiplication is tedious work. Indeed, multiplying two matrices requires computing many dot products, which themselves requires many scalar multiplications. In linear algebra it is important to understand how many operations are required to complete an operation, especially as the matrices become very large. The *computational complexity* of an operation is the relationship between the number of computations and the size of the operands. Let's analyze the computational complexity of matrix multiplication.

Imagine we're multiplying two $n \times n$ matrices. The product matrix will also have dimensions $n \times n$. Each entry in the product matrix is computed with a dot product between a row in the first factor and a column in the second factor. This dot product requires n scalar multiplications and $n-1$ additions. Overall, there are n^2 entries in the product matrix. If each requires a dot product, the total number of operations is $n^2 n$ multiplications and $n^2(n-1)$ additions. We say the number of operations required to multiply two $n \times n$ matrices is $\mathcal{O}(n^3)$. We can perform a similar analysis with two non-square matrices. Multiplying an $m \times n$ matrix by a $n \times p$ matrix requires $\mathcal{O}(mnp)$ operations.

The $\mathcal{O}$, or "big-O" notation indicates the rate of growth of a function for large values. Any polynomial of degree d is $\mathcal{O}(d)$.

The number of operations required to multiply three or more matrices depends on the order of the multiplication. Matrix multiplication is associative, so the product **ABC** can be computed as (**AB**)**C** or **A**(**BC**). Let's count the operations for both methods using the following sizes for **A**, **B**, and **C**.

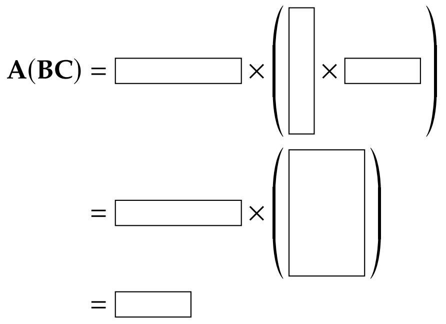




$100 \times 500 \times 100$ operations

$+\,100 \times 100 \times 300$ operations

$=$ **8,000,000 total operations**



$500 \times 100 \times 300$ operations

$+\,100 \times 500 \times 300$ operations

$=$ **30,000,000 total operations**

In general, the best order for matrix multiplications depends on the dimensions of the matrices. If the matrices **A**, **B**, **C** have dimensions

$$\begin{aligned} \dim(\mathbf{A}) &= m \times n \\ \dim(\mathbf{B}) &= n \times p \\ \dim(\mathbf{C}) &= p \times q \end{aligned}$$

it is more efficient to compute (**AB**)**C** if

$$mnp + mpq < npq + mnq$$

Otherwise, it is more efficient to compute **A**(**BC**).

Chapter 3

Rotation and Translation Matrices

A common problem in motion biomechanics is determining the position of a multi-bar linkage arm. For example, a two-bar linkage arm is shown in Figure 3.1. The arm contains two rigid bars of lengths l_1 and l_2. The bars are bent at angles θ_1 and θ_2. Given the bar lengths and the angles, can we calculate the position of the end of the arm (the point labeled **p** in Figure 3.1)?

Multi-bar linkages might seem like nightmarish geometry problems steeped in trigonometry; probably because they are. However, this chapter shows how matrix multiplication provides a straightforward, consistent method for solving problems with complex linkage arms. We will introduce two new operations: *rotation* and *translation*. Both rotation and translation can be described using matrix multiplication, and all linkage arms can be assembled from a series of rotations and translations.

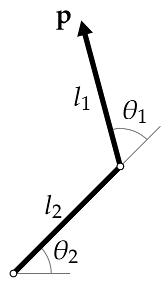


Figure 3.1: A two-bar linkage arm.

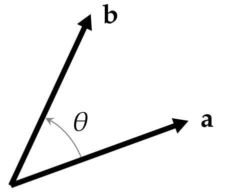


Figure 3.2: The vector **b** is equal to the vector **a** rotated counter-clockwise by the angle θ.

3.1 Rotation

We begin by rotating vectors as shown in Figure 3.2. We start with a vector that ends at the point **a**. If we rotate the vector about the origin (leaving its length the same), where is the new endpoint (called **b** in Figure 3.2)? Let's call the angle of rotation θ and define positive rotation ($\theta > 0$) to be in the counter-clockwise direction. The counter-clockwise definition is consistent with the righthand rule from physics. Now we can build a *rotation matrix* as follows:

$$\mathbf{R}(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

Given an angle θ, the rotation matrix $\mathbf{R}(\theta)$ is a 2×2 matrix containing sines and cosines of θ. Multiplying a vector by a rotation matrix $\mathbf{R}(\theta)$ is equivalent to rotating the vector by θ. Using the labels in Figure 3.2, the position after rotation is

$$\mathbf{b} = \mathbf{R}(\theta)\mathbf{a}$$

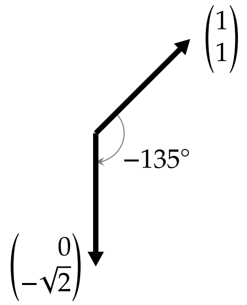
Let's do an example where we rotate the vector $\mathbf{a} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ by the angle $\theta = -135°$. Our first step is constructing the rotation matrix:

$$\mathbf{R}(-135°) = \begin{pmatrix} \cos(-135°) & -\sin(-135°) \\ \sin(-135°) & \cos(-135°) \end{pmatrix} = \begin{pmatrix} -\sqrt{2}/2 & \sqrt{2}/2 \\ -\sqrt{2}/2 & -\sqrt{2}/2 \end{pmatrix}$$

Now we can rotate the vector $\mathbf{a}$ by multiplying it by our rotation matrix.

$$\mathbf{R}(-135°)\mathbf{a} = \begin{pmatrix} -\sqrt{2}/2 & \sqrt{2}/2 \\ -\sqrt{2}/2 & -\sqrt{2}/2 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -\sqrt{2} \end{pmatrix}$$

The rotated vector points along the negative horizontal axis, as shown in Figure 3.3.



$\begin{pmatrix} 1 \\ 1 \end{pmatrix}$
$-135°$
$\begin{pmatrix} 0 \\ -\sqrt{2} \end{pmatrix}$

Figure 3.3: Negative angles rotate vectors in the clockwise direction.

3.1.1 Sequential Rotations

We can apply a sequence of rotations using successive multiplications by rotation matrices. Let's rotate a vector $\mathbf{x}$ by angle θ_1 followed by a second rotation of angle θ_2. The first rotation is the product $\mathbf{R}(\theta_1)\mathbf{x}$. Applying the second rotation gives $\mathbf{R}(\theta_2)\left(\mathbf{R}(\theta_1)\mathbf{x}\right)$. We can drop the parentheses by the associative property of matrix multiplication and write the result of both rotations as simply $\mathbf{R}(\theta_2)\mathbf{R}(\theta_1)\mathbf{x}$.

In general, we can rotate a vector $\mathbf{x}$ by k angles $\theta_1, \theta_2, \ldots, \theta_k$ with the expression

$$\mathbf{R}(\theta_k) \cdots \mathbf{R}(\theta_2)\mathbf{R}(\theta_1)\mathbf{x}$$

Pay attention to the order of the rotation matrices. The first rotation $\mathbf{R}(\theta_1)$ appears closest to the vector $\mathbf{x}$. The final rotation $\mathbf{R}(\theta_k)$ is farthest to the left side since it is applied last.

If the matrix $\mathbf{R}(\theta)$ rotates a vector by the angle θ, then the matrix $\mathbf{R}(-\theta)$ should undo the rotation since it rotates the same amount in the opposite direction. Indeed,

Remember that $\sin(-\theta) = -\sin\theta$, $\cos(-\theta) = \cos\theta$, and $\sin^2\theta + \cos^2\theta = 1$.

$$\begin{aligned}
\mathbf{R}(\theta)\mathbf{R}(-\theta) &= \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \cos(-\theta) & -\sin(-\theta) \\ \sin(-\theta) & \cos(-\theta) \end{pmatrix} \\
&= \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \\
&= \begin{pmatrix} \cos^2\theta + \sin^2\theta & \cos\theta\sin\theta - \sin\theta\cos\theta \\ \sin\theta\cos\theta - \cos\theta\sin\theta & \sin^2\theta + \cos^2\theta \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \mathbf{I}
\end{aligned}$$

so

$$\mathbf{R}(\theta)\mathbf{R}(-\theta)\mathbf{x} = \mathbf{I}\mathbf{x} = \mathbf{x}$$

Also, notice that $\mathbf{R}(-\theta)$ is the transpose of $\mathbf{R}(\theta)$. We will explore these special properties of rotation matrices in the coming chapters.

3.2 Translation

The second operation performed by linkage arms is translation, or shifting a point along each axis. In two dimensions, a point can be translated to a new location by shifting it a distance Δx along the horizontal axis and a distance Δy along the vertical axis (see Figure 3.4). Our goal is to find a *translation matrix* $\mathbf{T}(\Delta x, \Delta y)$ that can translate vectors with matrix multiplication. Using the labels in Figure 3.4, we are looking for a matrix $\mathbf{T}(\Delta x, \Delta y)$ such that

$$\mathbf{b} = \mathbf{T}(\Delta x, \Delta y)\mathbf{a}$$

Unlike rotation, translation cannot be easily expressed using matrix multiplication. Translation adds a constant to a vector, and this is not a linear operation. Recall from Section 1.5 that adding a constant is an affine operation (like the function $y = x + 3$). Matrix multiplication is a strictly linear operation. For two-dimensional vectors there is no 2×2 matrix that behaves like a translation matrix.

The good news is that we can cheat. We know how to rotate vectors, and translation in two dimensions is equivalent to a rotation in three dimensions. Consider the two dimensional plane shown in Figure 3.5. Our goal is to translate from point **a** on the plane to point **b**, which is also on the plane. If we look beyond the plane and into the third dimension, we see that a vector that points to **a** can be rotated to point to **b**. (Depending on the orientation of the plane, we might need to change the length of the three dimensional vector; this is easily done by scalar multiplication, which is a linear operation.)

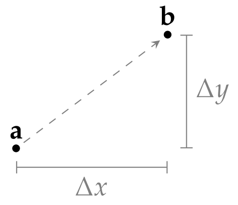


Figure 3.4: Translating point **a** to point **b** includes a horizontal shift (Δx) and a vertical shift (Δy).

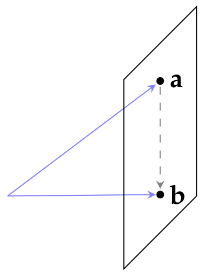


Figure 3.5: Translating point **a** to point **b** in two dimensions is equivalent to rotating the blue vectors in three dimensions.

It is possible to define a 3×3 translation matrix that shifts both the horizontal and vertical dimensions of a vector. The translation matrix is

$$\mathbf{T}(\Delta x, \Delta y) = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix}$$

The 3×3 translation matrix is designed to translate in two dimensions. For two dimensional vectors to be compatible we need to add a third, or "dummy" dimension that contains the value 1. For example, the two dimensional vector $\begin{pmatrix} x \\ y \end{pmatrix}$ becomes the three dimensional dummy vector $\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$. We can use dummy vectors to prove that the translation matrix does, in fact, translate:

$$\mathbf{T}(\Delta x, \Delta y)\mathbf{x} = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + \Delta x \\ y + \Delta y \\ 1 \end{pmatrix}$$

Notice how the translation matrix regenerates the value 1 in the dummy dimension. The value of the dummy dimension should never change upon either translation or rotation. It is an arbitrary offset, analogous to the distance to the rotation point in the dummy dimension (i.e. the distance between the plane and the starting point of the blue vectors in Figure 3.5).

3.2.1 Combining Rotation and Translation

Unlike translation, rotation in two dimensions did not require a dummy dimension. We can add a dummy dimension to the rotation matrix to make it compatible with translation and dummy vectors. The dummy version of the rotation matrix is

$$\mathbf{R}(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The dummy rotation matrix also regenerates the value 1 in the dummy dimension. Using dummy dimensions makes rotation matrices and translation matrices compatible for multiplication. In the next section we will combine rotation and translation to analyze multi-bar linkage arms.

3.3 Multi-bar Linkage Arms

Rotation and translation provide all the tools needed to analyze multi-bar linkage arms like the one shown in Figure 3.1. Pay attention two details in Figure 3.1. First, the segments and angles are numbered starting from the far end of the arm, so the far end of segment 1 is the endpoint of the arm. Second, the angles are defined as counter-clockwise rotations relative to a line that extends from the previous segment. For example, the first angle (θ_1) measures the rotation of segment 1 starting if segment 1 were perfectly in line with segment 2. Bearing these two details in mind, we now show how to calculate the final position of the linkage arm using sequential translations and rotations.

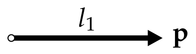
We begin with the end of the arm (point **p**) at the origin:

$$\mathbf{p} = \mathbf{0}_\mathrm{d}$$

We use a subscript "d" to remind us that $\mathbf{0}_\mathrm{d}$ is not a true zero vector, but rather a dummy vector with the value 1 in the final dimension.

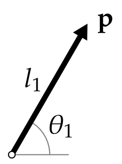
Then we translate **p** horizontally by the length of the first arm (l_1).

$$\mathbf{p} = \mathbf{T}(l_1, 0)\mathbf{0}_\mathrm{d}$$



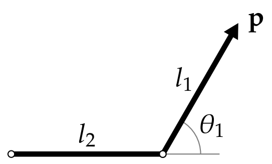
Next we rotate the first segment by the first angle (θ_1).

$$\mathbf{p} = \mathbf{R}(\theta_1)\mathbf{T}(l_1, 0)\mathbf{0}_\mathrm{d}$$



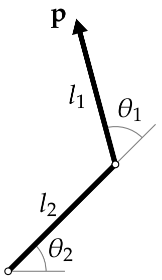
The first arm is translated horizontally by the length of the second arm (l_2).

$$\mathbf{p} = \mathbf{T}(l_2, 0)\mathbf{R}(\theta_1)\mathbf{T}(l_1, 0)\mathbf{0}_\mathrm{d}$$



Finally we rotate both arms by the second angle (θ_2).

$$\mathbf{p} = \mathbf{R}(\theta_2)\mathbf{T}(l_2, 0)\mathbf{R}(\theta_1)\mathbf{T}(l_1, 0)\mathbf{0}_\mathrm{d}$$



Let's do an example where $l_1 = l_2 = 3$ cm, $\theta_1 = 30°$, and $\theta_2 = 45°$. The position of the end of the arm is

$$\begin{aligned}
\mathbf{p} &= \mathbf{R}(45°)\mathbf{T}(3,0)\mathbf{R}(30°)\mathbf{T}(3,0)\mathbf{0}_\mathrm{d} \\
&= \begin{pmatrix} \sqrt{2}/2 & -\sqrt{2}/2 & 0 \\ \sqrt{2}/2 & \sqrt{2}/2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \sqrt{3}/2 & -1/2 & 0 \\ 1/2 & \sqrt{3}/2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} 0.259 & -0.966 & 2.90 \\ 0.966 & 0.259 & 5.02 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \\
&= \begin{pmatrix} 2.90 \\ 5.02 \\ 1 \end{pmatrix}
\end{aligned}$$

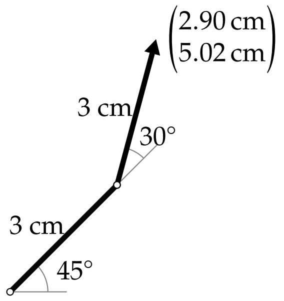

$\begin{pmatrix} 2.90\,\text{cm} \\ 5.02\,\text{cm} \end{pmatrix}$
3 cm
30°
3 cm
45°

Figure 3.6: Example two-bar linkage arm.

The final position of the arm is 2.90 cm along the horizontal and 5.02 cm along the vertical, as shown in Figure 3.6.

3.4 Rotating Shapes

The matrix formalism for rotation and translation naturally extends to shapes, paths, or other collections of points. Consider the triangle shown in Figure 3.7 with vertices at $(1, 0)$, $(2, 0)$, and $(2, 1)$. We can represent the triangle by collecting the vertices in a matrix with one column for each vertex:

$$\mathbf{X} = \begin{pmatrix} 1 & 2 & 2 \\ 0 & 0 & 1 \end{pmatrix}.$$

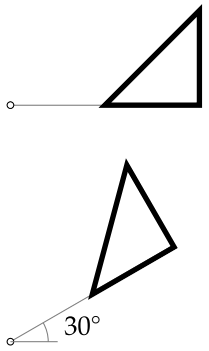


Figure 3.7: Rotating a triangle.

We can rotate counter-clockwise by 30° with the rotation matrix

$$\mathbf{R}(30°) = \begin{pmatrix} \cos 30° & -\sin 30° \\ \sin 30° & \cos 30° \end{pmatrix}.$$

We are not translating the shape—only rotating it—so there is no need for a dummy dimension in this case. The translated shape is the product of the rotation matrix and the matrix of vertices.

$$\begin{aligned}\mathbf{R}(30^\circ)\mathbf{X} &= \begin{pmatrix} \cos 30^\circ & -\sin 30^\circ \\ \sin 30^\circ & \cos 30^\circ \end{pmatrix} \begin{pmatrix} 1 & 2 & 2 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 0.866 & 1.732 & 1.232 \\ 0.500 & 1 & 1.866 \end{pmatrix}\end{aligned}$$

The new vertices are $(0.866, 0.500)$, $(1.732, 1)$, and $(1.232, 1.866)$. The matrix formalism allowed us to rotate all three points simultaneously. Computer graphics software uses rotation and translation matrices to efficiently manipulate shapes by matrix multiplication.

Chapter 4

Solving Linear Systems

Remember back to algebra when you were asked to solve small systems of equations like

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 &= y_1 \\ a_{21}x_1 + a_{22}x_2 &= y_2 \end{aligned}$$

Your strategy was to manipulate the equations until they reached the form

$$\begin{aligned} x_1 &= y_1' \\ x_2 &= y_2' \end{aligned}$$

We often use the prime symbol ($'$) to indicate that an unspecified new value is based on an old one. For example, y_1' is a new value calculated from y_1. In this case,

$$y_1' = \frac{y_1 a_{22} - a_{12} y_2}{a_{11}a_{22} - a_{12}a_{21}}$$

In matrix form, this process transforms the coefficient matrix **A** into the identity matrix

$$\begin{pmatrix} a_{11} & a_{21} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} y_1' \\ y_2' \end{pmatrix}$$

This leads us to our first strategy for solving linear systems of the form **Ax**= **y**. We manipulate both sides of the equation (**A** and **y**) until **A** becomes the identity matrix. The vector **x** then equals the transformed vector **y**′. Because we will be applying the same transformations to both **A** and **y**, it is convenient to collect them both into an *augmented matrix* $\begin{pmatrix} \mathbf{A} & \mathbf{y} \end{pmatrix}$. For 2×2 system above, the augmented matrix is

$$\begin{pmatrix} a_{11} & a_{12} & y_1 \\ a_{21} & a_{22} & y_2 \end{pmatrix}$$

What operations can we use to transform **A** into the identity matrix? There are three operations, called the *elementary row operations*, or EROs:

1. **Exchanging two rows.** Since the order of the equations in our system is arbitrary, we can re-order the rows of the augmented matrix at will. By working with the augmented matrix we ensure that both the left- and right-hand sides move together.

$$\begin{pmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{pmatrix} \xrightarrow{R_2 \leftrightarrow R_3} \begin{pmatrix}1 & 2 & 3\\ 7 & 8 & 9\\ 4 & 5 & 6\end{pmatrix}$$

2. **Multiplying any row by a scalar.** Again, since we are working with the augmented matrix, multiplying a row by a scalar multiplies both the left- and right-hand sides of the equation by the same factor.

$$\begin{pmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{pmatrix} \xrightarrow{3R_2} \begin{pmatrix}1 & 2 & 3\\ 12 & 15 & 18\\ 7 & 8 & 9\end{pmatrix}$$

3. **Adding a scalar multiple of any row to any other row.** We use the notation $kR_i \longrightarrow R_j$ to denote multiplying row R_i by the scalar k and adding this scaled row to row R_j. For example:

$$\begin{pmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{pmatrix} \xrightarrow{3R_2 \rightarrow R_1} \begin{pmatrix}13 & 17 & 21\\ 4 & 5 & 6\\ 7 & 8 & 9\end{pmatrix}$$

Let's solve the following system of equations using elementary row operations.

$$\begin{aligned} 4x_1 + 8x_2 - 12x_3 &= 44\\ 3x_1 + 6x_2 - 8x_3 &= 32\\ -2x_1 - x_2 &= -7 \end{aligned}$$

This is a linear system of the form **Ax**= **y** where the coefficient matrix **A** and the vector **y** are

$$\mathbf{A} = \begin{pmatrix}4 & 8 & -12\\ 3 & 6 & -8\\ -2 & -1 & 0\end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix}44\\ 32\\ -7\end{pmatrix}$$

The augmented matrix is therefore

$$\begin{pmatrix}4 & 8 & -12 & 44\\ 3 & 6 & -8 & 32\\ -2 & -1 & 0 & -7\end{pmatrix}$$

Now we apply the elementary row operations.

$$
\xrightarrow{\frac{1}{4}R_1}
\begin{pmatrix}
1 & 2 & -3 & 11 \\
3 & 6 & -8 & 32 \\
-2 & -1 & 0 & -7
\end{pmatrix}
$$
$$
\xrightarrow{-3R_1 \to R_2}
\begin{pmatrix}
1 & 2 & -3 & 11 \\
0 & 0 & 1 & -1 \\
-2 & -1 & 0 & -7
\end{pmatrix}
$$
$$
\xrightarrow{2R_1 \to R_3}
\begin{pmatrix}
1 & 2 & -3 & 11 \\
0 & 0 & 1 & -1 \\
0 & 3 & -6 & 15
\end{pmatrix}
$$

Notice that after three steps we have a zero at position (2,2). We need to move this row farther down the matrix to continue; otherwise we can't cancel out the number 3 below it. This operation is called a *pivot*.

$$
\xrightarrow{R_2 \leftrightarrow R_3}
\begin{pmatrix}
1 & 2 & -3 & 11 \\
0 & 3 & -6 & 15 \\
0 & 0 & 1 & -1
\end{pmatrix}
$$
$$
\xrightarrow{\frac{1}{3}R_2}
\begin{pmatrix}
1 & 2 & -3 & 11 \\
0 & 1 & -2 & 5 \\
0 & 0 & 1 & -1
\end{pmatrix}
$$

At this point we have a matrix in *row echelon form*. The bottom triangle looks like the identity matrix. We could stop here and solve the system using back substitution:

$$
\begin{aligned}
x_3 &= -1 \\
x_2 + -2(-1) = 5 \Rightarrow\ x_2 &= 3 \\
x_1 + 2(3) - 3(-1) = 11 \Rightarrow\ x_1 &= 2
\end{aligned}
$$

Or, we could keep going and place the augmented matrix into *reduced row echelon* *form*.

$$\xrightarrow{-2R_2 \to R_1} \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & -2 & 5 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$
$$\xrightarrow{-R_3 \to R_1} \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & -2 & 5 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$
$$\xrightarrow{2R_3 \to R_2} \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

The left three columns are the identity matrix, so the resulting system of equations has been simplified to

$$\begin{aligned} x_1 &= 2 \\ x_2 &= 3 \\ x_3 &= -1 \end{aligned}$$

4.1 Gaussian Elimination

Using EROs to transform the augmented matrix into the identity matrix is called *Gaussian elimination*. Let's develop an algorithm for Gaussian elimination for a general system of equations $\mathbf{A}\mathbf{x} = \mathbf{y}$ when $\mathbf{A}$ is an $n \times n$ coefficient matrix. We begin with the augmented matrix

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & y_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & y_2 \\ \vdots & & \ddots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & y_n \end{pmatrix}$$

We need the number 1 in the a_{11} position.

$$\xrightarrow{a_{11}^{-1} R_1} \begin{pmatrix} 1 & a_{11}^{-1} a_{12} & \cdots & a_{11}^{-1} a_{1n} & a_{11}^{-1} y_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & y_2 \\ \vdots & & \ddots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & y_n \end{pmatrix}$$

Now we zero out the a_{21} position using the first row multiplied by $-a_{21}$.

$$\xrightarrow{-a_{21}R_1 \to R_2} \begin{pmatrix} 1 & a_{11}^{-1}a_{12} & \cdots & a_{11}^{-1}a_{1n} & a_{11}^{-1}y_1 \\ 0 & a_{22} - a_{21}a_{11}^{-1}a_{12} & \cdots & a_{2n} - a_{21}a_{11}^{-1}a_{1n} & y_2 - a_{21}a_{11}^{-1}y_1 \\ \vdots & & \ddots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & y_n \end{pmatrix}$$

We keep zeroing out the entries a_{31} through a_{n1} using the first row. We end up with the matrix

$$\xrightarrow{-a_{n1}R_1 \to R_n} \begin{pmatrix} 1 & a_{11}^{-1}a_{12} & \cdots & a_{11}^{-1}a_{1n} & a_{11}^{-1}y_1 \\ 0 & a_{22} - a_{21}a_{11}^{-1}a_{12} & \cdots & a_{2n} - a_{21}a_{11}^{-1}a_{1n} & y_2 - a_{21}a_{11}^{-1}y_1 \\ \vdots & & \ddots & & \vdots \\ 0 & a_{n2} - a_{n1}a_{11}^{-1}a_{12} & \cdots & a_{nn} - a_{n1}a_{11}^{-1}a_{1n} & y_n - a_{n1}a_{11}^{-1}y_1 \end{pmatrix}$$

This is looking a little complicated, so let's rewrite the matrix as

$$\begin{pmatrix} 1 & a'_{12} & \cdots & a'_{1n} & y'_1 \\ 0 & a'_{22} & \cdots & a'_{2n} & y'_2 \\ \vdots & & \ddots & & \vdots \\ 0 & a'_{n2} & \cdots & a'_{nn} & y'_n \end{pmatrix}$$

The first column looks like the identity matrix, which is exactly what we want. Our next goal is to put the lower half of an identify matrix in the second column by setting $a'_{22} = 1$ and $a_{32}, \ldots, a_{n2} = 0$. Notice that this is the same as applying the above procedure to the sub-matrix

$$\begin{pmatrix} a'_{22} & \cdots & a'_{2n} & y'_2 \\ \vdots & \ddots & & \vdots \\ a'_{n2} & \cdots & a'_{nn} & y'_n \end{pmatrix}$$

After that, we can continue recursively until the left part of the augmented matrix is the identity matrix. We can formalize the Gaussian elimination algorithm as follows:

```
function GAUSSIAN ELIMINATION
    for $j = 1$ to $n$ do                          ▷ For every column
        $a_{jj}^{-1} R_j$                          ▷ Set the element on the diagonal to 1
        for $i = j+1$ to $n$ do                    ▷ For every row below the diagonal
            $-a_{ij} R_j \to R_i$                  ▷ Zero the below-diagonal element
        end for
    end for
end function
```

We're ignoring pivoting here. In general, we need to check that $a_{jj} \neq 0$; if it is, we swap the row for one below that has a nonzero term in the jth column.

4.2 Computational Complexity of Gaussian Elimination

How many computational operations are needed to perform Gaussian elimination on an $n \times n$ matrix? Let's start by counting operations when reducing the first column. Scaling the first row ($a_{11}^{-1} R_1$) requires n operations. (There are $n+1$ entries in the first row of the augmented matrix if you include the value y_1; however, we know the result in the first column, $a_{11}^{-1} a_{11}$, will always equal the number 1; we don't need to compute it.) Similarly, zeroing out a single row below requires n multiplications and n subtractions. (Again, there are $n+1$ columns, but we know the result in column 1 will be zero.) In the first column, there are $n-1$ rows below to zero out, so the total number of operations is

$$\underbrace{n}_{a_{11}^{-1} R_1} + \underbrace{2(n-1)n}_{-a_{i1} R_1 \to R_i} = 2n^2 - n$$

After we zero the bottom of the first row, we repeat the procedure on the $(n-1) \times (n-1)$ submatrix, and so on until we reach the 1×1 submatrix that includes only a_{nn}. We add up the number of operations for each of these n submatrices

$$\begin{aligned}
\text{total operations} &= \sum_{k=1}^{n} (2k^2 - k) \\
&= 2\sum_{k=1}^{n} k^2 - \sum_{k=1}^{n} k \\
&= \frac{n(n+1)(2n+1)}{3} - \frac{n(n+1)}{2} \\
&= \mathcal{O}(n^3)
\end{aligned}$$

Remember that

$$\sum_{k=1}^{n} 1 = n$$

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

$$\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$$

Thus the number of operations required to bring an $n \times n$ matrix into row-echelon form is on the order of n^3. The number of operations needed to perform back substitution and solve the system is $O(n^2)$. This raises two important points.

1. Gaussian elimination scales cubically. A system with twice as many equations takes eight times longer to solve.
2. The computational bottleneck is generating the row echelon matrix. Back substitution (or creating the reduced row echelon matrix) is significantly faster.

4.3 Solving Linear Systems in Matlab

Matlab has multiple functions for solving linear systems. Given variables `A` and `y`, you can use

- `linsolve(A,y)` to solve using LU decomposition, a variant of Gaussian elimination.
- `(A \ y)` to let Matlab choose the best algorithm based on the size and structure of `A`.
- `rref([A y])` to compute the reduced row echelon form of the augmented matrix.

Chapter 5

The Finite Difference Method

When you first learned about derivatives, they were probably introduced as the limit of a finite difference

$$\frac{du}{dx} = \lim_{a \to b} \frac{u(b) - u(a)}{b - a}$$

As the distance between points a and b shrinks to zero, the finite difference becomes infinitesimal. The resulting differential equations must be solved with integral calculus. This chapter presents an alternative, numerical method for solving differential equations. Rather than shrink the finite differences all the way to zero, we leave a small but finite gap. The resulting algebraic equations approximate the differential equation and can be solved without any tools from calculus.

5.1 Finite Differences

Solving a differential equation analytically yields a solution over the entire domain (the region between the boundary conditions). By contrast, numerical methods find solutions to a differential equation at only a discrete set of points, or *nodes*, in the domain. The derivatives in the differential equation are descretized by converting them into *finite differences*. For example, we can approximate a first derivative as the change between two nodes divided by the distance between the nodes:

$$\frac{du}{dx} \approx \frac{u^{(k+1)} - u^{(k)}}{\Delta x}$$

where $u^{(k)}$ is the value of the variable at the kth node. To approximate a second derivative, we calculate the finite difference between nodes $u^{(k+1)}$ and $u^{(k)}$; and

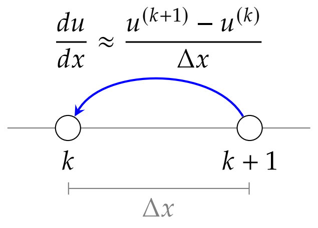

$\frac{du}{dx} \approx \frac{u^{(k+1)} - u^{(k)}}{\Delta x}$
k
$k+1$
Δx

Figure 5.1: First-order finite difference approximation.

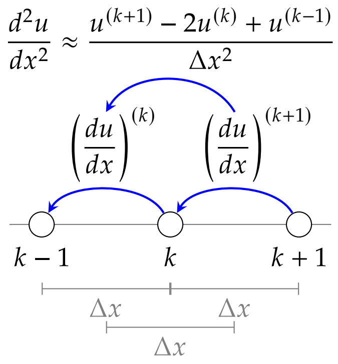
We write the value of u at node k as $u^{(k)}$. Using a superscript with parentheses avoids confusion with expressions u^k (the kth power of u) and u_k (the kth element of a vector named $\mathbf{u}$).

nodes $u^{(k)}$ and $u^{(k-1)}$. We divide this "difference between differences" by the distance between the centers of the nodes:

$$\frac{d^2u}{dx^2} \approx \frac{\left(\frac{du}{dx}\right)^{(k+1)} - \left(\frac{du}{dx}\right)^{(k)}}{\Delta x}$$

$$= \frac{\frac{u^{(k+1)} - u^{(k)}}{\Delta x} - \frac{u^{(k)} - u^{(k-1)}}{\Delta x}}{\Delta x}$$

$$= \frac{u^{(k+1)} - 2u^{(k)} + u^{(k-1)}}{\Delta x^2}$$



$\frac{d^2u}{dx^2} \approx \frac{u^{(k+1)} - 2u^{(k)} + u^{(k-1)}}{\Delta x^2}$
$\left(\frac{du}{dx}\right)^{(k)}$
$\left(\frac{du}{dx}\right)^{(k+1)}$
$k-1$
k
$k+1$
Δx
Δx
Δx

Figure 5.2: The second-order finite difference approximation is computed using two adjacent first-order approximations.

5.2 Linear Differential Equations

In order to generate a set of linear algebraic equations, the starting ODE or PDE must be linear. Linearity for differential equations means that the dependent variable (i.e. u) only appears in linear expressions; there can be nonlinearities involving only the independent variables. Remember that differentiation is a linear operator, so all derivatives of u are linear.

The finite difference method will still work on nonlinear PDEs; however, the resulting set of equations are nonlinear.

For example, consider the PDE with dependent variable $u(t, x)$:

$$t^2 \frac{\partial u}{\partial t} = c_1 \frac{\partial^2 u}{\partial x^2} + c_2 \sin(x) \frac{\partial u}{\partial x} + c_3 e^{tx}$$

This PDE is linear in u. However, the ODE

$$u \frac{du}{dx} = 0$$

is not linear. In general, for a variable $u(t, x)$, any terms of the form

$$f(t,x)\frac{\partial^n u}{\partial t^n} \quad \text{or} \quad f(t,x)\frac{\partial^n u}{\partial x^n}$$

are linear.

As a rule of thumb, you can tell if a PDE is linear in u by ignoring the derivative operators and seeing if the resulting algebraic equation is linear in u.

You might be wondering why we only require that a PDE be linear in the dependent variable. Why do nonlinearities in the independent variables not lead to nonlinear algebraic equations? Remember that in the finite difference method we discretize the independent variables across a grid and solve for the dependent variable at each node. Before solving, the value of the dependent variable is unknown. However, the value of all the independent variables are known, i.e. we know the location of every node in space and time. We can evaluate all terms involving the independent variables when setting up our equations.

5.3 Discretizing a Linear Differential Equation

Converting a linear differential equation into a set of linear algebraic equations requires three steps:

1. Divide the domain into n equally-sized intervals. Creating n intervals requires $n+1$ points, or *nodes*, labeled 0, 1, …, n. The spacing between each node is $\Delta x = l/n$ where l is the length of the domain.

2. Starting with the interior nodes $(1, 2, \ldots, n-1)$, we rewrite the differential equation at each node using finite differences.

$$\begin{aligned} u &\longrightarrow u^{(k)} \\ \frac{du}{dx} &\longrightarrow \frac{u^{(k+1)} - u^{(k)}}{\Delta x} \\ \frac{d^2u}{dx^2} &\longrightarrow \frac{u^{(k+1)} - 2u^{(k)} + u^{(k-1)}}{\Delta x^2} \end{aligned}$$

3. Add equations to enforce the boundary conditions at the boundary nodes.

For example, consider the ordinary differential equation

$$\frac{d^2u}{dx^2} + \frac{du}{dx} - 6u = 0, \quad u(0) = 0, \quad u(1) = 3$$

If we divide the domain $[0, 1]$ into four sections, then $n = 4$ and $\Delta x = l/n = 1/4 = 0.25$. We have five nodes (0, 1, 2, 3, 4), three of which are interior nodes (1, 2, 3). We rewrite the ODE using finite differences at each of the interior nodes.

$$\frac{u^{(2)} - 2u^{(1)} + u^{(0)}}{(0.25)^2} + \frac{u^{(2)} - u^{(1)}}{0.25} - 6u^{(1)} = 0 \quad \text{[node 1]}$$
$$\frac{u^{(3)} - 2u^{(2)} + u^{(1)}}{(0.25)^2} + \frac{u^{(3)} - u^{(2)}}{0.25} - 6u^{(2)} = 0 \quad \text{[node 2]}$$
$$\frac{u^{(4)} - 2u^{(3)} + u^{(2)}}{(0.25)^2} + \frac{u^{(4)} - u^{(3)}}{0.25} - 6u^{(3)} = 0 \quad \text{[node 3]}$$

which simplifies to the equations

$$16u^{(0)} - 42u^{(1)} + 20u^{(2)} = 0$$
$$16u^{(1)} - 42u^{(2)} + 20u^{(3)} = 0$$
$$16u^{(2)} - 42u^{(3)} + 20u^{(4)} = 0$$

Now we can add equations for the boundary nodes. Node 0 corresponds to $x = 0$, so the boundary condition tells us that $u^{(0)} = 0$. Similarly, node 4 corresponds to $x = 1$, so $u^{(4)} = 3$. Combining these two boundary equations with the equations for the interior nodes yields the final linear system

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 16 & -42 & 20 & 0 & 0 \\ 0 & 16 & -42 & 20 & 0 \\ 0 & 0 & 16 & -42 & 20 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u^{(0)} \\ u^{(1)} \\ u^{(2)} \\ u^{(3)} \\ u^{(4)} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 3 \end{pmatrix}$$

Solving by Gaussian elimination yields

$$\begin{pmatrix} u^{(0)} \\ u^{(1)} \\ u^{(2)} \\ u^{(3)} \\ u^{(4)} \end{pmatrix} = \begin{pmatrix} 0.00 \\ 0.51 \\ 1.07 \\ 1.84 \\ 3.00 \end{pmatrix}$$

We can compare our numerical solution to the exact solution for this expression, which is

$$u(x) = \frac{3}{e^2 - e^{-3}} \left(e^{2x} - e^{-3x} \right)$$

x	Numerical Solution	Exact Solution	Error
0	0.00	0.00	0.0%
0.25	0.51	0.48	5.7%
0.50	1.07	1.02	4.7%
0.75	1.84	1.79	2.6%
1	3.00	3.00	0.0%

We used only five nodes to solve this ODE, yet our relative error is less than 6%!

5.4 Boundary Conditions

There are two ways to write a finite difference approximation for the first derivative at node k. We can write the *forward difference* using node $k+1$:

$$\frac{du}{dx} \rightarrow \frac{u^{(k+1)} - u^{(k)}}{\Delta x}$$

or we can use the *backward difference* using the previous node at $k-1$:

$$\frac{du}{dx} \rightarrow \frac{u^{(k)} - u^{(k-1)}}{\Delta x}$$

The choice of forward or backward differences depends on the boundary conditions (Figure 5.3). For a first order ODE, we are given a boundary condition at either $x = 0$ or $x = l$. If we are given $u(0)$, we use backward differences. If we are given $u(l)$, we use forward differences. Otherwise, we run out of nodes when writing equations for the finite differences.

Second order finite differences require nodes on both sides. This is related to the necessity of two boundary conditions, since we cannot write finite difference approximations for nodes at either end. If your ODE comes with two boundary conditions, you can choose either forward or backward differences to approximate any first order derivatives.

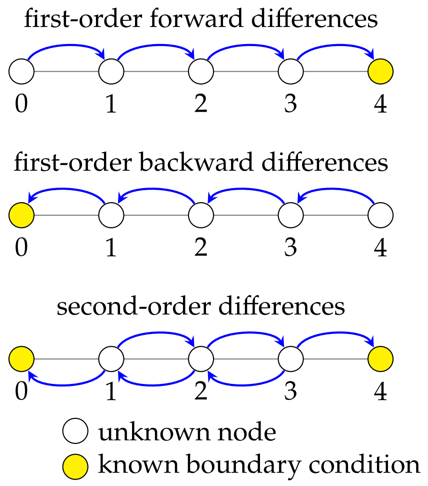


Figure 5.3: First-order equations use either forward or backward differences depending on the location of the boundary condition. Second-order equations use both forward and backward differences and require two boundary conditions.

You cannot avoid the issue by using the difference $u^{(0)} - u^{(1)}$ for node 0 and $u^{(1)} - u^{(0)}$ for node 1. These equations are *linearly dependent*, which leaves us with too little information when solving the system.

Chapter 6

Matrix Inverse

6.1 Defining the Matrix Inverse

So far we've demonstrated how Gaussian elimination can solve linear systems of the form $\mathbf{Ax} = \mathbf{y}$. Gaussian elimination involves a series of elementary row operations to transform the coefficient matrix $\mathbf{A}$ into the identity matrix. While Gaussian elimination works well, our initial goal of defining an algebra for vectors requires something stronger – a multiplicative inverse. For vectors, the multiplicative inverse is called a *matrix inverse*. For any square matrix, a matrix inverse (if it exists) is a square matrix $\mathbf{A}^{-1}$ such that

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{I} = \mathbf{A}\mathbf{A}^{-1}$$

This definition is analogous to the field axiom that there exists a^{-1} such that $a^{-1}a = 1$ for all nonzero a. Since scalar multiplication always commutes, $a^{-1}a = aa^{-1}$. Matrix multiplication does not commute, so we need to state this property separately.

If we could prove the existence of a *matrix inverse* for $\mathbf{A}$, we could solve a wide variety of linear systems, including $\mathbf{Ax}= \mathbf{y}$.

$$\begin{aligned}
\mathbf{Ax} &= \mathbf{y} \\
\mathbf{A}^{-1}\mathbf{Ax} &= \mathbf{A}^{-1}\mathbf{y} \\
\mathbf{I}\,\mathbf{x} &= \mathbf{A}^{-1}\mathbf{y} \\
\mathbf{x} &= \mathbf{A}^{-1}\mathbf{y}
\end{aligned}$$

Being amenable to Gaussian elimination is related to the existence of the matrix inverse. While the end result is the same (a transformation of the coefficient matrix into the identity matrix), the processes are different. The Gaussian elimination algorithm applies a series of elementary row operations while pivoting to avoid numerical issues. A matrix inverse (if it exists) must capture all of these operations in a single matrix multiplication. This condensing of Gaussian elimination is not a trivial task.

Our first goal for this chapter is to prove the existence of the matrix inverse for any coefficient matrix that can be solved by Gaussian elimination. Then we will derive a method to construct the matrix inverse if it exists. The following chapter formalizes the requirements for solvability of a linear system of equations, which is related to the existence of the matrix inverse.

6.2 Elementary Matrices

Before proving the existence of the matrix inverse, we need to add another matrix manipulation tool to our arsenal — the *elementary matrix*. An elementary matrix is constructed by applying any single elementary row operation to the identity matrix. For example, consider swapping the second and third rows of the 3×3 identity matrix:

We use the notation $\mathbf{E}_r$ to denote an elementary matrix constructed using the elementary row operation r.

$$\begin{pmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{pmatrix} \xrightarrow{R_2 \leftrightarrow R_3} \begin{pmatrix}1 & 0 & 0\\0 & 0 & 1\\0 & 1 & 0\end{pmatrix} = \mathbf{E}_{R_2 \leftrightarrow R_3}$$

Notice what happens when we left multiply a matrix with an elementary matrix.

$$\mathbf{E}_{R_2 \leftrightarrow R_3}\begin{pmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{pmatrix} = \begin{pmatrix}1 & 0 & 0\\0 & 0 & 1\\0 & 1 & 0\end{pmatrix}\begin{pmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{pmatrix} = \begin{pmatrix}1 & 2 & 3\\7 & 8 & 9\\4 & 5 & 6\end{pmatrix}$$

Multiplication by the elementary matrix exchanges the second and third rows — the same operation that created the elementary matrix. The same idea works for other elementary row operations, such as scalar multiplication

Multiplication by an elementary matrix on the right applies the same operation to the columns of the matrix.

$$\mathbf{E}_{3R_2} = \begin{pmatrix}1 & 0 & 0\\0 & 3 & 0\\0 & 0 & 1\end{pmatrix}$$

$$\mathbf{E}_{3R_2}\begin{pmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{pmatrix} = \begin{pmatrix}1 & 0 & 0\\0 & 3 & 0\\0 & 0 & 1\end{pmatrix}\begin{pmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{pmatrix} = \begin{pmatrix}1 & 2 & 3\\12 & 15 & 18\\7 & 8 & 9\end{pmatrix}$$

and addition by a scaled row

$$\mathbf{E}_{2R_3 \to R_2} = \begin{pmatrix}1 & 0 & 0\\0 & 1 & 2\\0 & 0 & 1\end{pmatrix}$$

$$\mathbf{E}_{2R_3\to R_2}\begin{pmatrix}1&2&3\\4&5&6\\7&8&9\end{pmatrix}=\begin{pmatrix}1&0&0\\0&1&2\\0&0&1\end{pmatrix}\begin{pmatrix}1&2&3\\4&5&6\\7&8&9\end{pmatrix}=\begin{pmatrix}1&2&3\\18&21&24\\7&8&9\end{pmatrix}$$

6.3 Proof of Existence for the Matrix Inverse

We are now ready to prove the existence of the inverse for any coefficient matrix that is solvable by Gaussian elimination, i.e. any square matrix that can be transformed into the identity matrix with elementary row operations. We will prove existence in three steps:

1. Construct a matrix **P** that looks like a left inverse (**PA** = **I**).
2. Show that this left inverse is also a right inverse (**AP** = **I**).
3. Show that the matrix inverse is unique, implying that **P** must be the inverse of **A**.

Theorem. *Suppose matrix* **A** *can be reduced to the identity matrix* **I** *by elementary row operations. Then there exists a matrix* **P** *such that* **PA** = **I**.

Proof. We assume that reducing **A** to **I** requires k elementary row operations. Let $\mathbf{E}_1, \ldots, \mathbf{E}_k$ be the associated elementary matrices. Then

$$\begin{aligned}\mathbf{E}_k\mathbf{E}_{k-1}\cdots\mathbf{E}_2\mathbf{E}_1\mathbf{A}&=\mathbf{I}\\(\mathbf{E}_k\mathbf{E}_{k-1}\cdots\mathbf{E}_2\mathbf{E}_1)\mathbf{A}&=\mathbf{I}\\\mathbf{PA}&=\mathbf{I}\end{aligned}$$

where $\mathbf{P}=\mathbf{E}_k\mathbf{E}_{k-1}\cdots\mathbf{E}_2\mathbf{E}_1$. □

Theorem. *If* **PA** = **I** *then* **AP** = **I**.

Proof.

$$\begin{aligned}\mathbf{PA}&=\mathbf{I}\\\mathbf{PAP}&=\mathbf{IP}\\\mathbf{P}(\mathbf{AP})&=\mathbf{P}\end{aligned}$$

Since **P** multiplied by **AP** gives **P** back again, **AP** must equal the identity matrix (**AP** = **I**). □

Theorem. *The inverse of a matrix is unqiue.*

Proof. Let $\mathbf{A}^{-1}$ be the inverse of $\mathbf{A}$, i.e. $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I} = \mathbf{A}\mathbf{A}^{-1}$. Suppose there exists another matrix $\mathbf{P} \neq \mathbf{A}^{-1}$ such that $\mathbf{PA} = \mathbf{I} = \mathbf{AP}$.

$$\begin{aligned} \mathbf{PA} &= \mathbf{I} \\ \mathbf{PAA}^{-1} &= \mathbf{IA}^{-1} \\ \mathbf{PI} &= \mathbf{A}^{-1} \\ \mathbf{P} &= \mathbf{A}^{-1} \end{aligned}$$

This contradicts our supposition that $\mathbf{P} \neq \mathbf{A}^{-1}$, so $\mathbf{A}^{-1}$ must be unique. □

6.4 Computing the Matrix Inverse

Our proof of existence for the matrix inverse provided a method to construct the inverse. We performed Gaussian elimination on a matrix and built an elementary matrix for each step. These elementary matrices were multiplied together to form the matrix inverse. In practice this method would be wildly inefficient. Transforming an $n \times n$ matrix to reduced row echelon form requires $O(n^2)$ elementary row operations, so we would need to construct and multiply $O(n^2)$ elementary matrices. Since naive matrix multiplication requires $O(n^3)$ operations per matrix, constructing a matrix inverse with this method requires $O(n^5)$ operations! Gaussian elimination is $O(n^3)$, so we would be far better off avoiding the matrix inverse entirely.

Gaussian elimination requires $O(n^2)$ row operations but $O(n^3)$ total operations. Each row operation requires $O(n)$ individual operations — one for each column.

The fastest matrix multiplication algorithm is $O(n^{2.7373})$, although this is not a big help in practice.

Fortunately, there are better methods for constructing matrix inverses. One of the best is called the *side-by-side method*. To see how the side-by-side method works, let's construct an inverse for the square matrix $\mathbf{A}$. We can use Gaussian elimination to transform $\mathbf{A}$ into the identity matrix, which we can represent with a series of k elementary matrices.

$$\underbrace{\mathbf{E}_k\mathbf{E}_{k-1}\cdots\mathbf{E}_2\mathbf{E}_1}_{\mathbf{A}^{-1}}\mathbf{A} = \mathbf{I}$$

What would happen if we simultaneously apply the same elementary row operations to another identity matrix?

$$\underbrace{\mathbf{E}_k\mathbf{E}_{k-1}\cdots\mathbf{E}_2\mathbf{E}_1}_{\mathbf{A}^{-1}}\mathbf{I} = \mathbf{A}^{-1}\mathbf{I} = \mathbf{A}^{-1}$$

In the side-by-side method, we start with an augmented matrix containing the $n \times n$ matrix $\mathbf{A}$ and an $n \times n$ identity matrix. Then we apply Gaussian elimination to transform $\mathbf{A}$ into $\mathbf{I}$. The augmented matrix ensures that the same elementary row operations will be applied to the identity matrix, yielding the inverse of $\mathbf{A}$, or

Like the augmented matrix $(\mathbf{A}\,\mathbf{y})$, there is no direct interpretation of $(\mathbf{A}\,\mathbf{I})$. It is simply a convenient way to apply the same EROs to both $\mathbf{A}$ and $\mathbf{I}$.

$$(\mathbf{A} \quad \mathbf{I}) \xrightarrow{\text{EROs}} \left(\mathbf{I} \quad \mathbf{A}^{-1}\right).$$

Let's solve the following system by constructing the matrix inverse.

$$\begin{aligned} 3x_1 + 2x_2 &= 7 \\ x_1 + x_2 &= 4 \end{aligned}$$

In matrix form,

$$\mathbf{A} = \begin{pmatrix} 3 & 2 \\ 1 & 1 \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} 7 \\ 4 \end{pmatrix}$$

We start with the augmented matrix for the side-by-side method.

$$\begin{aligned} \begin{pmatrix} 3 & 2 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix} &\xrightarrow{R_1 \leftrightarrow R_2} \begin{pmatrix} 1 & 1 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{pmatrix} \\ &\xrightarrow{-3R_1 \to R_2} \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & -1 & 1 & -3 \end{pmatrix} \\ &\xrightarrow{-R_2} \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & -1 & 3 \end{pmatrix} \\ &\xrightarrow{-R_2 \to R_1} \begin{pmatrix} 1 & 0 & 1 & -2 \\ 0 & 1 & -1 & 3 \end{pmatrix} \end{aligned}$$

Thus, the matrix inverse is

$$\mathbf{A}^{-1} = \begin{pmatrix} 1 & -2 \\ -1 & 3 \end{pmatrix}$$

which we can verify by multiplication on the left and right.

$$\begin{aligned} \mathbf{A}^{-1}\mathbf{A} &= \begin{pmatrix} 1 & -2 \\ -1 & 3 \end{pmatrix} \begin{pmatrix} 3 & 2 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \mathbf{I} \\ \mathbf{A}\mathbf{A}^{-1} &= \begin{pmatrix} 3 & 2 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & -2 \\ -1 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \mathbf{I} \end{aligned}$$

Finally, we can compute the solution by matrix multiplication.

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{y} = \begin{pmatrix} 1 & -2 \\ -1 & 3 \end{pmatrix} \begin{pmatrix} 7 \\ 4 \end{pmatrix} = \begin{pmatrix} -1 \\ 5 \end{pmatrix}$$

The advantage of having a matrix inverse is that if only the right hand side of a linear system changes, we do not need to retransform the coefficient matrix. For example, to solve

$$\begin{aligned} 3x_1 + 2x_2 &= 1 \\ x_1 + x_2 &= 3 \end{aligned}$$

we can re-use $\mathbf{A}^{-1}$ since $\mathbf{A}$ is unchanged (only $\mathbf{y}$ is different).

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{y} = \begin{pmatrix} 1 & -2 \\ -1 & 3 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \end{pmatrix} = \begin{pmatrix} -5 \\ 8 \end{pmatrix}$$

6.5 Numerical Issues

Matrix inversion is a powerful method for solving linear systems. However, calculating a matrix inverse should always be your last resort. There are far more efficient and numerically stable methods for solving linear systems. Reasons against using a matrix inverse include:

1. **Computation time.** The side-by-side method is more efficient than multiplying elementary matrices for constructing matrix inverses. Regardless, both methods are much slower than Gaussian elimination for solving linear systems of the form $\mathbf{A}\mathbf{x} = \mathbf{y}$. Both the side-by-side method and Gaussian elimination reduce an augmented matrix. For an $n \times n$ matrix $\mathbf{A}$, the augmented matrix for Gaussian elimination $(\mathbf{A}\,\mathbf{y})$ is $n \times (n+1)$. The augmented matrix for the side-by-side method $(\mathbf{A}\,\mathbf{I})$ is $n \times 2n$. Solving for the inverse requires nearly twice the computation as solving the linear system directly. Having the inverse allows us to “resolve" the system with a new right hand side vector for only the cost of a matrix multiplication. However, there are variants of Gaussian elimination – such as *LU* decomposition – that allow resolving without repeating the entire reduction of the coefficient matrix $\mathbf{A}$.

2. **Memory.** Most large matrices in engineering are *sparse*. Sparse matrices contain very few nonzero entries; matrices with less than 0.01% nonzero entries are not uncommon. Examples of sparse matrices include matrices generated from finite difference approximations or matrices showing connections between nodes in large networks. Computers store sparse matrices by only storing the nonzero entries and their locations. However, there is no guarantee that the inverse of a sparse matrix will also be sparse. Consider the arrow

Imagine the connection matrix for Facebook. It would have hundreds of millions of rows and columns, but each person (row) would only have nonzero entries for a few hundred people (columns) that they knew.

matrix, a matrix with ones along the diagonal and last column and row.

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

An $n \times n$ arrow matrix has n^2 entries but only $3n - 2$ nonzeros. However, the inverse of an arrow matrix always has 100% nonzeros. For example, the inverse of the 8×8 matrix above is

A 1000×1000 arrow matrix has less than 0.3% nonzeros.

$$\mathbf{A}^{-1} = \frac{1}{6}\begin{pmatrix} 5 & -1 & -1 & -1 & -1 & -1 & -1 & 1 \\ -1 & 5 & -1 & -1 & -1 & -1 & -1 & 1 \\ -1 & -1 & 5 & -1 & -1 & -1 & -1 & 1 \\ -1 & -1 & -1 & 5 & -1 & -1 & -1 & 1 \\ -1 & -1 & -1 & -1 & 5 & -1 & -1 & 1 \\ -1 & -1 & -1 & -1 & -1 & 5 & -1 & 1 \\ -1 & -1 & -1 & -1 & -1 & -1 & 5 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 \end{pmatrix}$$

Calculating the inverse of a large, sparse matrix could requires orders of magnitude more memory than the original matrix. Storing – much less computing – the inverse is impossible for some matrices.

Despite its disadvantages, the matrix inverse is still a powerful tool. The matrix inverse is necessary for many algebraic manipulations, and the inverse can be used to simply or prove many matrix equations. Just remember to think critically about the need for a matrix inverse before calculating one.

6.6 Inverses of Elementary Matrices

We conclude with another interesting property of elementary matrices. We said before that left multiplication by an elementary matrix performs an elementary row operation (the same ERO that was used to construct the elementary matrix). Left multiplication by the inverse of an elementary matrix "undoes" the operation of the elementary matrix. For example, the elementary matrix $\mathbf{E}_{3R_2}$ scales the second row by two. The inverse $\mathbf{E}_{3R_2}^{-1}$ would scale the second row by $1/2$, undoing the scaling by two. Similarly, $\mathbf{E}_{R_2\leftrightarrow R_3}$ swaps rows two and three, and $\mathbf{E}^{-1}_{R_2\leftrightarrow R_3}$ swaps them back. The proof of this property is straightforward.

Theorem. *If the elementary matrix $\mathbf{E}_r$ performs the elementary row operation r, then left multiplication by the inverse $\mathbf{E}_r^{-1}$ undoes this operation.*

Proof.

$$\mathbf{E}_r^{-1}(\mathbf{E}_r\mathbf{A}) = (\mathbf{E}_r^{-1}\mathbf{E}_r)\mathbf{A} = (\mathbf{I})\mathbf{A} = \mathbf{A}$$

□

Chapter 7

Rank and Solvability

7.1 Rank of a Matrix

Consider the linear system

$$\begin{pmatrix} 1 & 0 & 3 \\ 0 & 2 & -4 \\ -2 & 0 & -6 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ -2 \\ -4 \end{pmatrix}$$

and the row echelon form of the associated augmented matrix

$$\begin{pmatrix} 1 & 0 & 3 & 2 \\ 0 & 2 & -4 & -2 \\ -2 & 0 & -6 & -4 \end{pmatrix} \xrightarrow{\frac{1}{2}R_2} \xrightarrow{2R_1 \to R_3} \begin{pmatrix} 1 & 0 & 3 & 2 \\ 0 & 1 & -2 & -1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Notice that the last row is all zeros. We have no information about the last entry (x_3). However, this does not mean we cannot solve the linear system. Since x_3 is unknown, let us assign its value the symbol α. Then, by back substitution

$$\begin{aligned} & & x_3 &= \alpha \\ x_2 - 2x_3 = -1 &\Rightarrow & x_2 &= 2\alpha - 1 \\ x_1 + 3x_3 = 2 &\Rightarrow & x_1 &= 2 - 3\alpha \end{aligned}$$

The above linear system has not one solution, but infinitely many. There is a solution for every value of the parameter α, so we say the system has a *parameterized solution*.

Parameterized solutions are necessary any time row echelon reduction of a matrix leads to one or more rows with all zero entries. The number of nonzero rows in the row echelon form of a matrix is the matrix's *rank*. The rank of a matrix can be calculated by counting the number of nonzero rows after the matrix is transformed into row echelon form by Gaussian elimination. In general, if a matrix with n columns has rank n, it is possible to find a unique solution to the system $\mathbf{Ax} = \mathbf{y}$. If $\text{rank}(\mathbf{A}) < n$, there may be infinitely many solutions. These solutions require that we specify $n - \text{rank}(\mathbf{A})$ parameters.

We denote the rank of a matrix $\mathbf{A}$ as $\text{rank}(\mathbf{A})$.

Matrices have both a *row rank* (the number of nonzero rows in row-reduced echelon form) and a *column rank* (the number of nonzero columns in a column-reduced echelon form). Thus the concept of rank also applies to nonsquare matrices. However, the row and column ranks are always equivalent, even if the matrix is not square.

Theorem. *The row rank of a matrix equals the column rank of the matrix, i.e.* $\text{rank}(\mathbf{A}) = \text{rank}(\mathbf{A}^\mathsf{T})$.

Proof. We will defer the proof of this theorem until after we learn about matrix decompositions in Part III. □

The equivalence of the row and column ranks implies an upper bound on the rank of nonsquare matrices.

Theorem. *The rank of a matrix is less than or equal to the smallest dimension of the matrix, i.e.* $\text{rank}(\mathbf{A}) \le \min(\dim \mathbf{A})$.

Proof. The row rank of $\mathbf{A}$ is the number of nonzero rows in the row-reduced $\mathbf{A}$, so the rank of $\mathbf{A}$ must be less than the number of rows in $\mathbf{A}$. Since the row rank is also equal to the column rank, there must also be $\text{rank}(\mathbf{A})$ nonzero columns in the column-reduced $\mathbf{A}$. So the rank of $\mathbf{A}$ must never be larger than either the number of rows or number of columns in $\mathbf{A}$. □

We say that a matrix is *full rank* if it has the maximum possible rank (rank n for an $n \times n$ square matrix or rank $\min(m, n)$ for an $m \times n$ rectangular matrix). A matrix that is not full rank is *rank deficient*.

7.1.1 Properties of Rank

The rank of a matrix is connected with many other matrix operations. Here are a few properties of matrices and their ranks. For any $m \times n$ matrix $\mathbf{A}$:

1. $\text{rank}(\mathbf{A}) \le \min(m, n)$.
2. $\text{rank}(\mathbf{A}) = \text{rank}(\mathbf{A}^\mathsf{T})$, i.e. row rank equals column rank.

3. $\text{rank}(\mathbf{A}) = 0$ if and only if $\mathbf{A}$ is the zero matrix.
4. $\text{rank}(\mathbf{A}^\mathsf{T}\mathbf{A}) = \text{rank}(\mathbf{A}\mathbf{A}^\mathsf{T}) = \text{rank}(\mathbf{A})$.
5. $\text{rank}(\mathbf{AB}) \leq \min\{\text{rank}(\mathbf{A}), \text{rank}(\mathbf{B})\}$ provided $\mathbf{A}$ and $\mathbf{B}$ are conformable.
6. $\text{rank}(\mathbf{A}+\mathbf{B}) \leq \text{rank}(\mathbf{A}) + \text{rank}(\mathbf{B})$ for any $m \times n$ matrix $\mathbf{B}$.

7.2 Linear Independence

The notion of rank is deeply tied to the concept of *linear independence*. A vector $\mathbf{x}_i$ is linearly dependent on a set of n vectors $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_n$ if there exits coefficients c_1, c_2, ..., c_n such that

$$\mathbf{x}_i = c_1\mathbf{x}_1 + c_2\mathbf{x}_2 + \cdots + c_n\mathbf{x}_n$$

Note the difference between $\mathbf{x}_1, \ldots, \mathbf{x}_n$, a set of n vectors; and $x_1, \ldots, x_n$, a set of n scalars that form the elements of a vector $\mathbf{x}$.

A set of vectors are *linearly dependent* if one of the vectors can be expressed as a linear combination of some of the others. This is analogous to saying there exists a set of coefficients $c_1, \ldots, c_n$, not all equal to zero, such that

$$c_1\mathbf{x}_1 + c_2\mathbf{x}_2 + \cdots + c_n\mathbf{x}_n = \mathbf{0}$$

Imagine that a coefficient $c_i \neq 0$. If we move the term $c_i\mathbf{x}_i$ to the other side of the equation and divide by $-c_i$, we are back to the $\mathbf{x}_i = c_1\mathbf{x}_1 + \cdots + c_n\mathbf{x}_n$ definition of linear independence (although the coefficients will have changed by a factor of $-c_i$).

If a matrix with n rows has rank $k < n$, then $n - k$ of the rows are linearly dependent on the other k rows. Going back to our previous example, the matrix

$$\begin{pmatrix} 1 & 0 & 3 \\ 0 & 2 & -4 \\ -2 & 0 & -6 \end{pmatrix} \xrightarrow{\text{EROs}} \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & -2 \\ 0 & 0 & 0 \end{pmatrix}$$

has rank 2 since there are two nonzero rows in the row-reduced matrix. Therefore, one of the rows must be linearly dependent on the other rows. Indeed, we see that the last row $\begin{pmatrix} -2 & 0 & -6 \end{pmatrix}$ is -2 times the first row $\begin{pmatrix} 1 & 0 & 3 \end{pmatrix}$. During row reduction by Gaussian elimination, any linearly dependent rows will be completely zeroed out, revealing the rank of the matrix.

Rank and linear dependence tell us about the information content of a coefficient matrix. If some of the rows of the coefficient matrix are linearly dependent, then matrix is rank deficient and no unique solution exists. These matrices are also information deficient – we do not have one independent expression for each variable. Without a separate piece of information for each variable, we cannot uniquely map between the input $\mathbf{x}$ and the output $\mathbf{y}$. However, if we introduce a separate parameter for each zeroed row, we are artificially providing the missing information. We can find a new solution every time we specify values for the parameters.

7.3 Homogeneous Systems ($\mathbf{Ax} = \mathbf{0}$)

A linear systems of equations is *homogeneous* if and only if the right hand side vector ($\mathbf{y}$) is equal to the zero vector ($\mathbf{0}$). Homogeneous systems always have at least one solution, $\mathbf{x} = \mathbf{0}$, since $\mathbf{A0} = \mathbf{0}$. The zero solution to a homogeneous system is called the *trivial solution*. Some homogeneous systems have a nontrivial solution, i.e. a solution $\mathbf{x} \neq \mathbf{0}$. If a homogeneous system has a nontrivial solution, then it has infinitely many solutions. We prove this result below.

Theorem. *Any linear combination of nontrivial solutions to a homogeneous linear system is also a solution.*

This proof is equivalent to showing that $\mathbf{Ax} = \mathbf{0}$ satisfies our definition of a linear system: $f(k_1x_1 + k_2x_2) = k_1f(x_1) + k_2f(x_2)$.

Proof. Suppose we had two solutions, $\mathbf{x}$ and $\mathbf{x}'$, to the homogeneous system $\mathbf{Ax} = \mathbf{0}$. Then

$$\begin{aligned}\mathbf{A}(k\mathbf{x} + k'\mathbf{x}') &= \mathbf{A}(k\mathbf{x}) + \mathbf{A}(k'\mathbf{x}') \\ &= k(\mathbf{Ax}) + k'(\mathbf{Ax}') \\ &= k(\mathbf{0}) + k'(\mathbf{0}) \\ &= \mathbf{0}\end{aligned}$$

Since there are infinitely many scalars k and k', we can generate infinitely many solutions to the homogeneous system $\mathbf{Ax} = \mathbf{0}$. □

There is a connection between the solvability of nonhomogeneous systems $\mathbf{Ax} = \mathbf{y}$ and the corresponding homogeneous system $\mathbf{Ax} = \mathbf{0}$. If there exists at least one solution to $\mathbf{Ax} = \mathbf{y}$ and a nontrivial solution to $\mathbf{Ax} = \mathbf{0}$, then there are infinitely many solutions to $\mathbf{Ax} = \mathbf{y}$. To see why, let $\mathbf{x}_{\mathrm{nh}}$ be the solution to the nonhomogeneous system ($\mathbf{Ax}_{\mathrm{nh}} = \mathbf{y}$) and $\mathbf{x}_{\mathrm{h}}$ be a nontrivial solution to the homogeneous system $\mathbf{Ax}_{\mathrm{h}} = \mathbf{0}$. Then any of the infinite linear combinations $\mathbf{x}_{\mathrm{nh}} + k\mathbf{x}_{\mathrm{h}}$ is also a solution to $\mathbf{Ax} = \mathbf{y}$ since

$$\mathbf{A}(\mathbf{x}_{\mathrm{nh}} + k\mathbf{x}_{\mathrm{h}}) = \mathbf{Ax}_{\mathrm{nh}} + k\mathbf{Ax}_{\mathrm{h}} = \mathbf{y} + k\mathbf{0} = \mathbf{y}$$

7.4 General Solvability

We can use the rank of the coefficient matrix and the augmented matrix to determine the existence and number of solutions for any linear system of equations. The relationship between solvability and rank is captured by the Rouché-Capelli theorem:

Theorem. *A linear system* $\mathbf{A}\mathbf{x} = \mathbf{y}$ *where* $\mathbf{x} \in \mathbb{R}^n$ *has a solution if and only if the rank of the coefficient matrix equals the rank of the augmented matrix, i.e.* $\text{rank}(\mathbf{A}) = \text{rank}([\mathbf{A}\,\mathbf{y}])$. *Furthermore, the solution is unique if* $\text{rank}(\mathbf{A}) = n$*; otherwise there are infinitely many solutions.*

Proof. We will sketch several portions of this proof to give intuition about the theorem. A more rigorous proof is beyond the scope of this class.

1. **Homogeneous systems.** For a homogeneous system $\mathbf{A}\mathbf{x} = \mathbf{0}$, we know that $\text{rank}(\mathbf{A}) = \text{rank}([\mathbf{A}\,\mathbf{0}])$. (Since the rank of $\mathbf{A}$ is equal to the number of nonzero columns, adding another column of zeros will never change the rank.) Thus, we know that homogeneous systems are always solvable, at least by the trivial solution $\mathbf{x} = \mathbf{0}$. If $\text{rank}(\mathbf{A}) = n$, then the trivial solution is unique and is the only solution. If $\text{rank}(\mathbf{A}) < n$, there are infinitely many parameterized solutions.

2. **Full rank, nonhomogeneous systems**. For a nonhomogeneous system ($\mathbf{A}\mathbf{x} = \mathbf{y}$, $\mathbf{y} \neq \mathbf{0}$), we expect a unique solution if and only if adding the column $\mathbf{y}$ to the coefficient matrix doesn't change the rank. For this to be true, $\mathbf{y}$ must be linearly dependent on the other columns in $\mathbf{A}$; otherwise, adding a new linearly independent column would increase the rank. If $\mathbf{y}$ is linearly dependent on the n columns of $\mathbf{A}$, it must be true that there exists weights $c_1, c_2, \ldots, c_n$ such that

$$c_1\mathbf{A}(:,1) + c_2\mathbf{A}(:,2) + \cdots + c_n\mathbf{A}(:,n) = \mathbf{y}$$

   based on the definition of linear dependence. But the above expression can be rewritten in matrix form as

$$\left(\mathbf{A}(:,1)\,\mathbf{A}(:,2)\,\cdots\,\mathbf{A}(:,n)\right)\begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \mathbf{A}\mathbf{c} = \mathbf{y}$$

   which shows that the system has a unique solution $\mathbf{x} = \mathbf{c}$.

Take time to understand the connection between the solvability of $\mathbf{A}\mathbf{x} = \mathbf{y}$ and the ability to express $\mathbf{y}$ as a linear combination of the columns in $\mathbf{A}$. This relationship is important for Part III of the course.

3. **Rank deficient, nonhomogeneous systems**. Let $\text{rank}(\mathbf{A}) = k < n$. Then the row-reduced form of $\mathbf{A}$ will have k rows that resemble the identity matrix

and $n-k$ rows of all zeros:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1k} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2k} & \cdots & a_{2n} \\ & \vdots & & & \vdots & \\ a_{k1} & a_{k2} & \cdots & a_{kk} & \cdots & a_{kn} \\ a_{k+1,1} & a_{k+1,2} & \cdots & a_{k+1,k} & \cdots & a_{k+1,n} \\ & \vdots & & & \vdots & \\ a_{n1} & a_{n2} & \cdots & a_{nk} & \cdots & a_{nn} \end{pmatrix} \xrightarrow{\text{EROs}} \begin{pmatrix} 1 & a'_{12} & \cdots & a'_{1k} & \cdots & a'_{1n} \\ 0 & 1 & \cdots & a'_{2k} & \cdots & a'_{2n} \\ & \vdots & & & \vdots & \\ 0 & 0 & \cdots & 1 & \cdots & a'_{kn} \\ 0 & 0 & \cdots & 0 & \cdots & 0 \\ & \vdots & & & \vdots & \\ 0 & 0 & \cdots & 0 & \cdots & 0 \end{pmatrix}$$

Now imagine we performed the same row reduction on the augmented matrix $(\mathbf{A}\,\mathbf{y})$. We would still end up with $n-k$ rows with zeros in the first n columns (the columns of $\mathbf{A}$):

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1k} & \cdots & a_{1n} & y_1 \\ a_{21} & a_{22} & \cdots & a_{2k} & \cdots & a_{2n} & y_2 \\ & \vdots & & & \vdots & & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{kk} & \cdots & a_{kn} & y_k \\ a_{k+1,1} & a_{k+1,2} & \cdots & a_{k+1,k} & \cdots & a_{k+1,n} & y_{k+1} \\ & \vdots & & & \vdots & & \\ a_{n1} & a_{n2} & \cdots & a_{nk} & \cdots & a_{nn} & y_n \end{pmatrix} \xrightarrow{\text{EROs}} \begin{pmatrix} 1 & a'_{12} & \cdots & a'_{1k} & \cdots & a'_{1n} & y'_1 \\ 0 & 1 & \cdots & a'_{2k} & \cdots & a'_{2n} & y'_2 \\ & \vdots & & & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & \cdots & a'_{kn} & y'_k \\ 0 & 0 & \cdots & 0 & \cdots & 0 & y'_{k+1} \\ & \vdots & & & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & 0 & y'_n \end{pmatrix}$$

We know that if $y'_{k+1},\dots,y'_n=0$, we can solve this system by designating $n-k$ parameters for the variables $x_{k+1},\dots,x_n$ for which we have no information. However, notice what happens if any of the values $y'_{k+1},\dots,y'_n$ are nonzero. Then we have an expression of the form $0=y'_i\neq 0$, which is nonsensical. Therefore, the only way we can solve this system is by requiring that $y'_{k+1},\dots,y'_n=0$. This is exactly the requirement that the rank of the augmented matrix equal k, the rank of the matrix $\mathbf{A}$ by itself. If any of the $y'_{k+1},\dots,y'_n$ are nonzero, then the augmented matrix has one fewer row of zeros, so the rank of the augmented matrix would be greater than the rank of the original matrix. There are two ways to interpret this result. First, we require that the right hand side ($\mathbf{y}$) doesn't "mess up" our system by introducing a nonsensical expression. Second, if a row i in the matrix $\mathbf{A}$ is linearly dependent on the other rows in $\mathbf{A}$, the corresponding values y_i must have the same dependency on the other values in $\mathbf{y}$. If so, when the row i is zeroed out during row reduction, the value y_i will also be zeroed out, avoiding any inconsistency.

□

7.5 Rank and Matrix Inversion

For a general nonhomogeneous system $\mathbf{Ax} = \mathbf{y}$ with $\mathbf{A} \in \mathbb{R}^{n \times n}$, we know that a unique solution only exists if $\text{rank}(\mathbf{A}) = \text{rank}([\mathbf{A}\,\mathbf{y}]) = n$. If $\text{rank}(\mathbf{A}) = n$, we know that $\mathbf{A}$ can be transformed into reduced row form without generating any rows with all zero entries. We also know that if an inverse $\mathbf{A}^{-1}$ exists for $\mathbf{A}$, we can use the inverse to uniquely solve for $\mathbf{x} = \mathbf{A}^{-1}\mathbf{y}$. Putting these facts together, we can now definitely state necessary and sufficient conditions for matrix inversion:

An $n \times n$ matrix $\mathbf{A}$ has an inverse if and only if $\text{rank}(\mathbf{A}) = n$.

7.6 Summary

We've shown in this chapter the tight connections between matrix inversion, solvability, and the rank of a matrix. We will use these concepts many times to understand the solution of linear systems. However, we've also argued that despite their theoretical importance, these concepts have limited practical utility for solving linear systems. For example, computing the rank of a matrix requires transforming the matrix into reduced echelon form. This requires the same computations as solving a linear system involving the matrix, so one would rarely check the rank of a coefficient matrix before attempting to solve a linear system. Instead, we will see rank emerge as a useful tool only when considering matrices by themselves in Part III of this course.

Chapter 8

Linear Models and Regression

In Chapter 7 we learned that not all linear systems are solvable, depending on the amount and consistency of the information contained in the corresponding systems of equations. Too little information yielded an infinite number of solutions, and some systems have no solution at all. In all cases, we treated the entries of the coefficient matrix $\mathbf{A}$ and the output vector $\mathbf{y}$ as exact. We never doubted the accuracy of our measurements of either the system or its outputs. This view may be sufficient for linear algebra textbooks, but it is detached from the realities of the real world faced by engineers.

This chapter addresses two related problems. First, how do we solve linear systems when our measurements of either the system or its outputs are noisy? The practical solution for noisy data is to simply make more measurements, but leads to a second problem: How do we solve a linear system when we have more information than is required by the solvability conditions described in Chapter 7? The answer to both of these questions is *linear regression*, a powerful tool that combines linear algebra with statistics. This chapter introduces regression and the tools used to fit linear systems to noisy data. Linear regression is the first machine learning technique we've encountered, so we will present linear regression within a general framework of minimizing a loss function. This is not the usual way to present linear regression, but it will enable us to connect regression with other machine learning techniques.

The following chapter (Chapter 9) focuses on the practice of linear modeling, i.e. how to build and interpret models. You may be surprised by the flexibility of linear models and how many systems — both linear and nonlinear — can be analyzed with this single technique. Furthermore, all of these models can be solved using the matrix formalism described in this chapter.

8.1 The Problems of Real Data

Imagine you measured an input (x) and the corresponding output (y). You hypothesized a linear relationship between x and y, i.e.

$$y = \beta x.$$

We refer to the unknown quantity β as a *parameter*. Our goal is to estimate the value of β using our input and output measurements. If $y = 2.4$ when $x = 2$, then you can calculate the value of the parameter β with simple algebra.

$$\beta = y/x = 2.4/2 = 1.2$$

You could plot the relationship between x and y, which is just the line $y = 1.2x$. Given an input x, it is easily to calculate the corresponding value y. All of this works because we've assumed our measurements of the input x and output y are exact. With only one unknown (β), we have sufficient information to calculate a value with only one pair of input/output observations. However, life is messy. Measurements are noisy and filled with error and uncertainty. Even if the underlying relationship between y and x was really a factor of 1.2, we would never observe a value of y that was exactly 1.2 times x.

To compensate for the imprecision of the real world, engineers make multiple observations of their systems. Imagine we made five "noisy" measurements of the input x and output y, which we will call x^{true} and y^{true}.

x^{true}	y^{true}
0.07	−0.05
0.16	0.40
0.48	0.66
0.68	0.65
0.83	1.12

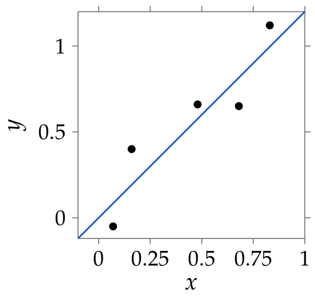


Figure 8.1: Five noisy observations (points) of the linear relationship $y = 1.2\,x$.

We refer to our observations of x and y as "true" because they are values we have actually measured. They are true in the sense that they appeared in the real world. Unfortunately, each pair of measurements gives an different estimate for the value of the parameter β:

1. $\beta = -0.05/0.07 = -0.7$
2. $\beta = 0.40/0.16 = 2.5$
3. $\beta = 0.66/0.48 = 1.3$
4. $\beta = 0.65/0.68 = 0.9$
5. $\beta = 1.12/0.83 = 1.3$

Using Figure 8.1 we can be reasonably certain that $y = 1.2x$ is a good representation of the relationship between y and x, so we expect that β is approximately 1.2. But our estimates for β vary wildly, from -0.7 to 2.5, and none of the estimates match the true value of 1.2. We could average the individual estimates to produce a single composite result ($\beta = 1.09$), but we would still miss the true value by nearly 9%.

Using the same five observations and the statistical techniques in this chapter, we can estimate β to be 1.21, an error of less than 1%.

This leads us to the central question in statistical modeling: "How accurate is our model?" So far we've been comparing our estimate of the parameter β to its "real" value, but this cannot be a solution. If we knew the actual value of β beforehand, then we don't need to estimate β from data! In practice we will never know the actual values of the parameters, so we cannot judge our models by the estimates of their parameters. Instead, the measure of a model is the accuracy of its predictions, and predictive accuracy is something we can easily quantify.

8.2 The Loss Function

After we estimate a value for the parameter β, we can make predictions using our model. The outputs of the model are only predictions; they have not been observed in the real world and exist only as informed guesses made by the model. We will call the predicted outputs from the model y^{pred}.

Our model's predictions (y^{pred}) will never match the observed values (y^{true}) exactly. Remember that the observed values include noise from both random fluctuations in the system and uncertainty in our measurement devices. We can compare the measured outputs (y^{true}) and the predicted outputs (y^{pred}) that we calculate using our model and the observed inputs ($y^{\text{pred}} = 1.2\, x^{\text{true}}$). The differences between the predicted outputs and the observed outputs ($y^{\text{pred}} - y^{\text{true}}$) are called the *residuals*.

x^{true}	y^{true}	Prediction ($y^{\text{pred}} = 1.2\, x^{\text{true}}$)	Residual ($y^{\text{pred}} - y^{\text{true}}$)
0.07	−0.05	0.084	0.134
0.16	0.40	0.192	−0.208
0.48	0.66	0.576	−0.084
0.68	0.65	0.816	0.166
0.83	1.12	0.996	−0.124

Our goal when fitting a model (finding values for the unknown parameters) is to minimize the residuals. But "minimizing the residuals" is subjective. We need a method to penalize large residuals and encourage small ones, and there are many choices for converting a residual into a penalty. For example, we might accept one very large residual if doing so makes all the other residuals tiny. Alternatively, we might want all of our residuals to be of a similar size so our model is equally uncertain over all possible inputs. The choice of penalties is entirely up to us. There is no wrong answer, but different penalties will give different parameter estimates for the same data.

The penalties calculated from the residuals is called the *loss*. There are many *loss functions*, each with strengths and weaknesses; however, all loss functions must have two properties. First, the loss must never be negative. Negative penalties are nonsensical and lead to problems during minimization. Second, the loss should be zero only when the corresponding residual is zero (i.e. when $y^{\text{pred}} = y^{\text{true}}$). If there is any discrepancy between a model's predictions and an observed value, there must also be a nonzero loss to signal that the model's predictions could be improved. Conversely, any improvement in the model's predictions — no matter how small — must also lead to a decrease in the loss.

There are two common loss functions that satisfy the above properties. The first is the *absolute loss*

$$\text{absolute loss}(y^{\text{true}}, y^{\text{pred}}) = \left|y^{\text{true}} - y^{\text{pred}}\right| .$$

The second is the *quadratic loss*

$$\text{quadratic loss}(y^{\text{true}}, y^{\text{pred}}) = \left(y^{\text{true}} - y^{\text{pred}}\right)^2 .$$

For fitting linear models we will use the quadratic loss because it has several advantages over the absolute loss:

1. Quadratic functions are continuously differentiable, while the derivative of the absolute value is discontinuous at zero. A continuous first derivative makes it easy to optimize functions involving the squared error.

2. The quadratic loss more harshly penalizes predictions that are far from the observed values. If a prediction is twice as far from an observation, the quadratic loss increases by a factor of four while the absolute loss only doubles. We will see shortly that assigning large penalties to far away points is more intuitive.

3. For linear models, there is always a single solution when minimizing quadratic loss, but there can be infinitely many solutions that minimize the absolute loss. We prefer having a unique solution whenever possible.

8.2.1 Loss Depends on Parameters, Not Data

The loss function quantifies how we've chosen to penalize the differences between the model predictions and the observed data. We must always remember that the loss is a function of the model's parameters, not the data. We will refer to the loss function as L, so the quadratic loss for a single piece of data is

$$L_i = \left(y_i^{\text{pred}} - y_i^{\text{true}}\right)^2 .$$

The subscript i reminds us that we are only considering the loss of one $(x^{\text{true}}, y^{\text{true}})$ pair in the dataset. Our simple linear model has the form $y^{\text{pred}} = \beta x^{\text{true}}$, which we substitute into the loss function.

$$L_i = \left(\beta x_i^{\text{true}} - y_i^{\text{true}}\right)^2$$

Consider the second observation from the dataset for Figure 8.1: $x_2^{\text{true}} = 0.16$, $y_2^{\text{true}} = 0.40$. The loss for this single point is

$$L_2 = \left(0.16\beta - 0.40\right)^2 .$$

The loss L_2 depends on β, not the quantities x_2^{true} and y_2^{true}. We minimize the model's loss by adjusting β. The training data x^{true} and y^{true} are fixed — we measured them before we started fitting the model. This can seem unusual, as previous math courses have conditioned us to always think of x and y as variables, not fixed quantities. We will write the loss function as $L(\beta)$ to remind us that loss is a function of the model's parameters.

8.2.2 Minimizing the Total Loss

In the previous section we discussed the loss for a single point. Let's write out the loss for all five points in the dataset from Figure 8.1.

$$
\begin{aligned}
L_1(\beta) &= \left(0.07\beta + 0.05\right)^2 \\
L_2(\beta) &= \left(0.16\beta - 0.40\right)^2 \\
L_3(\beta) &= \left(0.48\beta - 0.66\right)^2 \\
L_4(\beta) &= \left(0.68\beta - 0.65\right)^2 \\
L_5(\beta) &= \left(0.83\beta - 1.12\right)^2
\end{aligned}
$$

There is only one parameter β that appears in all five losses. When we fit a model by selecting a value for β, our goal is to minimize the total loss across all five points:

$$\text{total loss} = L(\beta) = L_1 + L_2 + L_3 + L_4 + L_5 = \sum_i L_i .$$

All of the individual losses depend on β, and this shared dependence complicates our search for a single "best" parameter value. Imagine we focused only on the first loss, L_1. The value of β that minimizes L_1 is $\beta = -0.05/0.07 \approx -0.714$. This value for β is ideal for the loss L_1, setting this individual loss to zero. However, the value $\beta = -0.714$ is a terrible choice for the other points in the dataset.

$$
\begin{aligned}
L_1(-0.714) &= [0.07(-0.714) + 0.05]^2 = 0 \\
L_2(-0.714) &= [0.16(-0.714) - 0.40]^2 = 0.264 \\
L_3(-0.714) &= [0.48(-0.714) - 0.66]^2 = 1.005 \\
L_4(-0.714) &= [0.68(-0.714) - 0.65]^2 = 1.289 \\
L_5(-0.714) &= [0.83(-0.714) - 1.12]^2 = 2.933
\end{aligned}
$$

The total loss at for the parameter value $\beta = -0.714$ is

$$L(-0.714) = 0 + 0.264 + 1.005 + 1.289 + 2.933 = 5.492.$$

Compare this loss to that of the optimal parameter value $\beta = 1.21$ (which we will find shortly):

$$L(1.21) = 0.018 + 0.043 + 0.006 + 0.230 + 0.013 = 0.110.$$

In section 8.3 we will learn how to find parameter values that minimize the total loss across the entire dataset. Until then, remember that the "best" parameter values minimize the total loss, and the best parameters are rarely optimal for minimizing the loss of an individual point taken alone.

8.2.3 Loss vs. Error

The machine learning community almost exclusively uses the term loss to describe the penalized difference between a model's outputs and the training data. Although linear regression is a type of machine learning, the linear modeling field often refers to loss as "error" and quadratic loss as "squared error". Minimizing the quadratic loss is therefore known as "least squares" estimation. All of these terms are correct, and you should be familiar with both conventions. We have decided to use loss exclusively throughout the book for two reasons.

1. **Consistency**. One of our goals is to unify the presentation of machine learning and draw parallels between linear and nonlinear models. Subsequent chapters on machine learning use "loss" as it is standard in those fields, and we want to avoid duplicate terms.

2. **Clarity**. Some students find it difficult to distinguish the many types of error in linear models. There is error in the measured training data, and the parameter estimates have an associated standard error. Referring to the loss as "error" only adds to the confusion. Also, students often associate the term "error" with uncertainty, which is not a correct interpretation of a model's loss. Models have loss not because their predictions are uncertain (at least for deterministic models), but because they simplified representations of the real world. We like the term "loss" as a reminder that switching from data to a model causes an inherent loss of information.

All nomenclature is preference, and there is no unique solution. We will use margin notes to remind the reader of alternative names.

8.3 Fitting Linear Models

The total loss function provides a quantitative measure of how well our model fits the training data. There are three steps for minimizing the total loss of a linear model.

1. Choose a model that you think explains the relationship between inputs (x) and outputs (y). The models should contain unknown parameters (β) that

you will fit to a set of observations. **The model should be linear with respect to the parameters (β). It does not need to be linear with respect to the inputs (x) or outputs (y).** We will discuss the linearity of models in Chapter 9.

2. To find values for the unknown parameters (β), we will minimize the total loss between the observed outputs (y^{true}) and the outputs predicted from the model
$$\min_{\beta} \sum_{\text{data}} L_i(\beta)$$
or, for the specific case when we choose the quadratic loss,
$$\min_{\beta} \sum_{i} \left(y_i^{\text{pred}} - y_i^{\text{true}} \right)^2 .$$
Substitute the model you selected in Step 1 in place of y^{pred} in the above minimization.

3. Minimize the function by taking the derivative of the total loss and setting it equal to zero. Solve for the unknown parameters β. You should also check that your solution is a minimum, not a maximum or inflection point by checking that the second derivative is positive.

8.3.1 Single Parameter, Constant Models: $y = \beta_0$

The simplest linear model has only a single parameter and no dependence on the inputs:
$$y^{\text{pred}} = \beta_0.$$
This might seem like a silly model. Given an input observation x^{true}, the model ignores the input and predicts that y^{pred} will always be equal to β_0. For example, imagine we are predicting someone's height (y^{pred}) given their age (x^{true}). Rather than make a prediction based on the person's age, we simply guess the same height for everyone (β_0).

Regardless of the utility of such a simple model, let's fit it to a set of n pairs of observations ($x_i^{\text{true}}, y_i^{\text{true}}$). We've already completed Step 1 by choosing the model $y^{\text{pred}} = \beta_0$. We begin Step 2 by writing our objective: to choose a value for β_0 using the n observations that minimizes the total loss:
$$\min_{\beta_0} \sum_{i=1}^{n} \left(y_i^{\text{pred}} - y_i^{\text{true}} \right)^2 .$$

Now we substitute our model in place of y^{pred}, making our objective

$$\min_{\beta_0} \sum_{i=1}^{n} \left(\beta_0 - y_i^{\text{true}}\right)^2 .$$

To minimize the loss we find where the derivative of the total loss **with respect to the parameter** β_0 is zero.

Remember that the derivative of a sum is the sum of the derivative.

Also, the sum

$$\sum_{i}^{n} C = nC$$

for any value C that does not depend on i.

$$\begin{aligned}
\frac{d}{d\beta_0}\left(\sum_{i=1}^{n}(\beta_0 - y_i^{\text{true}})^2\right) &= 0\\
\sum_{i=1}^{n}\left(\frac{d}{d\beta_0}(\beta_0 - y_i^{\text{true}})^2\right) &= 0\\
\sum_{i=1}^{n} 2(\beta_0 - y_i^{\text{true}})(1) &= 0\\
2\sum_{i=1}^{n}(\beta_0 - y_i^{\text{true}}) &= 0\\
\sum_{i=1}^{n}\beta_0 - \sum_{i=1}^{n} y_i^{\text{true}} &= 0\\
n\beta_0 - \sum_{i=1}^{n} y_i^{\text{true}} &= 0
\end{aligned}$$

We can rearrange the final equation and discover that the optimal value for the parameter β_0 is

$$\beta_0 = \frac{1}{n}\sum_{i=1}^{n} y_i^{\text{true}} = \text{mean}[y^{\text{true}}].$$

If our strategy is to always guess the same output value (β_0), the best value to guess is the mean of the observed outputs y^{true}. Said another way, the mean is the best fit of a constant model to a set of data. If we need to represent a set of numbers with a single number, choosing the mean minimizes the quadratic loss.

The quadratic loss is also called the "squared error", so the least squares estimator minimizes is also said to minimize the "sum of squares".

A couple interesting things come from this result. First, now you know where the mean comes from. It is the *least squares* estimate for a set of points. (The phase "least squares" is a convenient way of saying "minimizes the sum of the quadratic loss".) Second, the mean does not minimize the absolute value of the residuals —

this is a common misconception! If we repeated the same calculation using the absolute loss instead of the quadratic loss we would discover that the least absolute estimator for a set of numbers is the median, not the mean.

Going further, if we make our loss function binary (the loss is zero if $y^{\text{pred}} = y^{\text{true}}$ and one otherwise), the best estimator is called the *mode* — the most frequent value in the set of observed outputs.

8.3.2 Two Parameter Models: $y = \beta_0 + \beta_1 x$

Let's fit a more complicated model that uses the observed inputs x^{true} when making output predictions y^{true}. Our model has the form

$$y^{\text{pred}} = \beta_0 + \beta_1 x^{\text{true}}$$

with two unknown parameters β_0 and β_1. This is a linear model with respect to the parameters. We discovered earlier that the functions of the form $y = \beta_0 + \beta_1 x$ are not linear with respect to x (they are affine), but remember that x^{true} is not an independent variable in the model. It is a known, observed value — a constant. The unknowns in a linear model are the parameters, not y or x.

Now that we've chosen our model, we write our objective: to minimize the total quadratic loss.

$$\min_{\beta_0,\beta_1} \sum_{i=1}^{n} \left(y_i^{\text{pred}} - y_i^{\text{true}} \right)^2$$

Notice we are minimizing over both parameters β_0 and β_1. Substituting our model for the value y^{pred} yields

$$\min_{\beta_0,\beta_1} \sum_{i=1}^{n} \left(\beta_0 + \beta_1 x_i^{\text{true}} - y_i^{\text{true}} \right)^2$$

The total loss is minimized when the derivatives with respect to both β_0 and β_1 are zero. Let's start by taking the derivative or the loss with respect to β_0.

We are using partial derivatives since our loss is a function of more than one unknown parameter.

$$\begin{aligned}\frac{\partial}{\partial\beta_0}\sum_{i=1}^{n}\left(\beta_0+\beta_1 x_i^{\text{true}}-y_i^{\text{true}}\right)^2 &= \sum_{i=1}^{n}\frac{\partial}{\partial\beta_0}\left(\beta_0+\beta_1 x_i^{\text{true}}-y_i^{\text{true}}\right)^2\\ &= 2\sum_{i=1}^{n}\left(\beta_0+\beta_1 x_i^{\text{true}}-y_i^{\text{true}}\right)\\ &= 2\left(\sum_{i=1}^{n}\beta_0+\sum_{i=1}^{n}\beta_1 x_i^{\text{true}}-\sum_{i=1}^{n}y_i^{\text{true}}\right)\\ &= 2\left(n\beta_0+\beta_1\sum_{i=1}^{n}x_i^{\text{true}}-\sum_{i=1}^{n}y_i^{\text{true}}\right)\end{aligned}$$

We set this derivative equal to zero and solve for β_0.

We call β_0 (the affine parameter in a linear model) the *grand mean* since it equals the mean of the outputs when all inputs are zero.

$$\begin{aligned}\beta_0 &= \frac{1}{n}\sum_{i=1}^{n}y_i^{\text{true}}-\beta_1\frac{1}{n}\sum_{i=1}^{n}x_i^{\text{true}}\\ &= \text{mean}[y^{\text{true}}]-\beta_1\text{mean}[x^{\text{true}}]\end{aligned}$$

We see that β_0 depends on the mean input, the mean output, and the parameter β_1. Let's substitute our value for β_0 into the total loss function.

$$\begin{aligned}\sum_{i=1}^{n}\left(\beta_0+\beta_1 x_i^{\text{true}}-y_i^{\text{true}}\right)^2 &= \sum_{i=1}^{n}\left(\text{mean}[y^{\text{true}}]-\beta_1\text{mean}[x^{\text{true}}]+\beta_1 x_i^{\text{true}}-y_i^{\text{true}}\right)^2\\ &= \sum_{i=1}^{n}\Big(\beta_1\left(x_i^{\text{true}}-\text{mean}[x^{\text{true}}]\right)-\left(y_i^{\text{true}}-\text{mean}[y^{\text{true}}]\right)\Big)^2\end{aligned}$$

Now we find the optimal value for the parameter β_1. First we differentiate the total loss with respect to β_1.

$$\frac{\partial}{\partial\beta_1}\sum_{i=1}^{n}\Big(\beta_1\left(x_i^{\text{true}}-\text{mean}[x^{\text{true}}]\right)-\left(y_i^{\text{true}}-\text{mean}[y^{\text{true}}]\right)\Big)^2$$

$$
\begin{aligned}
&= \sum_{i=1}^{n} \frac{\partial}{\partial \beta_1}\Big(\beta_1\left(x_i^{\text{true}} - \text{mean}[x^{\text{true}}]\right) - \left(y_i^{\text{true}} - \text{mean}[y^{\text{true}}]\right)\Big)^2 \\
&= 2\sum_{i=1}^{n} \Big(\beta_1\left(x_i^{\text{true}} - \text{mean}[x^{\text{true}}]\right) - \left(y_i^{\text{true}} - \text{mean}[y^{\text{true}}]\right)\Big)\left(x_i^{\text{true}} - \text{mean}[x^{\text{true}}]\right) \\
&= 2\sum_{i=1}^{n} \Big(\beta_1\left(x_i^{\text{true}} - \text{mean}[x^{\text{true}}]\right)^2 - \left(x_i^{\text{true}} - \text{mean}[x^{\text{true}}]\right)\left(y_i^{\text{true}} - \text{mean}[y^{\text{true}}]\right)\Big)
\end{aligned}
$$

We set the derivative equal to zero and solve for the parameter β_1.

$$
\beta_1 = \frac{\sum_{i=1}^{n} \left(x_i^{\text{true}} - \text{mean}[x^{\text{true}}]\right)\left(y_i^{\text{true}} - \text{mean}[y^{\text{true}}]\right)}{\sum_{i=1}^{n} \left(x_i^{\text{true}} - \text{mean}[x^{\text{true}}]\right)^2}
$$

Let's try fitting the expression $y^{\text{pred}} = \beta_0 + \beta_1 x^{\text{true}}$ to the data from earlier in this chapter.

x^{true}	y^{true}
0.07	−0.05
0.16	0.40
0.48	0.66
0.68	0.65
0.83	1.12

First we calculate the means of the inputs and outputs.

$$
\begin{aligned}
\text{mean}[x^{\text{true}}] &= (1/5)(0.07 + 0.16 + 0.48 + 0.68 + 0.83) = 0.44 \\
\text{mean}[y^{\text{true}}] &= (1/5)(-0.05 + 0.40 + 0.66 + 0.65 + 1.12) = 0.56
\end{aligned}
$$

Now we can calculate the value for the parameter β_1. It's easiest to make a table.

x^{true}	y^{true}	$(x^{\text{true}} - \text{mean}[x^{\text{true}}])(y^{\text{true}} - \text{mean}[y^{\text{true}}])$	$(x^{\text{true}} - \text{mean}[x^{\text{true}}])^2$
0.07	−0.05	$(0.07 - 0.44)(-0.05 - 0.56) = 0.23$	$(0.07 - 0.44)^2 = 0.14$
0.16	0.40	$(0.16 - 0.44)(0.40 - 0.56) = 0.044$	$(0.16 - 0.44)^2 = 0.081$
0.48	0.66	$(0.48 - 0.44)(0.66 - 0.56) = 0.0037$	$(0.48 - 0.44)^2 = 0.0013$
0.68	0.65	$(0.68 - 0.44)(0.65 - 0.56) = 0.022$	$(0.68 - 0.44)^2 = 0.056$
0.83	1.12	$(0.83 - 0.44)(1.12 - 0.56) = 0.22$	$(0.83 - 0.44)^2 = 0.15$

$$\begin{aligned}\beta_1 &= \frac{\sum_{i=1}^{n}\left(x_i^{\text{true}} - \text{mean}[x^{\text{true}}]\right)\left(y_i^{\text{true}} - \text{mean}[y^{\text{true}}]\right)}{\sum_{i=1}^{n}\left(x_i^{\text{true}} - \text{mean}[x^{\text{true}}]\right)^2} \\ &= \frac{0.23 + 0.044 + 0.0037 + 0.022 + 0.22}{0.14 + 0.081 + 0.0013 + 0.056 + 0.15} \\ &= 1.21\end{aligned}$$

We can use the value of the parameter β_1 to find the other parameter β_0.

$$\begin{aligned}\beta_0 &= \text{mean}[y^{\text{true}}] - \beta_1 \text{mean}[x^{\text{true}}] \\ &= 0.56 - (1.21)(0.44) \\ &= 0.020\end{aligned}$$

According to our five observations, the best fit least squares estimate is

$$y = 0.020 + 1.21x.$$

This agrees well with our hypothesized relationship that $y = 1.2x$.

8.4 Matrix Formalism for Linear Models

You might be thinking that there has to be an easier method for fitting linear models. Finding formulae for the parameters is unwieldy, and the problem only worsens as the number of parameters grows. Fortunately, linear algebra can help.

Let's return to our two parameter model $y = \beta_0 + \beta_1 x$. Using the five data points from the previous section, we can write five linear equations, one for each point.

$$\begin{aligned}-0.05 &= \beta_0 + 0.07\beta_1 + \epsilon_1 \\ 0.40 &= \beta_0 + 0.16\beta_1 + \epsilon_2 \\ 0.66 &= \beta_0 + 0.48\beta_1 + \epsilon_3 \\ 0.65 &= \beta_0 + 0.68\beta_1 + \epsilon_4 \\ 1.12 &= \beta_0 + 0.83\beta_1 + \epsilon_5\end{aligned}$$

All we've done to write these equations is substituted the observed values for x and y and added an *residual term* (ϵ_i). Remember that each observation is imprecise, so the observed value of y will never exactly equal the predicted value $\beta_0 + \beta_1 x$. Since our equations must be exact, we add a term to each equation to hold the residual between the predicted and observed values. The same equations can be written in

The parameters β_0 and β_1 are the same for every equation, but each equation has its own residual term. In some fields the residual terms are called "errors".

matrix form as

$$\begin{pmatrix}-0.05\\0.40\\0.66\\0.65\\1.12\end{pmatrix} = \begin{pmatrix}1 & 0.07\\1 & 0.16\\1 & 0.48\\1 & 0.68\\1 & 0.83\end{pmatrix}\begin{pmatrix}\beta_0\\\beta_1\end{pmatrix} + \begin{pmatrix}\epsilon_1\\\epsilon_2\\\epsilon_3\\\epsilon_4\\\epsilon_5\end{pmatrix}.$$

Or, more succinctly,

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}.$$

There are several noteworthy things about the above expression.

- The variable $\mathbf{y}$ is a vector of the outputs (responses), $\boldsymbol{\epsilon}$ is a vector of residuals, and $\boldsymbol{\beta}$ is a vector of the unknown parameters.
- The inputs (or predictor variables) form a matrix $\mathbf{X}$ called the *model matrix*.
- The first column in $\mathbf{X}$ is all ones. This column corresponds to the constant parameter in the model, β_0.
- The unknowns in the equation are the parameters in the vector $\boldsymbol{\beta}$, not the values in the matrix $\mathbf{X}$. The values in $\mathbf{X}$ are known inputs from our dataset.

The residuals $\boldsymbol{\epsilon}$ are also unknown, but we do not solve for these explicitly. Once we have estimates for the parameters, we can calculate the residuals using $\boldsymbol{\epsilon} = \mathbf{y} - \mathbf{X}\boldsymbol{\beta}$.

Fitting a linear model involves finding a set of values for the vector $\boldsymbol{\beta}$. There are a few complications to solving the linear system $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$. First, our goal is not to find just any values for the parameters in $\boldsymbol{\beta}$, but to find the values that minimize the square of the residual terms in $\boldsymbol{\epsilon}$ (i.e. the least squares solution). Second, the matrix $\mathbf{X}$ is almost never square. We often have more rows (observations) that we have columns (parameters) to compensate for the noise in our measurements.

Fortunately, there is a tool from matrix theory — the *pseudoinverse* — that overcomes both these difficulties. The least squares solution to the problem $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$ is

$$\boldsymbol{\beta} = \mathbf{X}^+\mathbf{y}$$

where the matrix $\mathbf{X}^+$ is the pseudoinverse of the matrix $\mathbf{X}$.

8.5 The Pseudoinverse

Recall that a matrix is invertible if and only if it is square and full rank. For linear models the model matrix $\mathbf{X}$ is almost never square, so the matrix inverse $\mathbf{X}^{-1}$ does not exist. However, all nonsquare matrices have a pseudoinverse that somewhat approximates the behavior of the true inverse. Below are some properties of the pseudoinverse.

The pseudoinverse is also called the Moore-Penrose inverse.

1. The pseudoinverse $\mathbf{X}^+$ exists for *any* matrix $\mathbf{X}$.
2. The pseudoinverse of a matrix is unique.
3. If a square matrix $\mathbf{X}$ has a true inverse $\mathbf{X}^{-1}$, then $\mathbf{X}^+ = \mathbf{X}^{-1}$.
4. The pseudoinverse of a pseudoinverse is the original matrix: $(\mathbf{X}^+)^+ = \mathbf{X}$.
5. If $\dim(\mathbf{X}) = m \times n$, then $\dim(\mathbf{X}^+) = n \times m$.
6. It is **not** generally true that $\mathbf{X}^+\mathbf{X} = \mathbf{X}\mathbf{X}^+ = \mathbf{I}$. However, if $\mathbf{X}$ has full column rank then $\mathbf{X}^+\mathbf{X} = \mathbf{I}$, and if $\mathbf{X}$ has full row rank then $\mathbf{X}\mathbf{X}^+ = \mathbf{I}$.
7. If a matrix has only real entries, then so does its pseudoinverse.
8. If a matrix $\mathbf{X}$ is full rank, then $\mathbf{X}^+ = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}$.

It is always true that $\mathbf{X}^+\mathbf{X}\mathbf{X}^+ = \mathbf{X}^+$ and $\mathbf{X}\mathbf{X}^+\mathbf{X} = \mathbf{X}$; this is part of the definition of the pseudoinverse, along with the requirements that $(\mathbf{X}\mathbf{X}^+)^\mathsf{T} = \mathbf{X}\mathbf{X}^+$ and $(\mathbf{X}^+\mathbf{X})^\mathsf{T} = \mathbf{X}^+\mathbf{X}$.

To understand the final property, consider the linear system

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta}$$

Let's multiply both sizes by the matrix $\mathbf{X}^\mathsf{T}$.

$$\mathbf{X}^\mathsf{T}\mathbf{y} = \mathbf{X}^\mathsf{T}\mathbf{X}\boldsymbol{\beta}$$

We know that the matrix $\mathbf{X}$ is not square; however, the matrix $\mathbf{X}^\mathsf{T}\mathbf{X}$ is always square. Since $\mathbf{X}^\mathsf{T}\mathbf{X}$ is square, it is invertible provided it is full rank, which is guaranteed if $\mathbf{X}$ has full column rank (see §7.1.1). Assuming $(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}$ exists, let's multiply both sides of our equation by it.

If matrix $\mathbf{X}$ has m rows and n columns, the matrix $\mathbf{X}^\mathsf{T}\mathbf{X}$ has n rows and n columns.

$$(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y} = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{X}\boldsymbol{\beta}$$

Look carefully at the righthand side. We have the matrix $\mathbf{X}^\mathsf{T}\mathbf{X}$ multiplied by its inverse, $(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}$. This is equal to the identity matrix, so

$$(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y} = \boldsymbol{\beta}$$

We have solved the system $\mathbf{y} = \mathbf{X}\boldsymbol{\beta}$ for the vector $\boldsymbol{\beta}$, so the quantity $(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}$ on the lefthand side must be the pseudoinverse of the matrix $\mathbf{X}$.

Let's use pseudoinversion to solve our example model:

$$\begin{pmatrix} -0.05 \\ 0.40 \\ 0.66 \\ 0.65 \\ 1.12 \end{pmatrix} = \begin{pmatrix} 1 & 0.07 \\ 1 & 0.16 \\ 1 & 0.48 \\ 1 & 0.68 \\ 1 & 0.83 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \end{pmatrix}$$

Using MATLAB's `pinv` function we can calculate the pseudoinverse of the model matrix

$$\mathbf{X}^+ = \begin{pmatrix} 0.59 & 0.50 & 0.16 & -0.046 & -0.20 \\ -0.88 & -0.67 & 0.084 & 0.55 & 0.91 \end{pmatrix}$$

and find the least squares estimates for the parameters

$$\boldsymbol{\beta} = \mathbf{X}^+\mathbf{y} = \begin{pmatrix} 0.59 & 0.50 & 0.16 & -0.046 & -0.20 \\ -0.88 & -0.67 & 0.084 & 0.55 & 0.91 \end{pmatrix} \begin{pmatrix} -0.05 \\ 0.40 \\ 0.66 \\ 0.65 \\ 1.12 \end{pmatrix} = \begin{pmatrix} 0.020 \\ 1.21 \end{pmatrix}$$

Again, we see that $\beta_0 = 0.020$ and $\beta_1 = 1.21$.

8.5.1 Calculating the Pseudoinverse

Our new formula for the pseudoinverse ($\mathbf{X}^+ = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}$) gives us intuition about solving nonsquare linear systems — we are actually solving a related system involving the special matrix $\mathbf{X}^\mathsf{T}\mathbf{X}$. This form of the pseudoinverse requires calculating a matrix inverse, which we have all sworn never to do except in dire situations. The function `pinv` in MATLAB uses a matrix decomposition method to find pseudoinverses, which we will discuss in section 19.3.2. Matrix decomposition methods also work on rank deficient matrices since they do not require a true matrix inversion.

8.6 Dimensions of the Model Matrix

The pseudoinverse of $\mathbf{X}$ is part of the least squares solution for the linear model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$. Each row in $\mathbf{X}$ is an observation and each column corresponds to an unknown parameter. If $\mathbf{X}$ is square, we have one observation per parameter. We know that the pseudoinverse of a square, invertible matrix is identical to the ordinary inverse. We also know that linear systems with square coefficient matrices have a unique solution. There is no room to find a solution that minimizes the loss when there is only a single unique solution. We can fit all of the parameters, but as we will see later, we have no information about how well we did minimizing the loss.

If we have more observations than parameters ($\mathbf{X}$ has more rows than columns), the extra information in the observations can be used to estimate how well our solution minimizes the quadratic loss. The extra degrees of freedom can quantify our confidence in the model.

Finally, our system is *underdetermined* if we have fewer observations than parameters. The search space for parameters is simply too large, and we often cannot find a meaningful solution. Fitting these models requires special tools that we will discuss in Chapter 14.

Chapter 9

Building Regression Models

In Chapter 8 we outlined a framework for fitting linear models to data. Linear models are enormously flexible and can be applied to many problems in science and engineering. In this chapter we discuss how to formulate, solve, and interpret several types of linear models. The accompanying MATLAB workbook describes how to build and solve linear models using matrices and a formula-based interface.

A few notes on notation before we begin. Linear models can be expressed using either standard algebra or vector algebra. Consider a model with two predictor variables (inputs). We can write this model as

$$y_i = \beta_1 x_{1,i} + \beta_2 x_{2,i} + \epsilon_i$$

or using vector notation

$$\mathbf{y} = \beta_1 \mathbf{x}_1 + \beta_2 \mathbf{x}_2 + \boldsymbol{\epsilon}.$$

We prefer the latter form since it is simpler and it emphasizes that each predictor variable is a vector. These vectors are collected into the model matrix, which is pseudoinverted to solve the linear model. Notice how we've dropped the "pred" and "true" labels from our equations. In this chapter it should be clear that the model is always fit using the true, measured values.

9.1 The Intercept

The model's intercept is the only parameter not associated with an input. For the linear model

$$\mathbf{y} = \beta_0 + \beta_1 \mathbf{x}_1 + \beta_2 \mathbf{x}_2 + \boldsymbol{\epsilon}$$

the coefficient β_i is associated with input $\mathbf{x}_i$, so by convention we call the intercept β_0 since it has no associated input. The above model is a slight abuse of notation. The lefthand side ($\mathbf{y}$) is a vector, and all the non-intercept terms on the righthand side ($\beta_1\mathbf{x}_1$, $\beta_2\mathbf{x}_2$, and $\boldsymbol{\epsilon}$) are also vectors. The intercept β_0 is a scalar, and we have not defined an addition operator between scalars and vectors. When writing a linear model, we assume that the intercept multiplies an implicit vector of ones, so the real model is

$$\mathbf{y} = \beta_0\mathbf{1} + \beta_1\mathbf{x}_1 + \beta_2\mathbf{x}_2 + \boldsymbol{\epsilon}.$$

Although we usually omit the vector $\mathbf{1}$, it is a helpful reminder that models with an intercept have a column of ones in their model matrix. Rewriting the above equation in vector notation gives

Be careful to distinguish between $\mathbf{1}$, a vector of ones, and $\mathbf{I}$, the identity matrix.

$$\mathbf{y} = \begin{pmatrix}\mathbf{1} & \mathbf{x}_1 & \mathbf{x}_2\end{pmatrix}\begin{pmatrix}\beta_0\\ \beta_1\\ \beta_2\end{pmatrix} + \boldsymbol{\epsilon}.$$

The model matrix for this model is $\mathbf{X} = \begin{pmatrix}\mathbf{1} & \mathbf{x}_1 & \mathbf{x}_2\end{pmatrix}$. If we wanted to fit a model without an intercept, we would write

$$\mathbf{y} = \begin{pmatrix}\mathbf{x}_1 & \mathbf{x}_2\end{pmatrix}\begin{pmatrix}\beta_1\\ \beta_2\end{pmatrix} + \boldsymbol{\epsilon}$$

and the model matrix would be $\mathbf{X} = \begin{pmatrix}\mathbf{x}_1 & \mathbf{x}_2\end{pmatrix}$.

When all of the input variables $\mathbf{x}_i$ are zero, the model's output is the value of the intercept ($y^{\text{pred}} = \beta_0$). Thus, whether or not to include an intercept in your model depends on the output you would expect when all inputs are zero. For example, imagine you are building a model that predicts the height of a plant based on the number of hours of sunlight it receives. A plant that never sees the sun should not grow, so it would be reasonable to exclude an intercept from the model. If instead you build a model that relates plant height to amount of fertilizer added, you would include an intercept since a plant that receives no fertilizer could still grow with only the nutrients in the soil. Most times your models will include an intercept, and software packages like Matlab add them by default. However, you should consider if the intercept is needed and if it is reasonable for the output of the model to be nonzero when all of the input variables are zero.

9.2 Analyzing Models

There are two reasons to build models. The first is *prediction* — estimating a new output for inputs that were not included in the training set. For example, a set of clinical measures (blood pressure, resting pulse, white cell count) could be used to train a model that predicts a person's risk of heart attack. After training is finished, we can use the model for prediction by using a new person's clinical measures as inputs.

The second reason to build a model is *inference*. Inference focuses on how a model makes its predictions. For a heart attack model, we can ask which of the clinical measures are important for determining risk, or how much changing an input would raise or lower the model's prediction. It may surprise you how often we build models that are only used for inference and not for prediction. You may have seen studies that link coffee consumption to blood pressure. Such studies are analyzed by building a model that predicts blood pressure based on the number of cups of coffee consumed. This model has low predictive value; few people would need to predict what their blood pressure will be if they drank a certain amount of coffee. But the model has inferential value. Examining the parameters of the model can tell us how large of an effect coffee has on blood pressure and if this effect is statistically significant.

9.2.1 Prediction Intervals

The outputs of a model are only predictions. They are never exactly correct, and we would like some estimate of their accuracy. We can use our training data to assess our model's predictive power. First, we fit the model by finding parameter estimates that minimize the loss function. Model fitting uses the observed inputs and outputs ($\mathbf{x}^{\text{true}}$ and $\mathbf{y}^{\text{true}}$). Next, we feed the training inputs $\mathbf{x}^{\text{true}}$ back into the model to calculate the predicted output $\mathbf{y}^{\text{pred}}$. If the model fit the training data exactly, the predicted outputs $\mathbf{y}^{\text{pred}}$ would perfectly match the true outputs $\mathbf{y}^{\text{true}}$. In almost all cases, the predicted and true outputs will disagree, and we can use the discrepancy to estimate the model's accuracy.

A common measure of accuracy is the *root-mean-squared-error*, or RMSE. If the training data included n observations, then the model's RMSE is

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i^{\text{pred}} - y_i^{\text{true}}\right)^2}$$

The RMSE formula is best understood from the inside out. The squared error of a single prediction is $(y_i^{\text{pred}} - y_i^{\text{true}})^2$. Summing these errors and dividing by n gives the mean squared error. Unfortunately, the mean squared error is difficult to interpret, in part because the units of this error are the output's units squared. It is easiest if we transform the mean squared error back into output's units by taking

We prefer the term "loss" rather than "error" when training models, but error makes sense when measuring a model's accuracy.

the square root. Hence, the RMSE is the square root of the mean of the squared errors.

The RMSE is an estimate of the standard deviation of the prediction errors. When reporting a prediction when can include the RMSE to give the reader a feel for the model's accuracy. If a model predicts a patient's resting pulse rate as 68 bpm and the model's RMSE is 12 bpm, then we should report the prediction as 68 ± 12 bpm. It is often assumed that the model's prediction errors are normally distributed, so the RMSE can be used to estimate a *prediction interval*. The 95% prediction interval spans twice the RMSE on either side of the prediction. For our pulse example, the 95% prediction interval is $[68-2\times12,\ 68+2\times12] = [44, 92]$ bpm. Remember that if we transformed the output of a model (see §9.4), we need to undo the transformation for both the model prediction and the RMSE to get back to the original units.

9.2.2 Effect Sizes and Significance

Let's switch gears from prediction to inference. The goal of inference is to understand how the inputs relate to the output. You can think of model building as a multivariable hypothesis test. After collecting data, we construct a predictive model based on a set of inputs. We might discover that some of the inputs may not be useful for predicting the output, so model fitting in essence tests the strength of the relationship between each input and the output.

Model inference involves asking two questions about each of the models inputs.

1. How large of an effect does this input have on the output?
2. How confident are we in our estimate of this effect?

The first question concerns the *effect size* of the input. The effect size is another name for the coefficient that we estimate for the input. Consider the two input model

$$\mathbf{y} = 1.2 - 3.6\mathbf{x}_1 + 0.8\mathbf{x}_2 + \epsilon.$$

The effect size for input $\mathbf{x}_1$ is -3.6, and the effect size of input $\mathbf{x}_2$ is 0.8. The effect size quantifies the sensitivity of the output with respect to the input. Increasing $\mathbf{x}_1$ by one will *decrease* the output $\mathbf{y}$ by 3.6 (since the effect size is negative). Increasing $\mathbf{x}_2$ by one will *increase* the output by 0.8.

Increasing a variable by one is commonly called a "unit increase". This terminology is unrelated to the actual units of the variable, e.g. kilograms, seconds, etc.

Effect sizes have units. If the previous model predicted pulse rate in bpm and the input $\mathbf{x}_1$ was a person's age in years, the effect size would be -3.6 bpm/year. It is important to include any units when reporting effect sizes.

The effect size of an input answers the first inferential question ("how large of an effect does an input have on the output?"). Models estimate effects using inherently noisy data, so our estimates are never exact. Most statistical modeling packages will report a standard error and a p-value for each effect size. The standard error can be used to construct a confidence interval for the effect size. For example, the 95% confidence interval spans two standard errors on each size of the effect size. Often we want to know if an effect size is significantly different from zero. If an effect size is indistinguishable from zero (i.e. if the confidence interval includes zero), then we cannot reject the idea that the observed effect size is simply an artifact of the uncertainty in our data. Said another way, if an effect size is indistinguishable from zero, we should not be surprised if we refit the data with a new set of observations (of the same size) and find that the effect has "disappeared".

The t-test is a common method for testing if an effect size can be distinguished from zero. The p-value reported for an effect size is the p-value from a t-test. A threshold of $p < 0.05$ is frequently used to separate "significant" from "not significant" effects. If the p-value exceeds our threshold, we are not confident that the effect size is nonzero.

Be careful to distinguish between an effect size and its significance. The p-value is only related to the precision of our estimate; it has no bearing on the magnitude of the effect. A very small p-value indicates that an effect size is statistically distinguishable from zero, but the *practical significance* of the effect could be small. As an interesting example, consider a study released by the online dating site eHarmony (Cacioppo, et al., *Proc Nat Acad Sci*, 2013). The study reports a statistically significant increase in marital satisfaction among couple who met online vs. by other venues. While the results were statistically significant, the actual increase in marital satisfaction was only 2.9%. Even though the means of the two groups differed by less than 3%, the enormous sample size (19,131 couples) made the result statistically — but not practically — significant.

9.2.3 Degrees of Freedom

A model's parameters must be estimated from the observed data. We know from Chapter 7 that a linear system requires the information from one observation to estimate each unknown. Let's assume we use m observations to fit a model with n unknown parameters. If $m = n$, we have just enough information to estimate each parameter; however, we have no information left over to assess the accuracy of our predictions or the effect sizes. If we have more observations than unknown parameters ($m > n$), then we can use the remaining observations to build confidence intervals. We call the number of "extra" observations the *degrees of freedom*. A model with n parameters fit to m observations has $m - n$ degrees of freedom. Note that the intercept is an unknown parameter, so it should be included the parameter count. We need at least one degree of freedom to estimate prediction intervals or the significance of effect sizes. As the degrees of freedom increase, the RMSE and the standard errors of the effect sizes decrease.

9.3 Curvilinear Models

So far we've focused on models that are linear combinations of a set of inputs. It is important to remember that the parameters, not the inputs, are the unknowns in our models. We can apply any transformation we want to the inputs and retain a linear model. Consider the cubic polynomial model

$$\mathbf{y} = \beta_0 + \beta_1 \mathbf{x} + \beta_2 \mathbf{x}^2 + \beta_3 \mathbf{x}^3 + \boldsymbol{\epsilon}.$$

We define vector exponentiation as an elementwise operation, so

$$\mathbf{x}^2 = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \vdots \\ x_n^2 \end{pmatrix}.$$

This model is still linear with respect to its parameters, as we see when we rewrite it in matrix form.

$$\mathbf{y} = \begin{pmatrix} \mathbf{1} & \mathbf{x} & \mathbf{x}^2 & \mathbf{x}^3 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} + \boldsymbol{\epsilon}$$

This model has only a single input — the vector $\mathbf{x}$. We've created additional *features* for the model by squaring and cubing the entries in $\mathbf{x}$. Each of the transformed features appears as a separate column in the model matrix.

We can fit any polynomial using linear regression by transforming the input. These transformations are usually hidden "under the hood" in spreadsheet programs that let you perform polynomial regression. Spreadsheets allow their users to specify the degree of the polynomial. When a user clicks to increase the degree, the spreadsheet adds a new column to the model matrix and refits the model.

Any transformation, not just exponentiation, can be applied to the inputs of a model. Logarithms, square roots, and mean-centering (subtracting the mean of $\mathbf{x}$ from every entry) are common transformations. The outputs $\mathbf{y}$ can also be transformed. A model that includes transformed inputs or outputs is said to be *curvilinear* since it remains linear with respect to the parameters but the input/output relationship is no longer "straight".

9.4 Linearizing Models

We are able to fit curvilinear models using linear algebra because the models remained linear with respect to the parameters. Some models do not appear to be linear but can be made linear by transformation. For example, bacteria grow exponentially, so their growth can be describe by the model

$$N(t) = N_0 e^{\mu t}$$

where $N(t)$ is the number of bacteria at time t, N_0 is the initial number of bacteria, and μ is the growth rate. The parameters in this model are N_0 and μ, and the exponential growth model is not linear with respect to μ. Let's transform the model by taking the natural logarithm of both sides.

$$\begin{aligned}\log(N(t)) &= \log(N_0 e^{\mu t}) \\ &= \log(N_0) + \mu t\end{aligned}$$

Now let's make a few substitutions. Our output variable is $y = \log(N(t))$. The values $N(t)$ are all known, so we simply transform them with the natural logarithm before fitting the model. We will also set $\beta_0 = \log(N_0)$ and $\beta_1 = \mu$, making our final linear model

$$\mathbf{y} = \beta_0 + \beta_1 \mathbf{t} + \boldsymbol{\epsilon}.$$

We can use linear regression to estimate the parameters β_0 and β_1. These parameters can be transformed back into estimates of N_0 and μ:

$$\beta_0 = \log(N_0) \Rightarrow N_0 = e^{\beta_0}$$

$$\mu = \beta_1$$

The Michaelis-Menten equation is another nonlinear function that can be linearized. Recall that the velocity v of a reaction is a function of substrate concentration $[S]$ and two parameters, V_{max} and K_m.

$$v = \frac{V_{\text{max}}[S]}{K_m + [S]}$$

This equation is nonlinear with respect to the parameters V_{max} and K_m, but it can be linearized by inverting both sides.

This linearization of the Michaelis-Menten equations is called Lineweaver-Burk or double reciprocal method.

$$\begin{aligned}\frac{1}{v} &= \frac{K_m + [S]}{V_{\text{max}}[S]} \\ &= \frac{K_m}{V_{\text{max}}[S]} + \frac{[S]}{V_{\text{max}}[S]} \\ &= \frac{K_m}{V_{\text{max}}}\frac{1}{[S]} + \frac{1}{V_{\text{max}}}\end{aligned}$$

Using $1/[S]$ as the input $\mathbf{x}$ and $1/v$ as the output $\mathbf{y}$, the two parameters of the linear model are $\beta_0 = 1/V_{\text{max}}$ and $\beta_1 = K_m/V_{\text{max}}$.

Not every nonlinear model can be transformed into a linear one. Logarithmic transformations sometimes work for multiplicative models and models with parameters as exponents. Linearizing models allows us to find parameter values using linear regression, but there is a downside. The linearization can skew the distribution of our dataset and amplify measurement errors. For the linearized Michaelis-Menten equation, the output variable is $1/v$, the inverse of the reaction velocity. When the velocity v is very small, the transformed variable $1/v$ becomes very large. Any uncertainty in our velocity measurements will be amplified when v is small. An alternative approach is to avoid linearization and fit parameters directly to the nonlinear model. We will discuss nonlinear fitting methods in Part II.

9.5 Interactions

So far the inputs to our models are additive. Consider again the two input linear model

$$\mathbf{y} = 1.2 - 3.6\mathbf{x}_1 + 0.8\mathbf{x}_2 + \boldsymbol{\epsilon}.$$

For every unit increase in $\mathbf{x}_1$, the output $\mathbf{y}$ decreases by 3.6, regardless of the value of the other input $\mathbf{x}_2$. In modeling terms, we say there is no *interaction* between inputs $\mathbf{x}_1$ and $\mathbf{x}_2$. If we believe that the inputs do interact, that is, if the effect of changing one input depends on the value of the other input, then we can add a term to describe this interaction. The term we add is the product of the two inputs, and this term receives its own parameter. For example, a standard two-input linear model (without interaction) is

$$\mathbf{y} = \beta_0 + \beta_1\mathbf{x}_1 + \beta_2\mathbf{x}_2 + \boldsymbol{\epsilon}.$$

A similar model with an interaction term between $\mathbf{x}_1$ and $\mathbf{x}_2$ is

$$\mathbf{y} = \beta_0 + \beta_1\mathbf{x}_1 + \beta_2\mathbf{x}_2 + \beta_{12}\mathbf{x}_1{:}\mathbf{x}_2 + \boldsymbol{\epsilon}.$$

By convention we call the interaction parameter β_{12} since it measures the interaction effect between inputs $\mathbf{x}_1$ and $\mathbf{x}_2$. (It is read "1-2", not as the number "12".) We've used the colon to represent elementwise multiplication between the vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, as this is the syntax used to specify linear models in many software packages. Adding interaction terms does not violate the linearity of the model. Remember that the inputs $\mathbf{x}_i$ are known, so multiplying them together yields yet another known quantity.

The elementwise product of two vectors $\mathbf{a}$ and $\mathbf{b}$ is

$$\mathbf{a}{:}\mathbf{b} = \begin{pmatrix} a_1b_1 \\ a_2b_2 \\ \vdots \\ a_nb_n \end{pmatrix}.$$

The interaction effect β_{12} quantifies the effects of the inputs that cannot be explained by either input alone. Notice that our interaction model retains the independent terms $\beta_1\mathbf{x}_1$ and $\beta_2\mathbf{x}_2$. The interaction term $\beta_{12}\mathbf{x}_1{:}\mathbf{x}_2$ only describes the "above and beyond" effects. If the inputs $\mathbf{x}_1$ and $\mathbf{x}_2$ are strictly additive, then the estimate of the interaction effect β_{12} will be zero. In biomedical terms, you can think of the interaction as the "synergy" between the inputs.

You might be wondering why the interaction between two variables is measured by their product, rather than some other function. There are two explanations, and you are free to choose the one you like best.

1. The interaction depends on both inputs, so it should not have an effect when either input is missing (zero). The term $\beta_{12}\mathbf{x}_1{:}\mathbf{x}_2$ is the simplest expression that is zero when either $\mathbf{x}_1$ or $\mathbf{x}_2$ is zero.

2. Alternatively, we could assume that the effect size of input $\mathbf{x}_1$ depends on the value of $\mathbf{x}_2$. We could write a model

   $$y = (\beta_1 + \beta_{12}\mathbf{x}_2){:}\mathbf{x}_1 + \beta_2\mathbf{x}_2$$

   where the effect size of x_1 is not a single parameter but instead an expression that depends on $\mathbf{x}_2$. The coefficient β_{12} adjusts the coefficient of $\mathbf{x}_1$ for each unit change in $\mathbf{x}_2$. If we distribute $\mathbf{x}_1$ into its coefficient, we see that this model is identical to our standard interaction model $y = \beta_1\mathbf{x}_1 + \beta_2\mathbf{x}_2 + \beta_{12}\mathbf{x}_1{:}\mathbf{x}_2$.

Linear models can have higher-order interactions, like the following model with all possible two- and three-way interactions.

$$\begin{aligned}\mathbf{y} = \beta_0 &+ \beta_1\mathbf{x}_1 + \beta_2\mathbf{x}_2 + \beta_3\mathbf{x}_3 \\ &+ \beta_{12}\mathbf{x}_1{:}\mathbf{x}_2 + \beta_{13}\mathbf{x}_1{:}\mathbf{x}_3 + \beta_{23}\mathbf{x}_2{:}\mathbf{x}_3 \\ &+ \beta_{123}\mathbf{x}_1{:}\mathbf{x}_2{:}\mathbf{x}_3 + \boldsymbol{\epsilon}\end{aligned}$$

A model with n inputs has 2^n possible parameters (including the intercept and main effects) if all interactions are considered. Fortunately, there is rarely a need for higher-order interactions. A three-way interaction term measures the effects of the three inputs that cannot be explained by the main effects or the two-way interactions among the three inputs. It is rare to see three inputs interact in a way that cannot be explained by pairwise interactions. In the statistical literature, the rarity of significant higher-order interactions is called the *hierarchical ordering principle*.